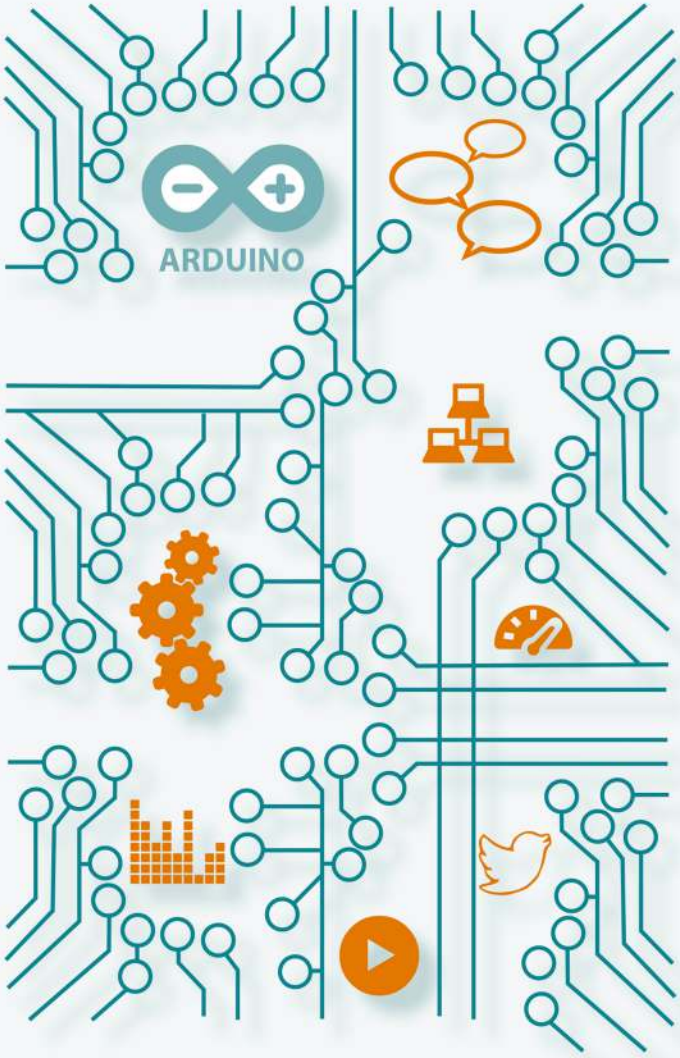
Rubén Oliva Ramos

MONITOREO, CONTROL Y ADQUISICIÓN DE DATOS CON ARDUINO Y VISUAL BASIC .NET



Contenidos Interactivos Web

Alfaomega

Monitoreo, control y adquisición de datos con Arduino y Visual Basic.NET

Rubén Oliva Ramos

Alfaomega

Director Editorial:
Marcelo Grillo Giannetto
mgrillo@alfaomega.com.mx

Jefe de Edición:
Francisco Javier Rodríguez Cruz
jrodriguez@alfaomega.com.mx

Datos catalográficos
Oliva, Rubén
Monitoreo, control y adquisición de datos con Arduino y Visual Basic.NET
Primera Edición
Alfaomega Grupo Editor, S.A. de C.V., México
ISBN: 978-607-622-757-2
Formato: 16.8 x 23 cm Páginas: 372

Monitoreo, control y adquisicón de datos con arduino y Visual Basic.NET
Rubén Oliva Ramos

Primera edición: Alfaomega Grupo Editor, México, febrero 2017

Dr. Isidoro Olvera (Eje 2 sur) No. 74, Col. Doctores, 06720. Ciudad de México

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-622-757-2

Acerca del autor

Rubén Oliva Ramos

Especialista en desarrollo de software para Arduino y Raspberry Pi con .NET; así como en plataformas de uso libre, desarrollo de aplicaciones para control y monitoreo de procesos a distancia, aplicaciones móviles para monitoreo y control, Sistemas SCADA, tecnologías Android, iOS, Windows Phone, Sitios web y Servicios web aplicados a las plataformas de desarrollo de Arduino y Raspberry Pi con diferentes Frameworks y aplicaciones para la nube (cloud services).

- Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de León.
- Maestro en Ingeniería de Sistemas Electrónicos y Computacionales por la Universidad de la Salle Bajío en León, Guanajuato.
- Especialista en teleinformática y redes por la Universidad de la Salle Bajío en León, Guanajuato.
- Docente a nivel bachillerato en el Centro de Bachillerato Tecnológico Industrial y de Servicios No. 225 (CBTis 225) en León, Guanajuato, en la especialidad de Mecatrónica y Programación, dando cursos de electrónica, automatización, microcontroladores, robótica y control, desarrollo de aplicaciones con Android, bases de datos y plataformas web para monitoreo y control; lo cual ha llevado a cabo desde el año 2011 hasta la actualidad.
- Desde el 2008 hasta la actualidad ha sido docente en la Universidad de La Salle Bajío a nivel posgrado en la Especialidad en Mecatrónica y en la maestría en Diseño e Ingeniería de Sistemas Mecatrónicos, donde también participó en el diseño curricular y en la actualización del plan de estudios; en esta misma institución, además, imparte clases a nivel licenciatura.
- Participación en proyectos de investigación y de vinculación con empresas a través del sistema conacyt en colaboración con la Universidad de la Salle Bajío.
- Desarrollo de múltiples prototipos de sistemas mecatrónicos y robóticos.
- Fundador de Edu-training, donde actualmente brinda capacitación y consultoría en cursos de formación en las áreas de automatización y electrónica, sistemas de monitoreo a distancia y tecnología aplicada.
- En el 2014 fue coordinador de cursos de educación continua en las áreas de mecatrónica, dispositivos de control y automatización en la Universidad de la Salle Bajío.
- Del 2009 al 2011 llevó a cabo la tarea de director de la Facultad de Ingeniería Mecatrónica en la Universidad de León.

Dedicatoria

A Dios por darme la sabiduría y el ejemplo para seguir y cumplir mis sueños, sin Él no los podría llevar a cabo; gracias por darme todo lo que tengo y soy.
A mi esposa Mayte y a mis hijos Rubén y Darío; gracias por sus vidas, por su comprensión y amor. Los amo mucho.
A mis papás Rubén y Rosalía por su gran ejemplo y dedicación a lo largo de mi vida; muchas gracias por ser mis papás. Los quiero mucho.
A mis hermanos Tomás y Rosalía; gracias por su comprensión y apoyo.

Contenido

Plataforma de contenidos interactivos

Para tener acceso al material de la plataforma de contenidos interactivos del libro: *Monitoreo, control y adquisiciÆn de datos con Arduino y Visual Basic.NET*, 1a. edición, siga los siguientes pasos:

1. Ir a la página: http://libroweb.alfaomega.com.mx

2. Ir a la sección Catálogo y seleccionar la imagen de la portada del libro, al dar doble clic sobre ella, tendrá acceso al material descargable.

NOTA: Se recomienda respaldar los archivos descargados de las páginas web en un soporte físico.

Introducción

Capítulo 1. Bienvenido a Arduino y Visual Basic .NET. A lo largo de este capítulo se hace una introducción al manejo de la plataforma de Arduino, a las instrucciones básicas de programación, a las configuraciones para preparar el sistema y a la comunicación para poder llevar a cabo los proyectos que se presentan en los siguientes capítulos.

Capítulo 2. Aspectos generales de la programación en Visual Basic .NET. En este apartado se explica la programación en Visual Basic .NET requerida para realizar las interfaces gráficas hombre-máquina; además, se realizará un programa con el que se comunicará Arduino con una pantalla hecha en .NET.

Capítulo 3. Estación meteorológica de monitoreo con Arduino y Visual Basic .NET. Bajo la guía de esta sección se conectará un sensor de temperatura y humedad DTH11 a una fotorresistencia, esto con el objetivo de realizar una estación meteorológica y enviar los datos capturados a una pantalla conectada a una PC a través del puerto serial. Además, se enlazará una pantalla LCD 20x4 y el módulo I2C para ejecutar la interfaz de la pantalla con la placa de Arduino, con lo cual se visualizarán constantemente los datos.

Capítulo 4. Detección de presencia inalámbrica con módulos XBEE. A lo largo de este capítulo se explica el uso de los módulos de comunicación XBEE para realizar una alarma inalámbrica integrando las tecnologías de Arduino y Visual Basic .NET; con ello, se buscará conectar un sensor de movimiento PIR y efectuar la conexión de los módulos de comunicación con el software de la interfaz.

Capítulo 5. Control de las luces desde una interfaz HMI. Esta sección abarca el control de dispositivos desde la interfaz, gracias a lo cual se enlazará un módulo de relevador al Arduino que se podrá controlar desde la interfaz HMI; por otro lado, se podrán también controlar dispositivos de potencia, tal como la conexión de una lámpara .

Capítulo 6. Control de un motor de corriente directa. En este capítulo se conectará un motor de corriente directa al Arduino a través de un puente H para controlar el giro y la velocidad; con ello, se realizará la interfaz requerida para el control del dispositivo.

Capítulo 7. Sistema de alarma inalámbrica. Este capítulo explica cómo realizar una alarma a través de comunicación inalámbrica; asimismo se llevará a cabo el diseño de la interfaz web para monitoreo del sistema.

Capítulo 8. Estación de Registro de datos. En este capítulo se integran los conocimientos de los proyectos realizados previamente, aquello se efectúa a través de un registro de datos de forma local. Por otro lado, se desarrollará como proyecto integrador un registro de información (*datalogger*) en una base de datos y se realizará un reporte para consultarlo en Excel.

Capítulo 9. Desarrollo de proyectos del Internet de las cosas basados en el Shield Ethernet de Arduino. En éste se desarrollarán y aplicarán proyectos adaptados al Internet de las cosas para alcanzar la comprensión de las tecnologías enfocadas en determinadas áreas.

Capítulo 10. Prototipo de un Sistema de Control Supervisorio y Adquisición de datos a distancia (SCADA). Apartado en el cual se aplicarán los conceptos de los sistemas de monitoreo y control, desarrollando una aplicación de un sistema SCADA utilizando la tecnología .NET y la plataforma de Arduino.

Capítulo 11. Rastreador móvil por medio de GSM/GPRS y GPS. Gracias a este capítulo se desarrollará un proyecto de tecnologías aplicadas para sistemas web en la nube con el uso de la placa de GSM/GPRS y GPS; igualmente, se podrá elaborar un prototipo de un tracker móvil con tecnología de programación y Arduino con el objetivo de aplicarlo en un ejemplo real y práctico.

Capítulo 12. Robot controlado inalámbricamente. En esta sección se aplicará todo lo aprendido para desarrollar un prototipo de robot controlado inalámbricamente con las tecnologías de Arduino y diferentes aplicaciones de robótica; ello con el objetivo de que se puedan ejecutar posteriormente proyectos mecatrónicos propios.

Capítulo 1

Bienvenido a Arduino y Visual Basic .NET

Objetivos

En este capítulo se asentarán las bases para utilizar la plataforma Arduino, se describirán las instrucciones básicas de programación y se mencionarán las configuraciones tanto para preparar el sistema como para establecer la comunicación, esto con la finalidad de que se sienten los cimientos para realizar los proyectos que se presentan en los siguientes capítulos.

1.1 Introducción

Este capítulo es una introducción a Arduino y a la programación entorno a él necesaria e importante para establecer una comunicación serial con los diferentes dispositivos externos para el control y la adquisición de datos. Dicha comunicación serial será fundamental en los proyectos de los siguientes capítulos. A continuación se explicará cómo se lleva a cabo la comunicación serial entre Visual Basic .NET y la tarjeta Arduino UNO, para enlazar los controles de los formularios y el aspecto gráfico con el hardware.

1.2 Entorno de programación de Arduino IDE

Arduino es un sistema de desarrollo para microcontroladores de la firma ATMEL. Fue desarrollado en Italia y está compuesto por un software editor-compilador (basado en Processing) en donde se escribe un programa en lenguaje C (basado en Wiring), así como un hardware que consiste en un microcontrolador ATMEL, el cual tiene precargado un sistema operativo (Bootstrap) que permite su programación directa *in-circuit* a través de señales seriales de comunicación.

1.2.1 Funciones básicas iniciales

Función Setup. Contiene todas las configuraciones iniciales del programa. Aquí se incluyen las condiciones iniciales para la operación de algunas instrucciones o librerías que se agregarán en el programa.

Función Loop. En ella se colocan todas las instrucciones que van a realizarse en forma repetitiva (*loop* significa lazo o bucle).

Además de estas funciones, se pueden agregar otras creadas por el usuario, aunque las anteriores no pueden omitirse en el programa. La estructura general de una función se describe en el ejemplo 1.1:

Ejemplo 1.1 Estructura general de una función.

```
void setup( )
{
 Instrucciones;
}
void loop( )
{
Instrucciones;
}

void usuario( )
{
Instrucciones;
}
```

Aquí se puede observar que cada función inicia con void, seguida del nombre de la función. En el caso de setup y loop, no es posible cambiar su nombre. En el caso de una función de usuario, es posible colocar cualquier nombre, empleando letras, números, guiones medio y bajo, aunque no acepta signos ni caracteres especiales. Seguido del nombre de la función se colocan los paréntesis (( )). Entre las llaves ({ }) se colocan los comandos o instrucciones; al final de cada una, se debe agregar el punto y coma (;).

Es importante que en cada función e instrucción se respeten las mayúsculas y minúsculas, así como los espacios y símbolos que deben emplearse; de lo contrario, aparecerá un código de error.

1.2.2 Agregar comentarios

Es posible agregar comentarios a nuestro programa. Para ello, se colocan dos diagonales (//) seguidas del texto con el comentario que se desea mostrar. Pueden colocarse en cualquier parte del programa. (Ver ejemplo 1.2.)

Ejemplo 1.2 Agregar comentarios al programa.

```
Void setup( ){
pinMode(buttonPin, INPUT);    // la variable buttonPin es
configurada como entrada.
pinMode(ledPin, OUTPUT);// la variable ledPin es configura-
da como salida.
}
```

También es posible escribir un bloque de texto, colocando diagonal y asterisco (/*) al inicio del bloque y asterisco diagonal (*/) al final del bloque. (Ver ejemplo 1.3.)

Ejemplo 1.3 Escribir bloques de texto.

```
/*
Programa de Ejemplo para activar un led
En intervalos de 0.s segundos Empleando el led integrado
a la tarjeta Arduino Duemilanove (pin 13)
*/
```

1.3 Conceptos básicos de programación

En esta sección se revisarán algunos conceptos básicos de programación.

1.3.1 Declaración de variables y constantes

Una regla básica para emplear variables y constantes es que siempre deben declararse para que puedan utilizarse. También es importante indicar el tipo de variable según su

formato numérico. En función de su capacidad numérica, las variables y constantes pueden ser de tipo:

Byte. Valores numéricos con capacidad de 8 bits (0-255). (Ver ejemplo 1.4.)

Ejemplo 1.4 Declaración de variables tipo byte.

```
// Asigna a la variable tiempo una longitud de byte (0-
255) sin valor inicial.
byte tiempo;
// Asigna a la variable temperatura una longitud de byte
(0-255) iniciando con el valor de 0.
byte temperatura = 0;
```

Int. Entero. Valores numéricos con capacidad de 16 bits con signo (-32768 y 32767). (Ver ejemplo 1.5.)

Ejemplo 1.5 Declaración de variables tipo int.

```
// Asigna a la variable como_se_llame, una longitud de
int, sin valor inicial.
int como_se_llame;

// Asigna a la variable como_se_llame, una longitud de
int, iniciando con el valor de 0.
int como_se_llame  = 1765;
```

Long. Extendido. Valores numéricos enteros con capacidad de 32 bits (-2147483648 a 2147483647). (Ver ejemplo 1.6.)

Ejemplo 1.6 Declaración de variables tipo long.

```
// Asigna a la variable cualquier_nombre, una longitud de
long, sin valor inicial.
long cualquier_nombre;

// Asigna a variable cualquier_nombre, una longitud de
long, iniciando con el valor de 0.
long cualquier_nombre  = 150000;
```

Float. Flotante. Valores numéricos con fracción decimal con capacidad de 32 bits (3.4028235E +38 y 3.4028235E -38). Es importante destacar que los resultados de las operaciones matemáticas sólo muestran dos decimales con redondeo. (Ver ejemplo 1.7.)

Ejemplo 1.7 Declaración de variables tipo float.

```
// Asigna a la variable ponle_un_nombre, una longitud de
int, sin valor inicial.
float ponle_un_nombre;

// Asigna a variable ponle_un_nombre, una longitud de
int, iniciando con el valor de 0.
float ponle_un_nombre  = 3.14;
```

1.3.2 Arreglos con variables

También es posible hacer arreglos con variables. De esta forma, se pueden asignar variables con el mismo nombre pero asignando una posición; o bien, realizar matrices de valores indicando el tipo de dato, el tamaño y asignar valores a una posición específica. (Ver ejemplo 1.8.)

Ejemplo 1.8 Arreglos con variables.

```
/*
En este arreglo, el primer valor se encuentra en la posi-
ción 0 de mi variable, el segundo valor, en la posición
1, el tercer valor, en la posición 2, etcétera.
*/

int mi_arreglo[]  = {10, 50, otra_variable}

En donde:

mi_arreglo[0]  = 10
mi_arreglo[1]  = 50
mi_arreglo[2]  = lo que vale otra variable
```

De igual forma, es posible declarar una matriz de valores indicando el tipo de datos y el tamaño, y, posteriormente, asignar valores a una posición específica. (Ver ejemplo 1.9.)

Ejemplo 1.9 Declarar matrices de valores.

```
int mi_arreglo[5];        // declara un arreglo de ente-
ros de 6 posiciones (0 - 5).
mi_arreglo[3]  = 10;   // asigna el valor 10 a la posi-
ción 4 del arreglo.
```

Recordemos que el valor que se coloca entre corchetes ([ ]) representa el máximo valor de elementos en el arreglo, comenzando desde el cero. El cero cuenta como la posición uno.

Es posible asignar a una variable una determinada terminal de la tarjeta, definida como *entrada* o *salida*, tanto analógica como digital. (Ver ejemplo 1.10.)

Ejemplo 1.10 Asignar una determinada terminal de la tarjeta a una variable.

```
Int ledPin = 13;                                    // asigna
la variable ledPin a la terminal (pin) 13 de la tarjeta
Arduino.
Int Valor_Analógico  =  0;          // asigna a la varia-
ble Valor_Analógico al pin 0 de la tarjeta.
```

1.3.3 Operaciones aritméticas

Arduino puede manejar las cuatro operaciones aritméticas básicas: suma, resta, multiplicación y división. Para efectuar cualquiera de estas operaciones, es posible emplear números directos o variables. Cuando se emplean variables, es importante declarar cada variable con el formato numérico deseado (int, long, float, etc.). Hay que recordar que el único formato que puede realizar operaciones con valores decimales es el float. (Ver ejemplo 1.11.)

Ejemplo 1.11 Operaciones aritméticas.

```
X = y + 5;
Z = x - 30;
P = valor * 50;
R = valor1/valor2;
```

Si se requiere realizar alguna ecuación, se escribe la operación entre paréntesis (( )). (Ver ejemplo 1.12.)

Ejemplo 1.12 Ecuaciones.

```
Temperatura = (valor_Analógico * 500) / 1023
```

1.3.4 Asignaciones compuestas

Las asignaciones compuestas combinan una operación aritmética con una variable asignada. Éstas comúnmente se utilizan en las condiciones de ciclos, tal como se describe más adelante (ver ejemplo 1.13). Estas asignaciones compuestas pueden ser las siguientes:

```
x ++    // igual que x = x + 1, o incrementar x en + 1
x --    // igual que x = x - 1, o decrementar x en -1
x += y    // igual que x = x + y, o incrementar x en +y
x -= y   // igual que x = x - y, o decrementar x en -y
x *= y   // igual que x = x * y, o multiplicar x por y
x /= y   // igual que x = x / y, o dividir x por y
```

Ejemplo 1.13 Asignaciones compuestas.

```
x * = 3 hace que x se convierta en el triple del antiguo
valor x y, por lo tanto, x se reasigna al nuevo valor.
```

1.3.5 Operadores de comparación

Las comparaciones entre variables y constantes se utilizan con frecuencia en las estructuras condicionales if para establecer si una condición es verdadera, como se mostrará más adelante. Los operadores de comparación en Arduino son los siguientes:

```
x == y              // x es igual a y
x != y               // x no es igual a y
x < y            // x es menor que y
x > y           // x es mayor que y
x <= y  // x es menor o igual que y
x >= y  // x es mayor o igual que y
```

1.3.6 Operadores lógicos

Comúnmente, los operadores lógicos son una forma de comparar dos expresiones y dar como respuesta un estado *verdadero* o *falso*, dependiendo del operador lógico utilizado. Existen tres operadores lógicos: AND (&&), OR (||) y NOT (!), los cuales se utilizan generalmente en condicionales de tipo if. A continuación se muestran algunos ejemplos. (Ver ejemplo 1.14.)

Ejemplo 1.14 Ejemplos de operadores lógicos AND, OR y NOT.

Lógica AND:
if (x > 0 && x < s) // condición es verdadera sólo si las dos expresiones son ciertas

Lógica OR:
if (x > 0 || y > 0) // condición es verdadera si cualquiera de las expresiones es cierta

Lógica NOT:
if (!x > 0) // condición es verdadera solo si la expresión (x > 0) es falsa

1.3.7 Constantes

El lenguaje de programación de Arduino contiene valores predeterminados, que son llamados "constantes". Estos valores, realizan funciones específicas que pueden ser empleadas, principalmente, por valores digitales, o bien, por valores numéricos.

En el caso de una constante numérica, se antepone la palabra *const*, seguida de su formato numérico (int, float, byte, etc.). Finalmente, se le asigna un valor. (Ver ejemplo 1.15.)

Ejemplo 1.15 Ejemplos de constantes.

```
const int nombre_variable = 10;    // asigna el valor de
10 a la constante declarada
const float b = 3.14;// asigna el valor de 3.14 a la cons-
tante llamada b
const int ledPin = 13;// asigna el valor de 13 a la cons-
tante ledPin
```

Cuando se emplean valores digitales –también llamados valores booleanos–, se pueden utilizar las siguientes constantes asignadas, que la programación Arduino establece directamente.

TRUE/FALSE

FALSE se define como un valor de cero. TRUE se asocia con un valor de 1; sin embargo, cualquier entero que es no-cero es true, en un sentido booleano. Es decir, -1, 2 y -200 son todos true. Es importante considerar que las constantes TRUE y FALSE se escriben en mayúsculas. (Ver ejemplo 1.16.)

Ejemplo 1.16 Constantes TRUE y FALSE.

```
if (x == TRUE);
{
ejecutar estas instrucciones;
}
```

INPUT/OUTPUT

Estas constantes se emplean al inicio del programa —dentro de la función void_setup–, para definir el sentido de una señal digital, si es de entrada —INPUT—o si es de salida —OUTPUT–. Estas constantes deberán escribirse siempre en mayúsculas. Para que puedan operar, se requiere anteponer la instrucción *pinMode*, seguida del valor de la terminal –pin– que se desea direccionar separado éste por una coma (,) del sentido de la señal; todo esto, entre paréntesis (()). (Ver ejemplo 1.17.)

Ejemplo 1.17 Constantes INPUT y OUTPUT.

```
pinMode (13,OUTPUT);  // asigna el pin 13 del Arduino
como una salida
```

También es posible que el pin sea asignado a través de una variable previamente definida. (Ver ejemplo 1.18.)

Ejemplo 1.18 Pin asignado a través de una variable previamente definida.

```
Int ledPin = 13;                                    //
asigna a la variable ledPin el valor de 13
pinMode(ledPin, OUTPUT);    // asigna el valor de la va-
riable ledPin como una salida
```

HIGH/LOW

Estas constantes también se escriben en mayúscula y establecen el estado lógico directamente de un pin de entrada/salida. HIGH establece que el nivel lógico es de 1; LOW establece que el nivel lógico es de 0. Estas constantes forman parte de las funciones de lectura –digitalRead– o escritura –digitalWrite– de valores digitales en los pines de entrada/salida de Arduino. (Ver ejemplo 1.19.)

Ejemplo1.19 Constantes HIGH y LOW.

```
digitalWrite(ledPin, HIGH); // envía a un estado lógico 1
al pin establecido en la variable
digitalWrite(ledPin, LOW);             // envía a un es-
tado lógico 0 al pin establecido en la variable
```

1.3.8 Estructuras de control

A continuación se revisarán algunas estructuras de control y sus formatos de instrucción.

Condicional if

If es una instrucción que se utiliza para determinar si una condición se ha cumplido; por ejemplo, averiguar si un valor analógico está por encima de un cierto número. Si la condición se cumple, deberá ejecutar una serie de instrucciones que se escriben dentro de llaves ({ }). Si no se cumple, el flujo del programa salta esta condición y no ejecuta las operaciones que están dentro de las llaves.

El formato de instrucción es el siguiente:

```
if (Variable Condición Valor)
{
Instrucciones a ejecutar;
```

En donde:

Variable es un valor o dato que se desea condicionar. Puede estar dado en cualquier formato para valores numéricos.

Condición es cualquiera de los operadores de comparación (==, !=, >, <, =>, =<).

Valor es el dato numérico (valor directo o variable) que se desea comparar. (Ver ejemplo 1.20.)

Ejemplo 1.20 Ejemplos de condicional if.

```
If (temperatura >= 30)      // si la variable temperatura
es mayor o igual a 30:
{
digitalWrite(ledPin, HIGH);    // activa el led conectado
en el pin designado (ledpin).
}


If (valor_actual == valor_establecido)   // si la varia-
ble valor_actual es igual a variable
{       // valor_establecido
Serial.print("Valor Alcanzado");   // envía por el serial
del Arduino el texto.
}       // "Valor Alcanzado".



If (x != 50)   // si la variable x no es igual a s0
{
digitalWrite(10, LOW)                  // apaga el pin
10.
}
```

Condicional if else

If else es una instrucción que se emplea para definir lo que deberá realizarse en el programa en caso de que una determinada condición no se cumpla. Esto equivale a decir: "Si se cumple esta condición, realiza estas operaciones; de lo contrario (else), realiza éstas". (Ver ejemplo 1.21.)

El formato de instrucción es el siguiente:

```
if (variable condición valor)
{
Instrucciones a ejecutar;
```

```
}
else
{
Instrucciones a ejecutar;
}
```

Ejemplo 1.21 Condición if else.

```
if (boton == HIGH) { // si el estado contenido en la
variable botón esta en alto
digitalWrite(13, HIGH);     //entonces, activa el pin 13.
}
else {  // de lo contrario,
digitalWrite(13, LOW);      // apaga el pin 13.
}
```

Ciclo for

Este comando realiza un cierto número de veces un ciclo repetitivo de las operaciones que se encuentran dentro de él. Una vez que termina el ciclo, el programa continúa ejecutando las instrucciones fuera de éste. Si se quiere repetir el ciclo, se puede recomenzar.

El formato de instrucción es el siguiente:

```
for (Inicialización; Condición; Expresión)
{
Instrucciones a ejecutar;
}
```

En donde:

Inicialización es el valor inicial dado a una variable. Éste es el valor de inicio del ciclo.

Condición es el operador de comparación que indica hasta dónde puede llegar el valor de la variable (valor final).

Expresión indica si el conteo de los ciclos dentro del valor inicial y final, se incrementa o decrementa. Aquí se emplea una asignación compuesta (++, --). (Ver ejemplo 1.22.)

Ejemplo 1.22 Ciclo for.

```
for (int x = 0; x < 12; x++)            // x vale desde 0
hasta 12 y se incrementa de 1 en 1.
{
```

```
Serial.println(x);   // escribe al serial el valor que
toma x
delay(500);    // espera 0.s segundos antes de enviar
otro valor
}
```

Es importante observar en el ejemplo 1.22 que la variable x se puede declarar dentro de la instrucción como un valor int (entero), o bien, puede ser previamente declarada al inicio del programa, con cualquier formato numérico.

El ejemplo 1.23 muestra cómo podemos realizar ciclos en los que se incrementen de 2 en 2 los valores declarados en la función.

Ejemplo 1.23 Ciclos en los que se incrementan de 2 en 2 los valores declarados en la función.

```
for (int x = 0; x < 12; x+=2)              // x vale
desde 0 hasta 12 y se incrementa de 2 en 2.
{
Serial.println(x);   // escribe al serial el valor que
toma x.
delay(500);    // espera 0.s segundos antes de enviar
otro valor.
}
```

En el ejemplo 1.24 se observa cómo realizar un ciclo negativo de valores –decremento– de 1 en 1.

Ejemplo 1.24 Ciclo negativo de valores.

```
for (int x = 10; x > -1; x--)              // x vale
desde 10 hasta 0 y se incrementa de 1 en 1.
{
Serial.println(x);  // escribe al serial el valor que
toma x.
delay(s00); // espera 0.s segundos antes de enviar otro
valor.

}
```

Comparador múltiple Switch Case

Esta instrucción relaciona los valores de una variable con diferentes condiciones; las compara y realiza las instrucciones indicadas para cada caso (case). Al finalizar las instrucciones que deberán ejecutarse en cada uno, se coloca una instrucción *break*, necesaria para que el programa pueda seguir su flujo normal. Al final del último case, se coloca el comando

default, para indicar que, si no se cumplió ninguno, el programa continúa en la siguiente línea. (Ver ejemplo 1.25.)

El formato de instrucción es el siguiente:

```
switch (var) {
case etiqueta: instrucciones; break;
case etiqueta: instrucciones;
break;
default:
instrucciones;
}
```

Ejemplo 1.25 Comparador múltiple Switch Case.

```
switch (var) {
case 1:                                    // cuando valor
de var sea igual a 1:
digitalWrite(13, HI  H);
break;
case 2:                                   // cuando valor de
var sea igual a 2:
digitalWrite(13, LOW);
break;
}
```

Ciclo while

Los ciclos while se ejecutan continuamente hasta que la expresión dentro del paréntesis (condición de comparación) deja de cumplirse. Algo debe modificar la variable comprobada; de lo contrario, el ciclo while nunca terminará. Lo que modifique la variable puede estar dentro del código, como una variable que se incrementa, o ser una condición externa; por ejemplo, el valor de un sensor.

El formato de instrucción es el siguiente. (Ver ejemplo 1.26):

```
while(expresión)
{
instrucciones;
}
```

Ejemplo 1.26 Ciclo while.

```
while(var <  200) { // mientras que var sea menor que 200:
Serial.pritnln(var); // envía por el serial, el valor de
var
    var++;     // incrementa en 1 el valor de var
}
```

Ciclo do while

El ciclo do while ejecuta las instrucciones especificadas, mientras que la condición indicada dentro del paréntesis (condición de comparación) se cumpla. Cuando la condición se cumple, se repite el ciclo.

El formato de instrucción es el siguiente. (Ver ejemplo 1.27):

```
do
{
Instrucciones;
}
while (condición);
```

Ejemplo 1.27 Ciclo do while.

```
{
delay(50);   // espera a que los sensores se estabilicen
x =
readSensors(); // comprueba los sensores

} while (x < 100);          //si se cumple la condición se
repite el bucle
```

Instrucción continue

Cuando se escribe esta instrucción dentro de un ciclo (do, for o while), el programa se "salta" éste y se dirige a las siguientes instrucciones fuera del ciclo. (Ver ejemplo 1.28.)

Ejemplo 1.28 Instrucción continue.

```
for (x = 0; x < 2ss; x ++)
{
if (x > 20 && x < 100){ // Si se cumple esta condición:
continue;      // se salta desde el 21 hasta el 99 del
ciclo for
}

analogWrite(10, x);
delay(s0);
}
```

1.4 Tipos de comunicación

Uno de los objetivos de este libro es explicar la comunicación serial con el Arduino, puesto que se trata de una parte fundamental, además de ser una herramienta de suma importancia para llevar a cabo los proyectos que se presentan en los siguientes capítulos. En este punto hablaremos de la comunicación Serial UART y la comunicación serial por software a través de la librería SoftwareSerial.

1.4.1 Comunicación Serial UART

Se utiliza para la comunicación entre la tarjeta Arduino y una PC u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie (también conocido como un UART o USART). Éste utiliza los pines 0 (RX) y 1 (TX), y se comunica con el ordenador a través de USB. Los pines digitales están reservados para esta función.

Se puede usar el monitor serial incorporado en el entorno de Arduino para comunicarse con una placa Arduino. Para ello, hacer clic en el botón de monitor de serie en la barra de herramientas y seleccionar la misma velocidad de transmisión que se ha programado en las funciones de velocidad de transmisión. (Ver figura 1.1.)

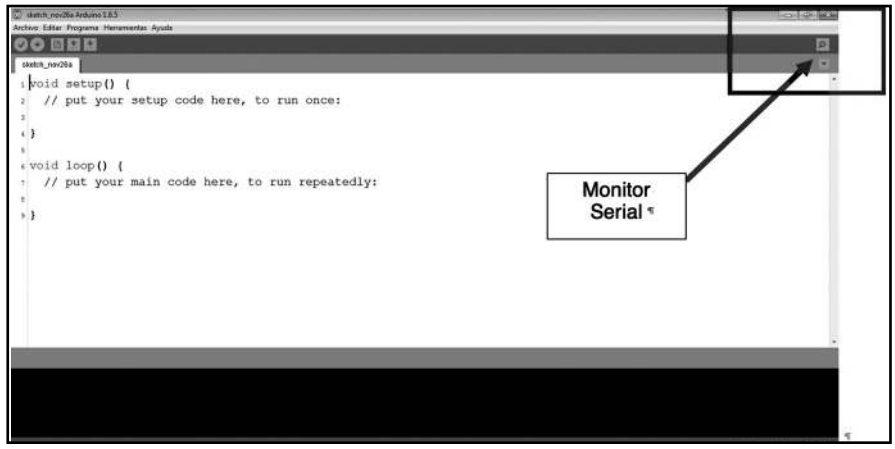


Figura 1.1 Comunicación del monitor serial con la placa Arduino.

Funciones de comunicación serial

Al emplear la comunicación serial, es importante considerar que los pines 0 (Rx) y 1 (Tx) de la placa Arduino se reservan para esta función. También es importante recordar que este puerto se utiliza para la programación del Arduino; sin embargo, mientras se emplea la función serial, no hay interferencia entre ambas operaciones.

Las funciones relacionadas con la comunicación serial son las siguientes:

```
Serial.begin( )
Serial.end( )
Serial.available( )
```

```
Serial.read( )
Serial.flush( )
Serial.print( )
Serial.println( )
Serial.write( )
```

Función Serial.begin( );

En esta función se abre el puerto serial del Arduino y se determina la velocidad de transferencia en bps (bps = bits por segundo) de datos entre dispositivos. Los valores más comunes pueden ser 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 o 115200. Esta función deberá colocarse dentro de la rutina void setup(). (Ver ejemplo 1.29.)

Ejemplo 1.29 Función Serial.begin( );

```
Serial.begin(9600);     // configura la comunicación
serial a 9600 bps.
```

Función Serial.end();

Desactiva la comunicación serial, permitiendo a los pines 0 (Rx) y 1 (Tx) utilizarse como entradas o salidas digitales. Esta función se coloca dentro de cualquier rutina en el lugar o instante en que se desea desactivar la comunicación. Para volver a reactivar la comunicación, es necesario volver a colocar en el lugar o instante deseado la función Serial.begin(bps), y así volver a inicializar la transferencia de datos.

Función Serial.available();

Esta función cuenta y guarda el número de bytes disponibles para que el puerto serie los lea. Esto se refiere a los datos que ya han sido recibidos y están almacenados en el búfer de recepción del puerto, el cual tiene una capacidad de hasta 128 bytes. Más adelante se realizará un ejercicio para ejemplificar esta función.

Función Serial.read();

Lee los datos (byte) que son recibidos por el puerto serial. Esta función requiere de una variable para cargar el valor del dato recibido por la función. Si la variable se declara como int, el valor que recibe la función es el equivalente al valor decimal del valor ASCII enviado. Si la variable se declara como char, se recibe el valor directo del valor ASCII enviado. (Ver ejemplo 1.30.)

Ejemplo 1.30 Función Serial.read();

```
datoRecibido  =  Serial.read();  // guarda el dato leído
por el puerto serial en la variable datoRecibido.
```

Función Serial.flush();

Con esta función se vacía el búfer de entrada de datos del puerto serial.

Función Serial.print();

Imprime –o envía– los datos al puerto serie como texto ASCII. Esta función puede tomar diversas opciones. Los números se imprimen mediante un juego de caracteres ASCII para cada dígito. Los valores de tipo float se imprimen en forma de dígitos ASCII con dos decimales por defecto. Los valores tipo byte se envían como un único carácter. Los caracteres y las cadenas se envían como son. (Ver ejemplo 1.31.)

Ejemplo 1.31 Función Serial.print();

```
Serial.print(15);                    // imprime "15"
Serial.print(3.1416); // imprime "3.1416"
Serial.print(byte(65));    // imprime "A" (cuyo código
ASCII es 65)
Serial.print('A');   // imprime "A"
Serial.print("Hola");      // imprime "Hola"
```

Un segundo parámetro opcional especifica la base (formato a usar). Los valores permitidos son BYTE, BIN (binarios o base 2), OCT (octales o base 8), DEC (decimales o base 10) y HEX (hexadecimales o base 16).

```
Serial.print(65, BYTE);     // imprime "A"
Serial.print(65, BIN);      // imprime "1000001"
Serial.print(65, OCT):      // imprime "101"
Serial.print(65, DEC);      // imprime "65"
Serial.print(65, HEX);      // imprime "41"
```

Para imprimir números enteros o de punto flotante, hay que agregar un valor separado por una coma. Esto especifica el número de dígitos a usar (de izquierda a derecha), empezando en la posición 0. (Ver ejemplo 1.32.)

Ejemplo 1.32 Imprimir números enteros o de punto flotante.

```
Serial.print(3.1416, 0);    // imprime "3"
Serial.print(3.1416, 2);    // imprime "3.14"
Serial.print(3.1416, 4);    // imprime "3.1416"
```

Para agregar un tabulador, hay que agregar un comando "\t" dentro de la función. (Ver ejemplo 1.33.)

Ejemplo 1.33 Agregar un tabulador.Serial.print("Valor A: ");

```
Serial.print("Valor A: "); Serial.print("\t"); // el
siguiente texto se imprime  tabulado.
Serial.print("Valor B: ");
```

Para aplicar un retorno de carro y avance de línea, hay que agregar un comando "\n" dentro de la función. (Ver ejemplo 1.34.)

Ejemplo: 1.34 Aplicar un retorno de carro y avance de línea.

```
Serial.print("Resultado A:");
Serial.print("\n")                          // el siguiente
texto se imprime al inicio de la siguiente línea.
Serial.print("Resultado B:");
```

Función Serial.println();

Imprime o envía los datos al puerto serie como texto ASCII, seguido de un comando de retorno de carro y avance de línea. Para esta función aplican todas las opciones anteriores de Serial.print(). (Ver ejemplo 1.35.)

Ejemplo 1.35 Función Serial.println();

```
Serial.println("Hola");     // imprime "Hola" y envía el
comando de retorno de carro y avance de línea.
```

Función Serial.write();

Esta función es muy similar a Serial.print(), a diferencia de que, si se envían valores numéricos en forma directa, los interpretará como valores ASCII. Se recomienda, entonces, emplear la función Serial.print(). Los comandos "\t" (tabulador) y "\n" (retorno de carro y avance de línea) aplican de igual forma. (Ver ejemplo 1.36.)

Ejemplo 1.36 Función Serial.write();

```
Serial.write(65);     // envía la letra A (ASCII 6s = A).
Serial.write("Hola");   // envía el texto "Hola".
Serial.write("\t");       // aplica tabulador.
Serial.write("\n");     // aplica retorno de carro y
avance de línea.
```

1.4.2 Comunicación Serial por software

El hardware Arduino ha incorporado un soporte para la comunicación en serie entre los pines 0 y 1, que también se conectan a la computadora a través de la conexión USB.

La comunicación serial por software se ha desarrollado para permitir la comunicación en serie en otros pines digitales del Arduino. Es decir, se usa el software para replicar la funcionalidad: es posible tener por software múltiples puertos serie con velocidades de hasta 115200 bps. Un parámetro permite la señalización invertida para dispositivos que requieren dicho protocolo.

1.4.3 Librería SoftwareSerial

La biblioteca cuenta con las siguientes limitantes:

- Si utiliza software de múltiples puertos serie, se reciben los datos uno a la vez.
- No todos los pines funcionan en el Arduino Mega, por lo que solamente se puede utilizar para RX: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).

1.5 Tarjetas Arduino y los puertos de comunicación

En las siguientes secciones se describirán los tipos de tarjetas Arduino y sus puertos de comunicación.

1.5.1 Arduino UNO

Utiliza los pines digitales 0 (RX) y 1 (TX) para establecer comunicación con los dispositivos.

1.5.2 Arduino MEGA

El Arduino Mega tiene tres puertos seriales adicionales: Serial1, en los pines 19 (RX) y 18 (TX); Serial2, en los pines 17 (RX) y 16 (TX); Serial3, en los pines 15 (RX) y 14 (TX). Para utilizar estos pines para comunicarse con la PC, se requiere de un adaptador adicional de USB a serie, ya que no están conectados al adaptador de los Mega-USB a serie.

Para utilizarlos para comunicarse con un dispositivo serie TTL externo, conectar el pin TX al pin RX del dispositivo; el RX a TX pin del dispositivo, y el suelo del Mega a tierra del dispositivo. Estos pines no se deben conectar directamente a un puerto serie RS232, ya que operan a ±12 V y pueden dañar la placa Arduino.

1.5.3 Arduino Due

El Arduino Due tiene tres puertos seriales TTL de 3.3 V Serial1, en los pines 19 (RX) y 18 (TX); Serial2, en los pines 17 (RX) y 16 (TX); Serial3, en los pines 15 (RX) y 14 (TX). Los pines 0 y 1 también están conectados a los pines correspondientes del USB

-to- Serial TTL chip de ATmega16U2, que está conectado al puerto USB de depuración. Además, hay un puerto USB-SERIAL nativo en el chip SAM3X.

1.5.4 Arduino YUN

Utiliza los pines digitales 0 (RX) y 1 (TX) para establecer comunicación con los dispositivos.

Para fines prácticos, en los proyectos se utilizará la tarjeta Arduino UNO para establecer comunicación serial con los diferentes dispositivos.

1.6 Aspectos de comunicación serial con Visual Basic .NET

Para establecer comunicación con una PC a través de una interfaz HMI (interfaz hombre-máquina), se utilizará el lenguaje de programación de Microsoft Visual Basic .NET. Para llevar a cabo la comunicación, se utilizará la clase de .NET SerialPort, que permitirá realizar una comunicación serial con la tarjeta Arduino, con el fin de leer y controlar lo que se le conecte.

1.6.1 Clase SerialPort

El control SerialPort del entorno de Visual Basic .NET permite establecer una comunicación serial con cualquier dispositivo; en este caso, la computadora. Para ello, se debe arrastrar al formulario. (Ver figuras 1.2 y 1.3.)

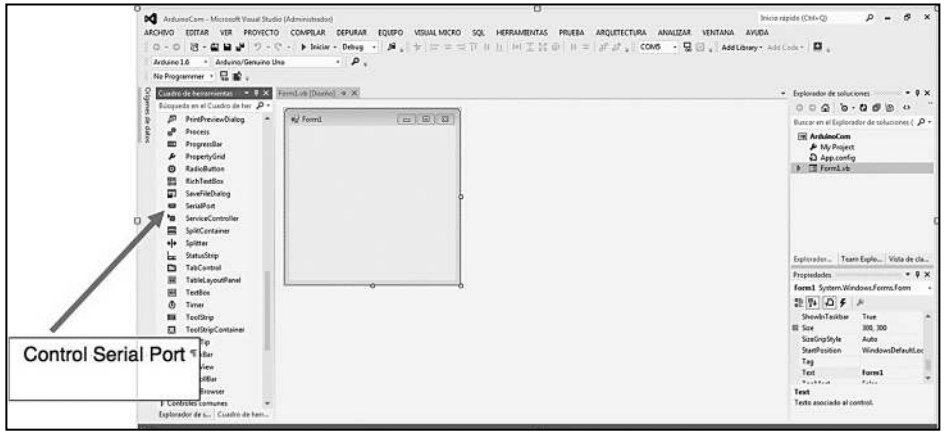


Figura 1.2 Tomar el control SerialPort del menú Cuadro de herramientas.

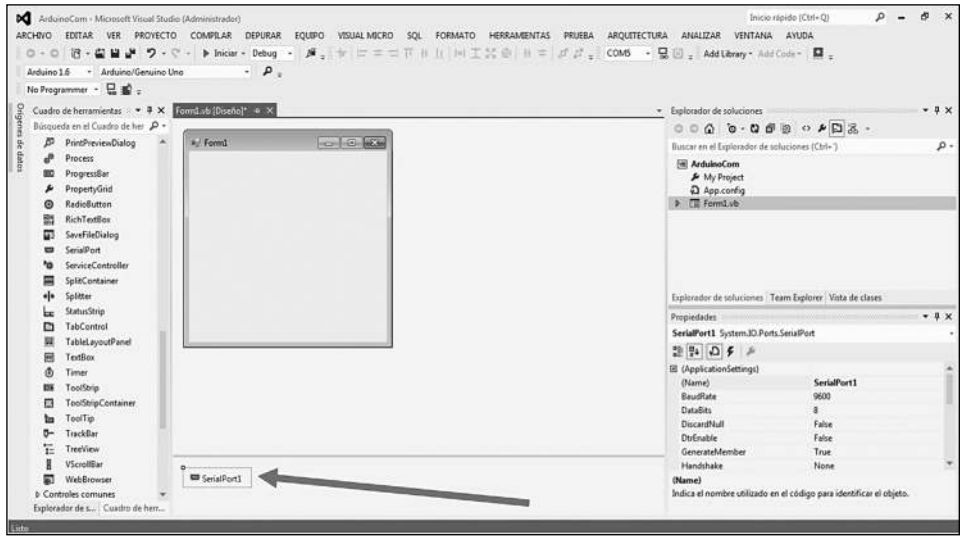


Figura 1.3 Arrastrar el control SerialPort al formulario.

Una vez que se tiene el control serial, se pueden ver sus propiedades, que deben configurarse como se muestra en la figura 1.4.

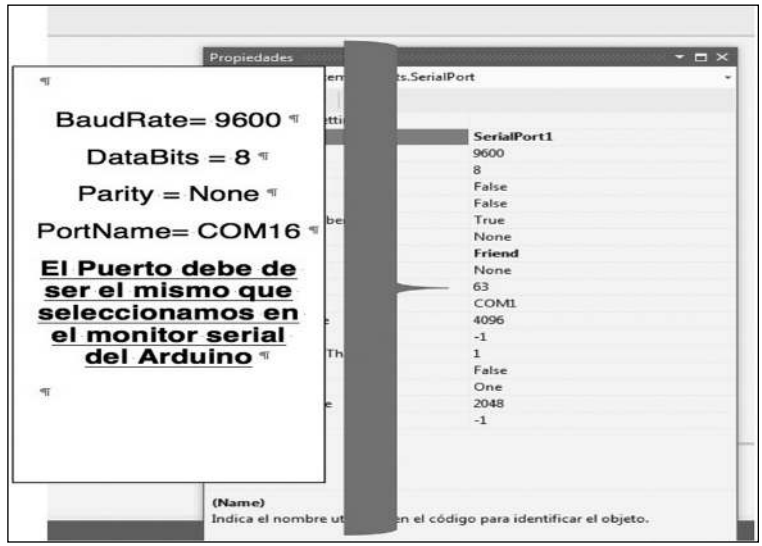


Figura 1.4 Configuración del control SerialPort.

1.6.2 Funciones de comunicación serial de Clase SerialPort

A continuación se describen algunas funciones de comunicación serial de la Clase SerialPort.

Nombre	Descripción
SerialPort()	Inicializa una nueva instancia de la clase SerialPort.
SerialPort(IContainer)	Inicia una nueva instancia de la clase SerialPort utilizando el objeto IContainer especificado.
SerialPort(String)	Inicializa una instancia nueva de la clase SerialPort empleando el nombre de puerto especificado.
SerialPort(String, Int32)	Comienza una instancia nueva de la clase SerialPort con el nombre de puerto y la velocidad en baudios especificados.
SerialPort(String, Int32, Parity)	Inicializa una instancia nueva de la clase SerialPort utilizando el nombre del puerto, la velocidad en baudios y el bit de paridad determinados.
SerialPort(String, Int32, Parity, Int32)	Inicializa una instancia nueva de la clase SerialPort utilizando el nombre del puerto, la velocidad en baudios, el bit de paridad y los bits de datos especificados.
SerialPort(String, Int32, Parity, Int32, StopBits)	Inicializa una instancia nueva de la clase SerialPort utilizando el nombre del puerto, la velocidad en baudios, el bit de paridad, los bits de datos y el bit de parada especificados.

1.6.3 Puertos de comunicación y sus propiedades

A continuación se describen algunos puertos de comunicación.

Nombre	Descripción
BaseStream	Obtiene el objeto Stream subyacente para un objeto SerialPort.
BaudRate	Obtiene o establece la velocidad en baudios del puerto serie.
BreakState	Obtiene o establece el estado de la señal de interrupción.
BytesToRead	Obtiene el número de bytes de datos en el búfer de recepción.
BytesToWrite	Obtiene el número de bytes de datos en el búfer de envío.
CanRaiseEvents	Obtiene un valor que indica si el componente puede generar un evento (heredado de Component).
CDHolding	Obtiene el estado de la línea de detección de portadora para el puerto.
Container	Obtiene IContainer que contiene Component (heredado de Component).
CtsHolding	Obtiene el estado de la línea 'Listo para enviar'.
DataBits	Obtiene o establece la longitud estándar de los bits de datos por byte.
DesignMode	Obtiene un valor que indica si Component está actualmente en modo de diseño (heredado de Component).
DiscardNull	Obtiene o establece un valor que indica si no se tienen en cuenta los bytes nulos en las transmisiones entre el puerto y el búfer de recepción.
DsrHolding	Obtiene el estado de la señal 'Conjunto de datos preparado' (DSR).
DtrEnable	Obtiene o establece un valor que habilita la señal 'Terminal de datos preparada' (DTR) durante la comunicación en serie.
Encoding	Obtiene o establece la codificación de bytes para la conversión de texto previa y posterior a la transmisión.
Events	Obtiene la lista de controladores de eventos asociados a Component (heredado de Component).

Nombre	Descripción
Handshake	Obtiene o establece el protocolo de enlace para la transmisión de datos a través del puerto serie desde Handshake.
IsOpen	Obtiene un valor que indica el estado abierto o cerrado del objeto SerialPort.
NewLine	Obtiene o establece el valor utilizado para interpretar el final de una llamada a los métodos ReadLine y WriteLine.
Parity	Obtiene o establece el protocolo de comprobación de la paridad.
ParityReplace	Obtiene o establece el byte que reemplaza los bytes no válidos en una secuencia de datos cuando se produce un error de paridad.
PortName	Obtiene o establece el puerto de comunicaciones, incluidos por lo menos todos los puertos COM disponibles.
ReadBufferSize	Obtiene o establece el tamaño del búfer de entrada de SerialPort.
ReadTimeout	Obtiene o establece el número de milisegundos que transcurren antes de que se agote el tiempo de espera si una operación de lectura no finaliza.
ReceivedBytesThreshold	Obtiene o establece el número de bytes en el búfer de entrada interno antes de que ocurra un evento DataReceived.
RtsEnable	Obtiene o establece un valor que indica si la señal 'Solicitud de envío' (RTS) está habilitada durante la comunicación en serie.
Site	Obtiene o establece ISite de Component (heredado de Component).
StopBits	Obtiene o establece el número estándar de bits de parada por byte.
WriteBufferSize	Obtiene o establece el tamaño del búfer de salida del puerto serie.
WriteTimeout	Obtiene o establece el número de milisegundos que transcurren antes de que se agote el tiempo de espera si una operación de escritura no finaliza.

1.6.4 Configuraciones iniciales y métodos para ejecutarse

A continuación se describen algunas configuraciones iniciales y los métodos para ejecutarlas.

Nombre	Descripción
Close()	Cierra la conexión del puerto, establece el valor de la propiedad IsOpen en false y elimina el objeto Streaminterno.
CreateObjRef(Type)	Crea un objeto que contiene toda la información relevante necesaria para generar un proxy utilizado para comunicarse con un objeto remoto (heredado de MarshalByRefObject).
DiscardInBuffer()	Descarta los datos del búfer de recepción del controlador serie.
DiscardOutBuffer()	Descarta los datos del búfer de transmisión del controlador serie.
Dispose()	Libera todos los recursos utilizados por Component (heredado de Component).
Dispose(Boolean)	Libera los recursos no administrados que usa SerialPort y libera los recursos administrados de forma opcional (reemplaza a Component.Dispose(Boolean)).
Equals(Object)	Determina si el objeto especificado es igual al objeto actual (heredado de Object).
Finalize()	Libera recursos no administrados y realiza otras operaciones de limpieza antes de que se reclame el objeto Component durante la recolección de elementos no utilizados(heredado de Component).
GetHashCode()	Sirve como una función hash para un tipo en particular (heredado de Object).
GetLifetimeService()	Recupera el objeto de servicio de duración actual que controla la directiva de duración de esta instancia (heredado de MarshalByRefObject).
GetPortNames()	Obtiene una matriz con los nombres de los puertos serie del equipo actual.

Nombre	Descripción
GetService(Type)	Devuelve un objeto que representa el servicio suministrado por Component o por Container (heredado de Component).
GetType()	Obtiene el Type de la instancia actual (heredado de Object).
InitializeLifetimeService()	Obtiene un objeto de servicio de duración para controlar la directiva de duración de esta instancia (heredado de MarshalByRefObject).
MemberwiseClone()	Crea una copia superficial del Object actual (heredado de Object).
MemberwiseClone(Boolean)	Crea una copia superficial del objeto MarshalByRefObject actual (heredado de MarshalByRefObject).
Open()	Abre una nueva conexión de puerto serie.
Read(Byte(), Int32, Int32)	Lee varios bytes del búfer de entrada de SerialPort y los escribe en una matriz de bytes en la posición de desplazamiento especificada.
Read(Char(), Int32, Int32)	Lee un número de caracteres del búfer de entrada de SerialPort y los escribe en una matriz de caracteres en la posición de desplazamiento especificada.
ReadByte()	Lee sincrónicamente un byte del búfer de entrada de SerialPort.
ReadChar()	Lee sincrónicamente un caracter del búfer de entrada de SerialPort.
ReadExisting()	Lee todos los bytes inmediatamente disponibles, basándose en la codificación, en la secuencia y en el búfer de entrada del objeto SerialPort.
ReadLine()	Lee hasta el valor de NewLine en el búfer de entrada.
ReadTo(String)	Lee una cadena hasta el valor especificado en el búfer de entrada.
ToString()	Devuelve un objeto String que contiene el nombre del objeto Component, en caso de que exista. Este método no debe reemplazarse (heredado de Component).

Nombre	Descripción
Write(Byte(), Int32, Int32)	Escribe un número especificado de bytes en el puerto serie utilizando los datos de un búfer.
Write(Char(), Int32, Int32)	Escribe un número especificado de caracteres en el puerto serie utilizando los datos de un búfer.
Write(String)	Escribe la cadena especificada en el puerto serie.
WriteLine(String)	Escribe la cadena especificada y el valor de NewLine en el búfer de salida.

1.6.5 Comandos de lectura y escritura

Para fines prácticos, las instrucciones a utilizar para establecer comunicación serán las siguientes y se aplicarán en los proyectos presentados en los siguientes capítulos.

Comando	Función
Serialport.open	Abre el puerto COM.
Serialport.close	Cierra el puerto COM.
Serialport.ReadExisting	Lee los caracteres enviados.
Serialport.write	Escribe un valor en el puerto.

1.7 Resumen

En este capítulo se trataron los aspectos importantes de comunicación. Primeramente, los principios de programación en Arduino y las estructuras que se van a requerir para darle una lógica de lecturas y escrituras. Otro aspecto importante son los fundamentos de la comunicación serial en Arduino así como los principios y las funciones para establecer comunicación desde la interfaz de Visual Basic. NET. En el siguiente capítulo se establecerá cómo crear una interfaz HMI para control y monitoreo.

1.8 Problemas

1. Realice la siguiente operación y mostrar el resultado a = (a + b) * (a * c) / 1024.
2. Lleve a cabo un programa que resuelva las potencias de 2^{10}.
3. Efectuar un programa que lea un valor de temperatura en grados Celsius y que después lo convierta a las escalas Kelvin y Fahrenheit.
4. Capture un número y desplegar su conversión en formato binario, octal y hexadecimal.
5. Realizar un programa que registre dos números y muestre en pantalla cuál es el mayor.
6. Haga un programa que capture tres números y determine cuál es el mayor, el menor y si son iguales.
7. Escriba un programa que lea cinco números de tipo entero y determine cuál es el mayor de estos.
8. Realice un programa que lea un valor simulando la temperatura y lleve a cabo las siguientes comparaciones: si es menor o igual a 18, está frío; si es mayor a 25, temperatura media; si es mayor que 35 grados, está caliente; si mide entre 36 y 42, está muy caliente.
9. Escriba un programa que registre el nombre, la edad y una contraseña; después, si el nombre y la contraseña son correctos, deberá permitir el acceso. Haga una variante en la que con cualquiera de los tres ítems se pueda acceder.
10. Elabore un programa que lea un caracter y determine si es una letra minúscula o mayúscula y que también despliegue si se trata de un número o un símbolo especial.
11. Lleve a cabo un programa que registre una letra o un número y despliegue el código ASCII del valor capturado.
12. Realice un programa que codifique los valores para mostrar el mensaje "Hola" desde Arduino.
13. Obtenga un programa que permita solamente capturar los números del 0 al 9 y caracteres alfanuméricos.
14. Realizar un programa que muestre los números del 0 al 100.
15. Escriba un programa que muestre los números pares del 0 al 255.

16. Realice un programa que sume los números del 0 al 10.

17. Escribir un programa que obtenga la sumatoria de los números con base en un número específico de sumandos.

18. Elabore un programa que capture 10 números diferentes y que calcule el promedio, la sumatoria, el número menor y el número mayor.

19. Realizar un programa que permita capturar los números que el usuario desee, el programa debe estar controlado por una S (sí) y una N (no) cuando ya no requiera de realizar más capturas, calcular el promedio y la cantidad de valores capturados.

20. Leer en valor de tipo entero.

21. Mandar llamar tres programas diferentes y ejecutar cada uno de ellos (utilizar la instrucción de control Switch Case).

22. Elaborar un contador con incrementos de uno en uno que cuando llegue a grupos de 12 despliegue un mensaje; al finalizar un lote de 1000 debe mostrar el mensaje: "Se llegó al conteo esperado".

Capítulo 2

Aspectos generales de la programación en Visual Basic .NET

Objetivos

En este capítulo se explicará la programación en Visual Basic .NET requerida para realizar las interfaces gráficas hombre-máquina. También, se podrá realizar un primer programa con el cual la tarjeta Arduino se comunicará con una pantalla hecha en .NET.

2.1 Introducción

En este capítulo se explicará cómo desarrollar interfaces HMI en el entorno de programación de Visual Basic .NET para monitoreo, control y adquisición de datos desde la tarjeta Arduino.

Asimismo, conforme se vaya desarrollando el proyecto aquí presentado, se aprenderá a hacer un proyecto en Windows Forms; a utilizar los controles y aplicarlos para control, lectura y adquisición de datos; a utilizar la comunicación entre SerialPort y Arduino; a establecer comunicación entre Arduino y .NET y, finalmente, a integrar todos los elementos para conformar el sistema.

2.2 Requerimientos de software y hardware

Para este proyecto se necesita una tarjeta Arduino, así como varios compontes, que se irán explicando en cada uno de los proyectos y sus aplicaciones.

En cuanto a software, se requiere el entorno de programación de Arduino. También es necesario instalar Microsoft Visual Studio 2012 en la computadora.

2.3 Configuración de hardware

Primeramente, hay que realizar el código en la placa Arduino para recibir el dato enviado desde Visual Basic .NET:

Las instrucciones para el manejo del puerto serial son las siguientes:

- Serial.Begin: inicializa la comunicación a una velocidad configurada.
- Serial.Available: verifica si el puerto serial está disponible.
- Serial.Read: lee el dato enviado.

El formato de instrucción se detalla en el ejemplo 2.1.

Ejemplo 2.1 Formato de instrucción para el manejo del puerto serial.

```
int ledpin = 13;
int dato;
void setup()
{
  pinMode(ledpin,OUTPUT);
  Serial.begin(9600);                  // configurar comu-
nicación serial a 9600 bps.
}
void loop(){
      if(Serial.available()){
```

```
        dato = Serial.read();

        if(dato == '1'){
        digitalWrite(ledpin,HIGH);
      }
      if dato == '0'){
        digitalWrite(ledpin,LOW);
       }
}
```

Una vez realizado el código, el siguiente paso es verificar el puerto COM (puerto serial) desde el menú Herramientas, como se muestra en la figura 2.1; después hay que descargar el *sketch* realizado.

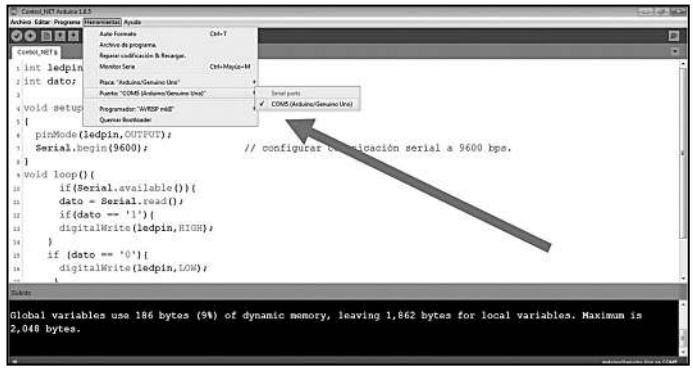


Figura 2.1 Verificar el puerto COM.

Como se puede observar en la figura 2.1, el puerto de comunicación COM5 es el que configuró Windows desde que se instaló la tarjeta Arduino con sus controladores.

Una vez verificado el puerto COM, se debe cargar el sketch haciendo clic en el icono de la flecha (⇨), como se muestra en la figura 2.2.

Figura 2.2 Cómo cargar el sketch.

Para realizar una prueba de comunicación y verificar que nuestro programa cargado sea el correcto, se utiliza la consola de comunicación del IDE de Arduino haciendo clic en la esquina superior derecha de la pantalla. (Ver figura 2.3.)

Figura 2.3 Verificación del programa cargado.

Al hacer clic en la esquina superior derecha, marcada con el recuadro en la figura 2.3, se abre la pantalla de la interfaz serial; entonces, escribimos un carácter de número uno (1) y el led conectado al pin número 13 debe encenderse; si capturamos un carácter cero (0), debe de apagarse. (Ver figuras 2.4, 2.5 y 2.6.)

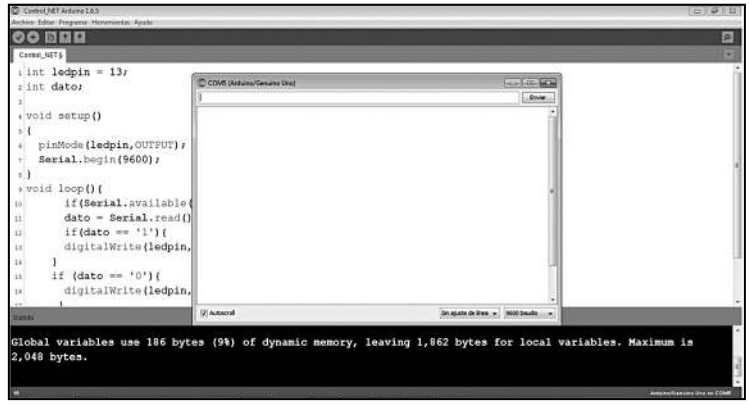


Figura 2.4 Se abre la pantalla de interface serial para incluir caracteres.

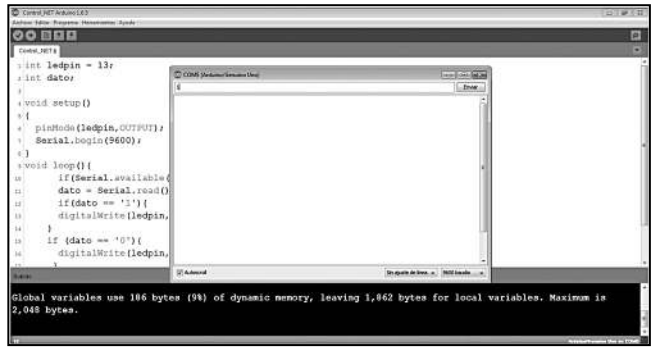


Figura 2.5 Si se escribe carácter uno (1), el led debe encenderse.

Figura 2.6 Si se escribe carácter cero (0), el led debe apagarse.

Es importante que la conexión de la tarjeta Arduino con Visual Basic .NET sea la correcta. Para ello se muestra el diagrama de conexión en la figura 2.7.

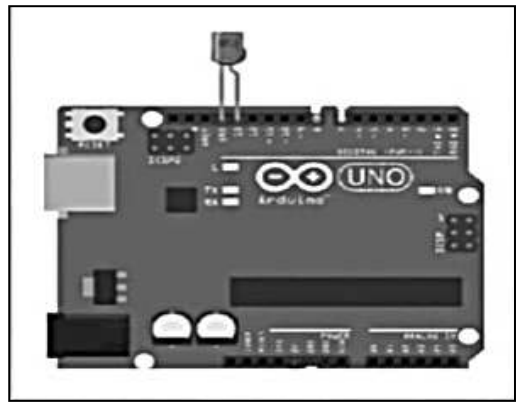


Figura 2.7 Diagrama de conexión en la tarjeta para el control desde Visual Basic .NET.

2.4 Escritura de un programa en Visual Basic .NET

En esta sección se explicará el procedimiento para crear una aplicación en .NET con el fin de establecer la comunicación serial con la tarjeta Arduino UNO.

El primer paso es crear una aplicación en Visual Basic .NET 2012. Es decir, hay que abrir dicha aplicación para crear una aplicación de Windows Forms, tal como se muestra en la figura 2.8.

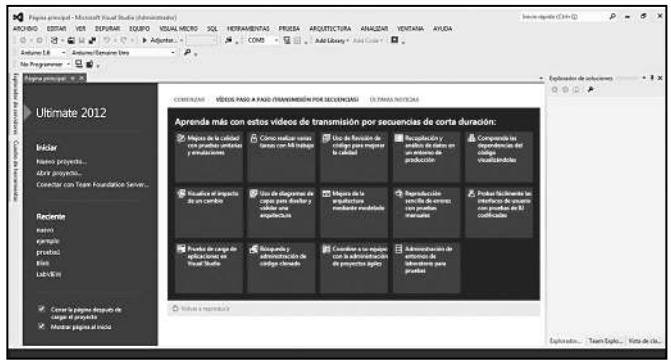


Figura 2.8 Abrir la aplicación.

Una vez que se abre Visual Basic .NET 2012, hay que hacer clic en el menú Archivo; posteriormente, hay que seleccionar el comando Nuevo y, enseguida, el comando Proyecto, tal como se muestra en la figura 2.9.

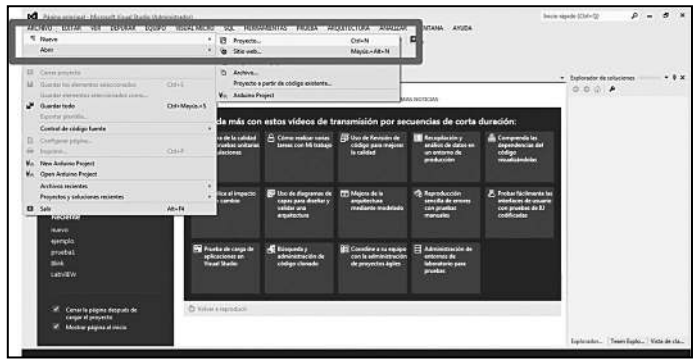


Figura 2.9 Dar clic en menú Archivo, elegir Nuevo y enseguida, Proyecto.

Posteriormente, seleccionar Aplicación de Visual Basic de tipo Windows y después dar clic en Aplicación de Windows Forms; enseguida, en la barra inferior se escribe el nombre de la aplicación, en este caso ComSerial. (Ver el recuadro de la figura 2.10.)

Figura 2.10 Crear una aplicación en Windows Forms.

Con el fin de familiarizarse con Visual Basic .NET, en la figura 2.11 se explican los elementos básicos en pantalla: pantalla de interfaz, explorador de proyecto, cuadro de herramientas y pantalla de propiedades.

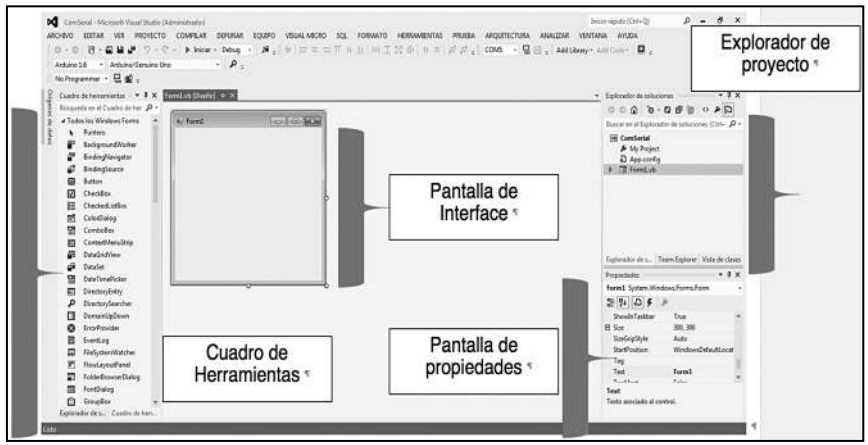


Figura 2.11 Elementos básicos de la pantalla de Visual Basic .NET.

2.5 Manejo de controles en pantalla

A continuación se explicarán los controles que se utilizan para el control, el monitoreo y la adquisición de datos, ubicados en los menús Controles comunes, Componentes y Visual Basic PowerPacks, que se pueden apreciar en la figura 2.12.

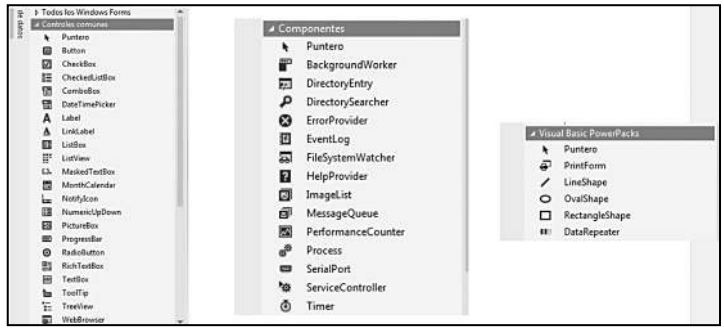


Figura 2.12 Controles que se utilizan para el control, el monitoreo y la adquisición de datos.

En la siguiente tabla se enlistan dichos controles, se describe su función y en qué contexto se aplican.

Nombre del control	Función	Aplicación
Label	Poner letreros en la pantalla.	Títulos en la pantalla; por ejemplo, temperatura, humedad, etcétera.
TextBox	Permitir leer un valor en formato numérico o en una palabra.	Por ejemplo, permitir leer el valor de la temperatura.
Button	Permitir hacer clic y ejecutar una función.	Al presionar el botón se envía un carácter para encender un led conectado al Arduino.
VScrollBar o HScrollBar1	Moverse en un rango de valores.	Al controlar la velocidad de motor.
SerialPort	Establecer la comunicación a través de sus funciones de lectura y escritura.	Comunicación con la tarjeta Arduino.
Timer	Establecer un tiempo de sincronización para leer los datos.	Sincronizar la lectura de los datos enviados.
Visual Basic Power Packs (LineShape, OvalShape, RectangleShape)	Hacer figuras de círculos, líneas y rectángulos de color.	Para monitorear lo que se envía o se recibe, así como realizar interfaces interactivas.

La figura 2.13 ubica el control Button el cual permite que al hacer clic se ejecute una acción. Este control será el que se utilizará en este proyecto para encender y apagar la tarjeta.

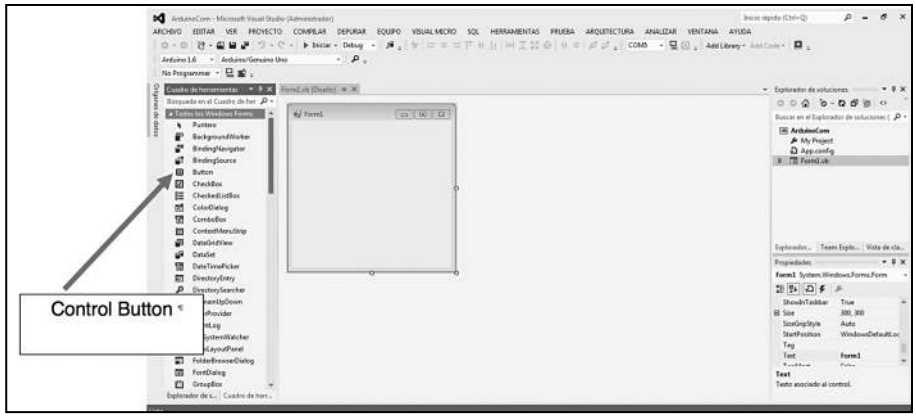


Figura 2.13 El control Button.

La figura 2.14 ubica el control SerialPort, que permite establecer una comunicación serial con cualquier dispositivo, en este caso, la computadora. Este control es el que utilizaremos para establecer comunicación entre la tarjeta y los controles Button.

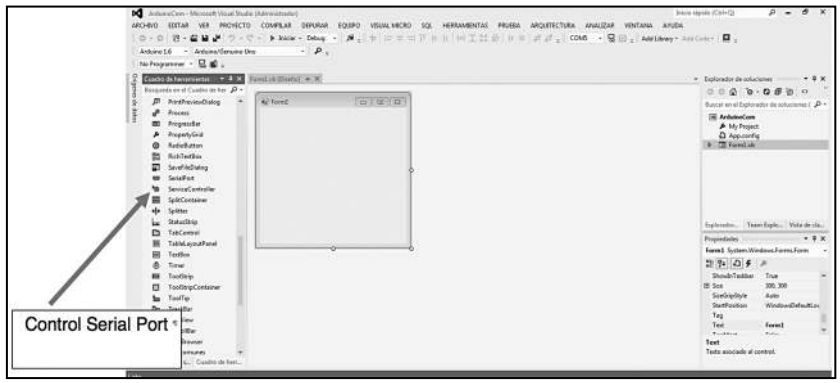


Figura 2.14 El control SerialPort.

2.6 Prueba de comunicación con Arduino

En esta sección se aprenderá a activar un bit de la tarjeta Arduino con sólo hacer clic en un botón para encenderlo y clic en otro botón para apagarlo.

Primeramente, se debe arrastrar el control SerialPort al formulario, como se muestra en la figura 2.15.

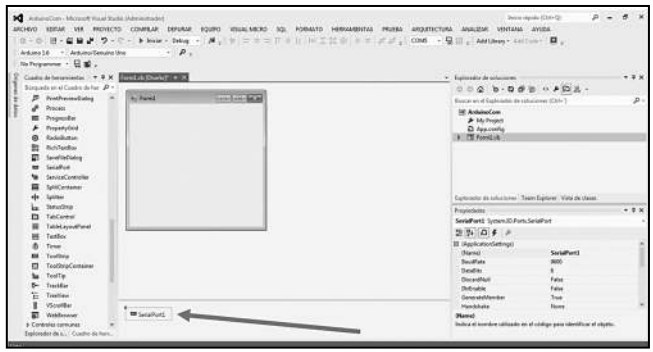


Figura 2.15 Arrastrar el control SerialPort al formulario.

Una vez que el control SerialPort se encuentra en el formulario, es posible ver sus propiedades. Esto es crucial, pues el control se debe configurar con los siguientes datos: BaudRate de 9600, DataBits de 8, Parity None y PortName COM1; de acuerdo con la figura 2.16.

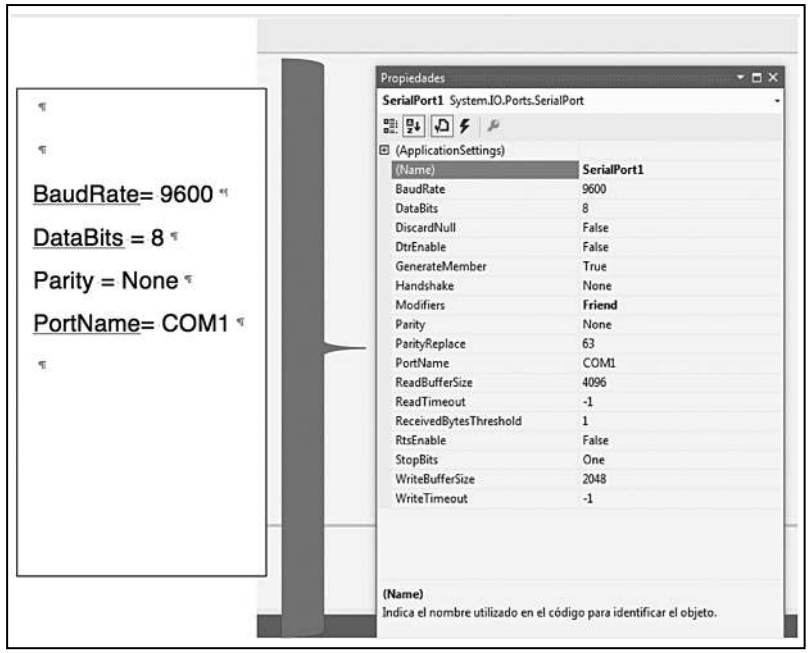


Figura 2.16 Configuración del control SerialPort.

El siguiente paso es insertar dos controles Button a la pantalla, como se muestra en la figura 2.17.

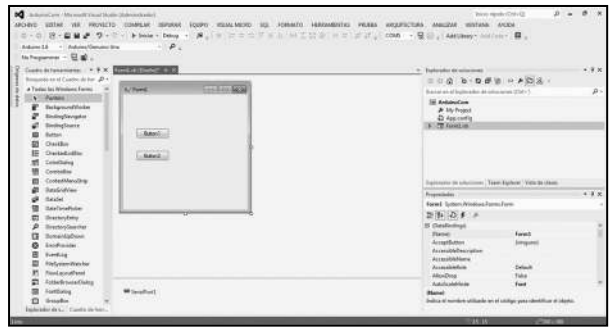


Figura 2.17 Insertar dos controles Button.

Posteriormente, hay que cambiar la propiedad a cada control Button (botón) a texto. Para ello, hay que seleccionar, en el menú de Propiedades, el comando Text. Cada control debe nombrarse "1" y "0", respectivamente. (Ver figuras 2.18 y 2.19.)

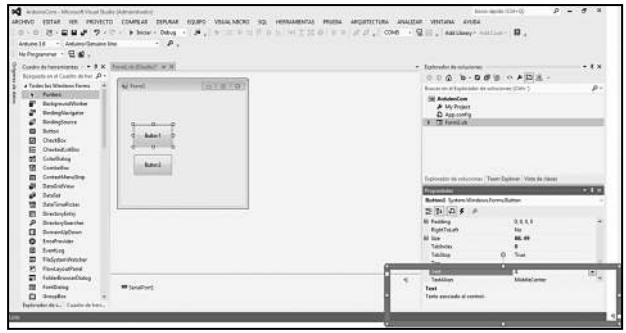


Figura 2.18 Cambiando la propiedad de los controles Button a Text.

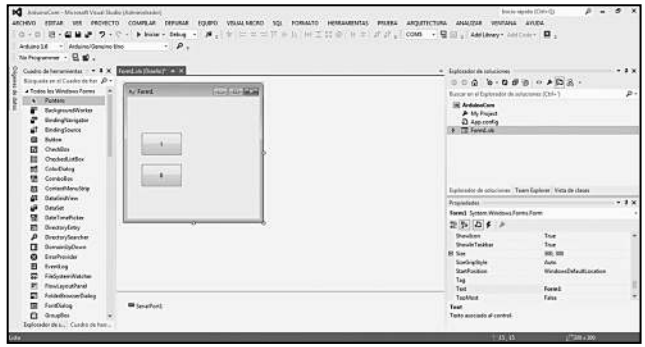


Figura 2.19 Nombrando cada Button como "1" y "0", respectivamente.

2.6.1 Comandos del control SerialPort

Posteriormente, hay que establecer comunicación entre los botones y la tarjeta con el control SerialPort. A continuación se enlistan los comandos en éste y se describen brevemente sus funciones.

```
Serialport.Open: abre la comunicación.
Serialport.Close: cierra la conexión.
Serialport.Write: escribe en el puerto.
Serialport.ReadExisting: lee un valor del puerto.
```

El siguiente paso es abrir la comunicación serial desde la aplicación. Para realizar esto, se debe hacer doble clic sobre la forma, en la pantalla de interfaz (recordar figura 2.11), de tal manera que se abra la pantalla de inicio de la forma, como lo muestra la figura 2.20.

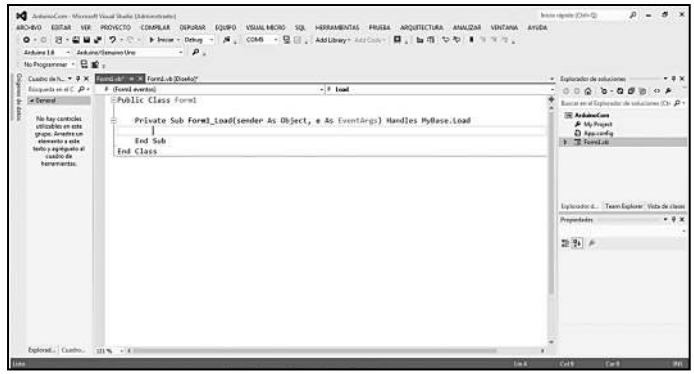


Figura 2.20 Pantalla de inicio de la forma.

En esa misma pantalla de inicio, en el código de la forma, teclear el comando del control SerialPort.Open; es decir, el comando que abre la comunicación. (Ver figura 2.21.)

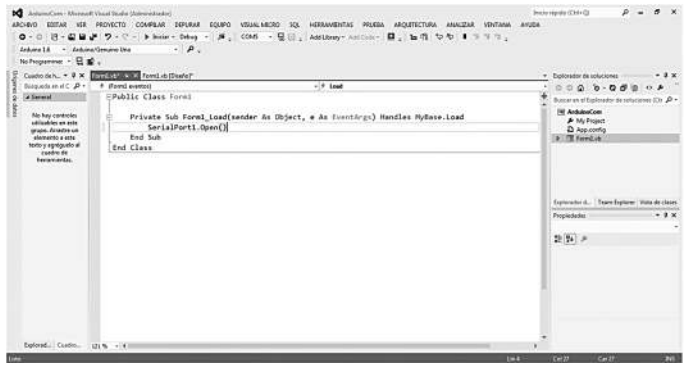


Figura 2.21 Abrir la comunicación desde la aplicación por medio del comando Serial-Port.Open.

Una vez abierta la comunicación, hay que regresar a la forma de diseño para continuar programando los botones. Para ello, dar doble clic sobre el botón que tiene el texto 1; entonces aparece el código del botón 1 (Button 1). (Ver la figura 2.22.)

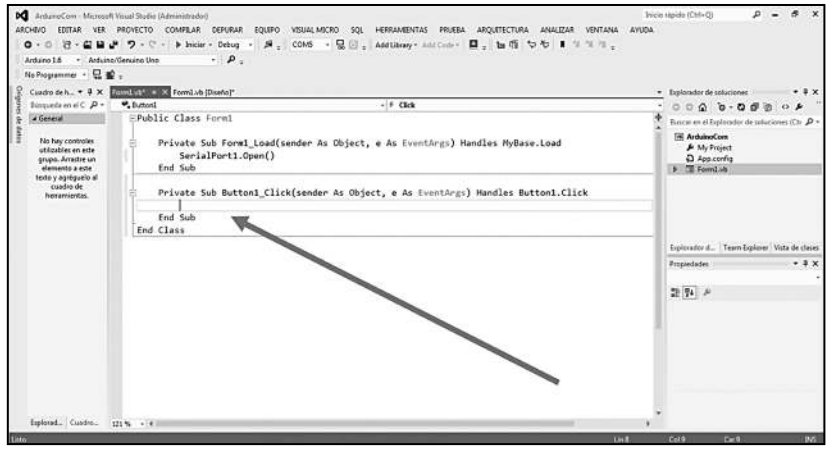


Figura 2.22 Iniciar la programación de los botones.

Puesto que se va a escribir en el puerto, hay que utilizar el comando de escritura, SerialPort.Write("1"). Se agrega el número uno entre paréntesis y comillas ("1") porque es el carácter que se enviará en este proyecto. (Ver figura 2.23.)

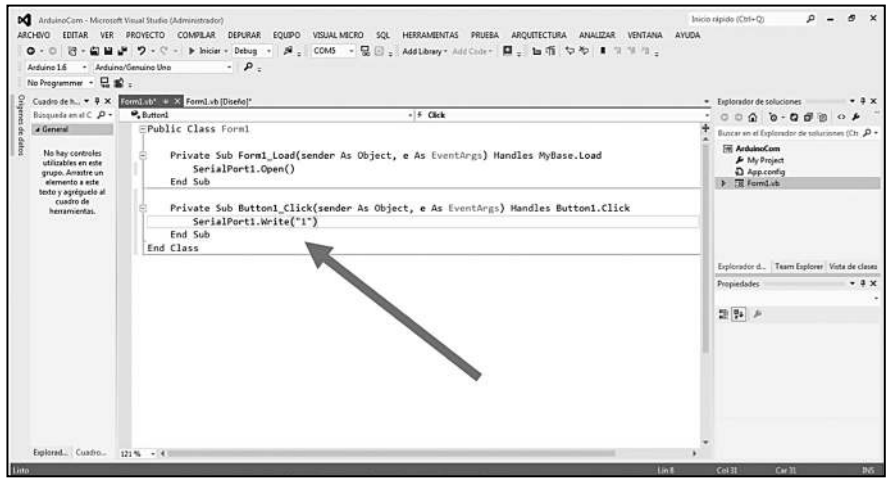


Figura 2.23 Escribir en el puerto con el comando SerialPort.Write().

Después se programa el Button 2, cuyo carácter será cero (0). No olvidar las comillas. (Ver figura 2.24.)

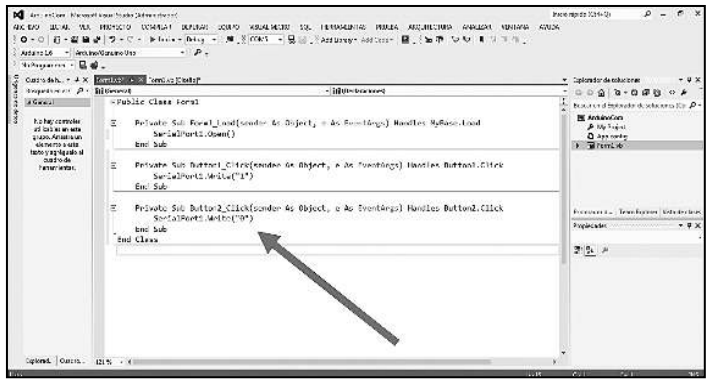


Figura 2.24 Programación del Button 2 con el carácter cero.

Por último, se verifica que el puerto de comunicación sea el mismo que se configuró en el entorno de IDE de Arduino; de otro modo, no se establecerá ninguna comunicación. Esto se revisa en la pantalla de propiedades, en el menú PortName. (Ver el recuadro en la figura 2.25.)

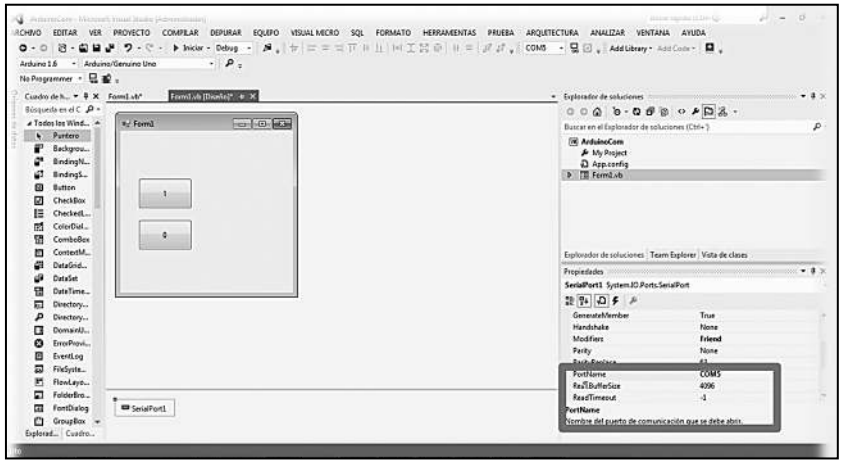


Figura 2.25 Verificación del puerto de comunicación.

Ahora es recomendable hacer una prueba de funcionamiento de los botones. Se puede apreciar el correcto funcionamiento en las figuras 2.26, 2.27 y 2.28.

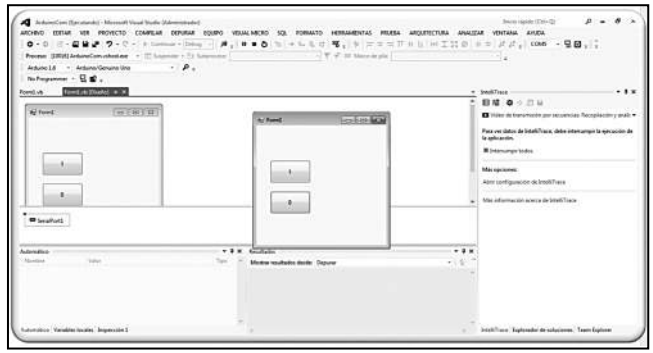


Figura 2.26 Iniciar las pruebas de funcionamiento de los botones.

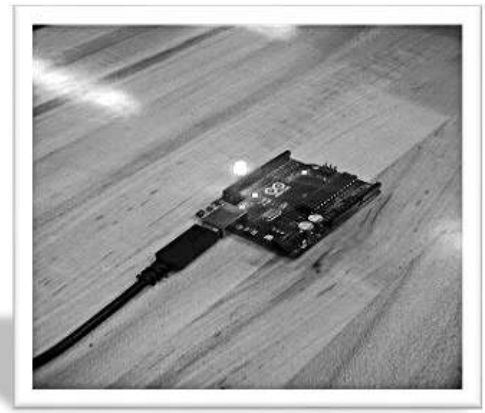


Figura 2.27 El botón 1 enciende la tarjeta.

Figura 2.28 El botón 2 apaga la tarjeta.

2.7 Resumen

En este capítulo se aprendió cómo realizar una interfaz HMI en Visual Basic .NET con los controles básicos para comunicación serial. En el siguiente capítulo se aplicará lo aprendido hasta este momento en la creación de proyectos para control y monitoreo de señales de entrada y de salida desde la tarjeta Arduino. También se aprenderá a conectar sensores y un relevador para encenderlo y apagarlo desde la interfaz HMI creada en Visual Basic .NET.

2.8 Problemas

1. Diseñe una interfaz en Windows Forms utilizando Radiobuttons que permita encender cinco leds, cada radiobutton es un led conectado a la tarjeta Arduino.
2. Realice un programa que contenga cuatro botones de modo que al enviarle los caracteres 'a' o 'A', se encienda la salida del led conectado al Arduino, hacer lo mismo para el apagado de la salida.
3. Ejecute un programa que tenga dos botones, con uno de ellos realizará una secuencia de encendido y con el otro, la secuencia inversa.
4. Escribir un programa que encienda y apague una secuencia de leds cierta cantidad de veces, el dato de control debe ser enviado desde la interfaz de Visual Basic .NET.
5. Realice un programa en una interfaz de Windows Forms en el que se encienda y se apague una salida conectada al Arduino.
6. Por medio de un programa, controle un semáforo simple.
7. Repita el ejercicio 6 para el caso de un semáforo crucero.
8. Realizar una interfaz de monitoreo donde se capturen dos donde, si el primero es mayor al segundo, se envíe una activación de salida al Arduino de forma automática.
9. Capturar el usuario y la contraseña, si ambos son correctos: activar una secuencia de leds; de lo contrario, activar la secuencia en sentido inverso.
10. Realice un contador del 0 al 255 que envíe la secuencia de encendido y apagado al Arduino en formato binario.

11. Elabore un programa que permita enviar un número del 0 al 9 y, en Arduino, desplegar el número capturado a través del encendido de leds.

12. Diseñe un programa que despliegue una matriz de led controlada desde el software de Visual Basic .NET.

Capítulo 3

Estación meteorológica de monitoreo con Arduino y Visual Basic .NET

Objetivos

En este capítulo se conectará un sensor de temperatura y humedad DTH11 con una fotoresistencia, con el fin de realizar una estación meteorológica y enviar los datos capturados a la pantalla de una computadora conectada por el puerto serial de la PC. Además, se conectará una pantalla LCD 20x4 y el módulo I2C para conectar la pantalla con la placa de Arduino y así poder visualizar los datos.

3.1 Introducción

En múltiples ocasiones, es conveniente y necesario conectar el microcontrolador con una computadora personal (PC) para realizar el control y visualización de datos. Realizar este tipo de comunicación es posible gracias a las herramientas de Arduino y a la programación en Visual Basic .NET, una de las herramientas de mayor uso en el desarrollo de software gráfico.

En los capítulos anteriores se estudiaron los aspectos relacionados con la programación en Arduino, la comunicación serial entre dispositivos y el manejo de interfaces gráficas realizadas para el control y monitoreo. También se explicó cómo realizar un programa para enviar datos al monitor serial y establecer comunicación con la computadora.

Ahora, en este capítulo, se explicará cómo monitorear la temperatura y la humedad, y detectar luz desde una pantalla LCD. Asimismo, se describirá cómo crear una pantalla de monitoreo donde se muestren esos valores.

3.2 Requerimientos de software y hardware

Para este proyecto se necesita una tarjeta Arduino UNO, aunque una tarjeta Arduino MEGA también funciona.

Para leer la temperatura se requiere de un sensor DHT11, una resistencia de 4.7k, una fotoresistencia (sensor de luz) y una resistencia de 10k.

También se requiere de una pantalla LCD 20 x 4 y un módulo de comunicación I2C (ver figura 3.1.) para desarrollar la interfaz de la pantalla con la tarjeta Arduino. Se recomienda utilizar esta comunicación, ya que solamente se requieren dos pines del Arduino para realizar el envío de los datos.

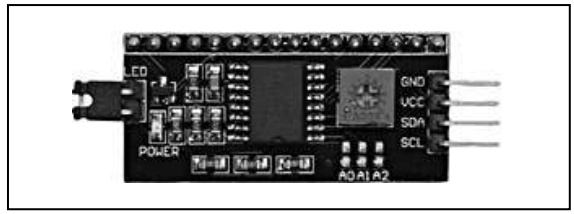


Figura 3.1 Módulo de comunicación 12C.

Para realizar las conexiones se requiere de un protoboard y cables macho-macho y hembra-macho.

A continuación se muestra la lista de todos los componentes para el proyecto:

- Arduino UNO
- Sensor DHT11
- Pantalla LCD 20 x 4

- Módulo I2C para pantalla LCD
- Protoboard
- Resistencia de 4.7k
- Resistencia de 10k
- Cables macho-macho y macho-hembra

En cuanto al software, se requiere de la librería LiquidCrystal_I2C.h para la pantalla LCD I2C, así como la librería Wire.h y la librería DHT11.h. Para instalar las librerías es necesario copiar la carpeta de la librería descargada en la carpeta Arduino/libraries.

3.3 Cómo conectar los diferentes componentes

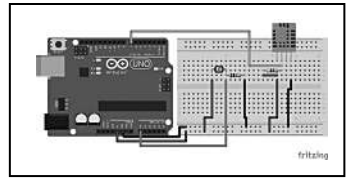


Figura 3.2 Diagrama de la conexión de los componentes.

En la figura 3.2 se puede apreciar la conexión de todos los componentes. A continuación se explicará cómo conectar cada uno.

3.3.1 Conexión de la tarjeta Arduino al protoboard

Primero se debe conectar el pin de +5V de la Arduino a la línea color rojo del protoboard; luego, el pin de GND a la línea color negro.* (Ver figura 3.2.)

3.3.2. Conexión del sensor DHT11 al protoboard

Para conocer los pines del sensor DHT11 se muestra la figura 3.3.

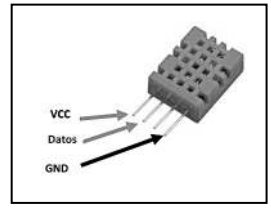


Figura 3.3 Pines del sensor DHT11.

Después hay que conectar el pin número 1 del sensor DHT11 (VCC) a la línea roja* del protoboard y el pin número 4 (GND) a la línea azul.* También, se debe conectar

***Nota del editor:** véase la figura a color en la página web del libro.

el pin número 2 del sensor al pin número 7 de la placa Arduino. Finalmente, hay que conectar la resistencia de 4.7k ohms entre los pines 1 y 2 del sensor DHT11. (Ver figura 3.2.)

3.3.3 Conexión de la fotorresistencia al protoboard

Por otro lado, hay que colocar la fotorresistencia en serie con la resistencia de 10k ohm en el protoboard (ver figura 3.2). Luego, se debe conectar el extremo restante de la fotorresistencia con la línea roja** del protoboard y el extremo restante de la resistencia a la línea azul** (tierra). Por último, conectar el pin común entre la fotorresistencia y la resistencia de 10k al pin A0 analógico del Arduino. (Ver línea naranja de la figura 3.2.)

3.3.4 Conexión de la pantalla LCD

Ahora, se conectará la pantalla LCD. Puesto que es una pantalla LCD con una interfaz I2C, sólo se necesitará conectar dos cables para la señal y dos para la energía. Conectar el pin del módulo I2C, llamado VCC, a la línea roja** en el protoboard y el pin GND a la línea azul** en el protoboard. A continuación, como se puede apreciar en la figura 3.4, conectar el pin SDA del módulo al pin A4 del Arduino y el pin SCL al pin A5 del Arduino.

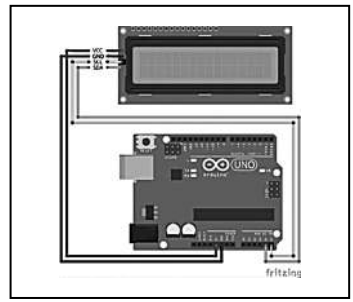


Figura 3.4 Conexión de la pantalla LCD a la tarjeta Arduino.

La figura 3.5 muestra el proyecto totalmente montado.

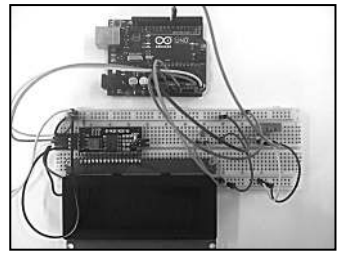


Figura 3.5 Así se debe ver la conexión de todos los componentes.

****Nota del editor:** ver la figura a color en la página web del libro.

3.4 Prueba de los sensores

Una vez que el hardware del proyecto está totalmente montado, hay que probar los diferentes sensores. Para ello, se escribe un sencillo sketch en Arduino. Simplemente se leerán los datos de los sensores y se imprimirán estos datos sobre el puerto serial. El código es el siguiente:

```
// Librerias
#include "DHT.h"


// Sensor de temperatura y humedad
#define DHTPIN 7
#define DHTTYPE DHT11

// instancia del sensor
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  // inicializa el Puerto serial
  Serial.begin(9600);
    // Inicializa el sensor
  dht.begin();
}

void loop()
{
  float tem = dht.readTemperature();
  float hum = dht.readHumidity();

  // Mide el sensor de luz
  float sensor = analogRead(0);
  float luz = sensor / 1024 * 100;



  // Despliega la temperatura
  Serial.print("Temperatura: ");
  Serial.print(temp);
  Serial.println(" C");

   // Despliega la humedad
  Serial.print("Humedad: ");
  Serial.print(hum);
  Serial.println("%");

  // Despliega el nivel de luz
  Serial.print("Luz: ");
  Serial.println(luz);
  Serial.println("%");
```



```
Serial.println("");

// se espera 700 ms
delay(700);
}
```

Para entender todas las partes del código, ver la figura 3.6.

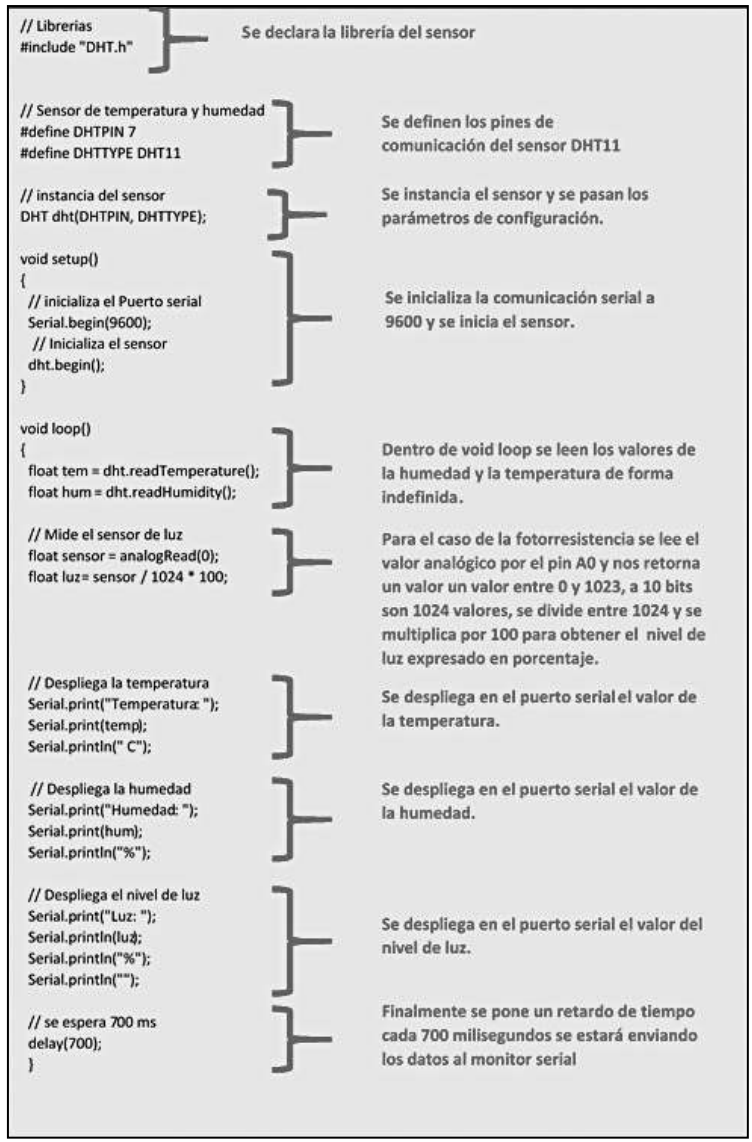


Figura 3.6 Explicación del código de prueba de los sensores.

El código debe descargarse en la placa de Arduino. También debe abrirse el monitor serial para visualizar los datos enviados. Es importante verificar que la velocidad de transmisión del puerto serial esté a 9600. En la figura 3.7 se muestra cómo debe configurarse el código.

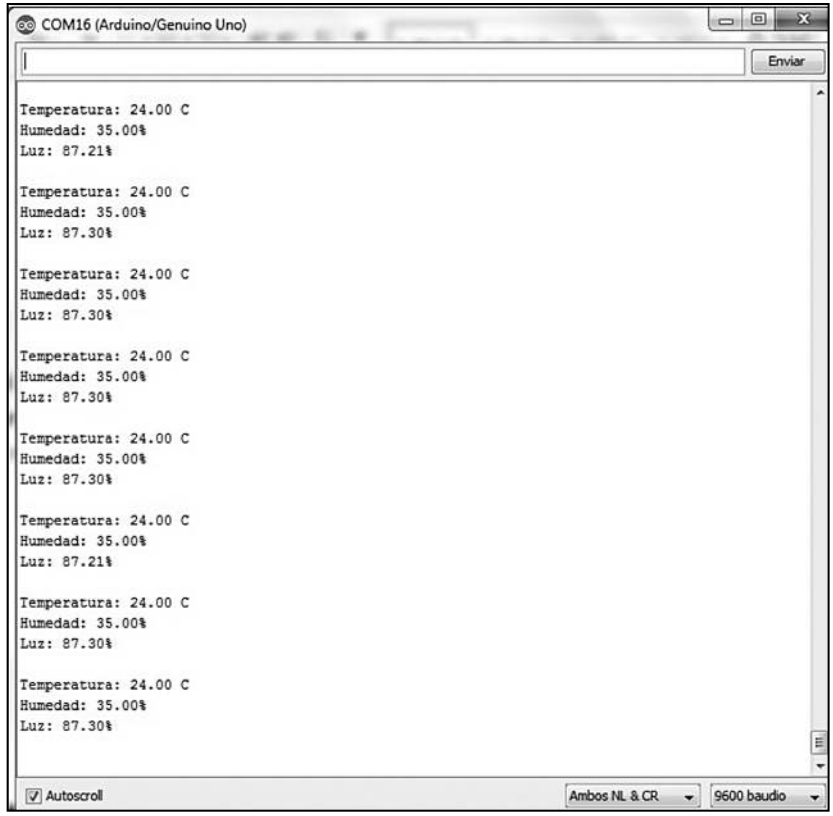


Figura 3.7 El código en la placa Arduino.

3.5 Desplegado de datos en la pantalla LCD

El siguiente paso es integrar todos los componentes para visualizar la información en la pantalla LCD. El procedimiento para la lectura de los sensores será el mismo, solamente se detallará lo que se refiere a la comunicación y a mostrar los datos en la LCD. En la figura 3.8 se explica el código completo para ello.

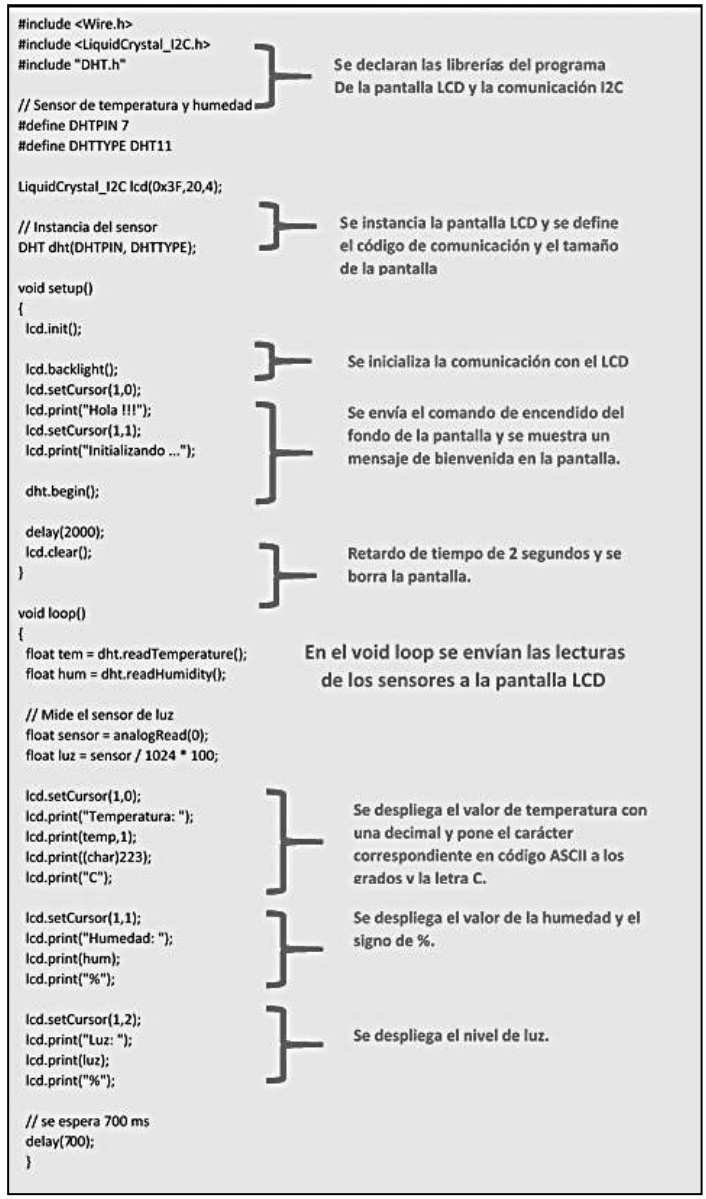


Figura 3.8 Código para establecer comunicación con la LCD.

El siguiente paso es descargar el ejemplo en la placa Arduino. Hay que esperar un poco para visualizar las lecturas en el LCD. En la figura 3.9 se aprecia el proyecto en acción.

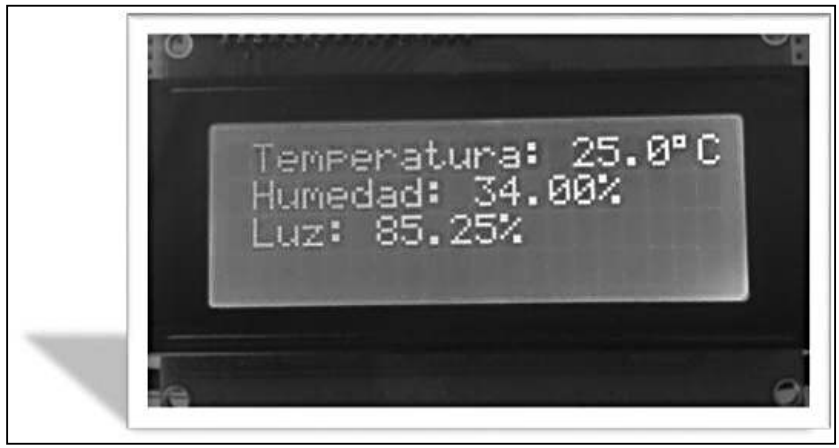


Figura 3.9 Lecturas en la pantalla LCD.

3.6 Pantalla de monitoreo del sistema

En esta parte se explica el monitoreo de los valores enviados desde la tarjeta Arduino para mostrar los datos en una pantalla de interfaz gráfica.

Los pasos para comunicar Arduino con Visual Basic .NET son los siguientes:

1) Enviar los valores a leer (mandar los datos por el puerto serial en un mismo renglón).
2) Abrir un puerto de la PC.
3) Leer el dato enviado desde Arduino.
4) Dividir los datos enviados y guardarlos en una variable.
5) Desplegar los datos en los controles para mostrar la información de forma gráfica.

A continuación se muestran las instrucciones para realizar la transferencia de los datos por el puerto serial; este código se explicó en el punto de prueba de sensores ya antes visto (sección 3.4), aunque en este caso, el envío de los datos se hace en un solo renglón, cada dato separado por comas, para identificar cada uno de los valores. (Ver la figura 3.10.)

Figura 3.10 Instrucciones para transferencia de los datos.

La figura 3.11 muestra cómo se ven los datos enviados en el monitor serial.

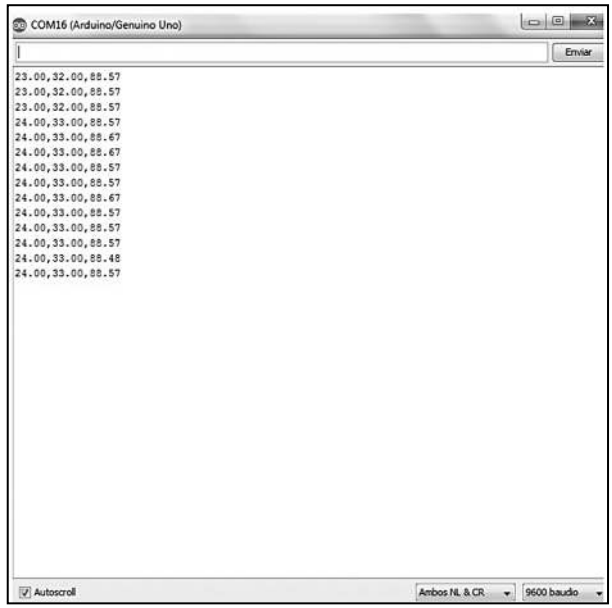


Figura 3.11 Monitor serial mostrando los datos enviados desde Arduino.

3.6.1 Pasos para crear la interfaz de monitoreo

El primer paso para crear la interfaz de monitoreo es crear una aplicación en Visual Basic .NET 2012. Para ello, hay que abrir el programa de Visual Basic .NET, tal como se muestra en la figura 3.12.

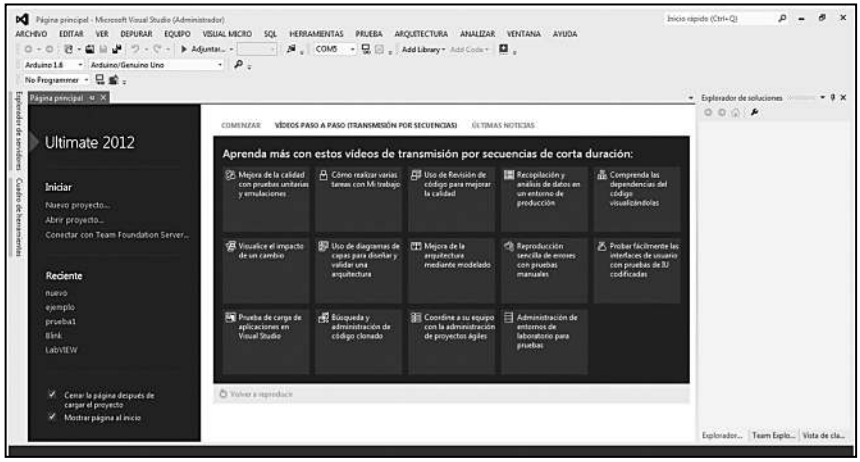


Figura 3.12 Inicialización del programa Visual Basic .NET para crear la aplicación.

Una vez abierto el programa, hacer clic en el menú Archivo, seleccionar Nuevo y luego el proyecto a crear, tal como lo muestra el recuadro de la figura 3.13.

Figura 3.13 Selección de un nuevo proyecto para crear la interfaz de monitoreo.

Hay que seleccionar la aplicación de Visual Basic de tipo Windows y después la aplicación de Windows Forms. Se debe escribir el nombre de la aplicación; en este caso, ComSerial y dar clic en Aceptar. (Ver figura 3.14.)

Figura 3.14 Selección de la aplicación de Windows Forms en Visual Basic .NET.

En la figura 3.15 se explican los componentes de la pantalla de Visual Basic .NET en el entorno de trabajo.

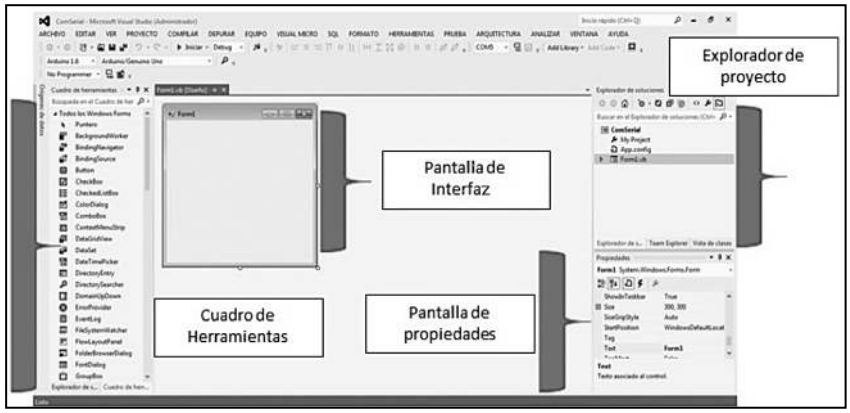


Figura 3.15 Componentes del entorno de trabajo de Visual Basic .NET.

Será útil reconocer, en los controles comunes de Visual Basic, cuáles son los que se utilizarán en este proyecto. En la figura 3.16 se señalan tales controles; a saber: Label, Textbox, Serialport y Timer, con los cuales se desarrollará la aplicación:

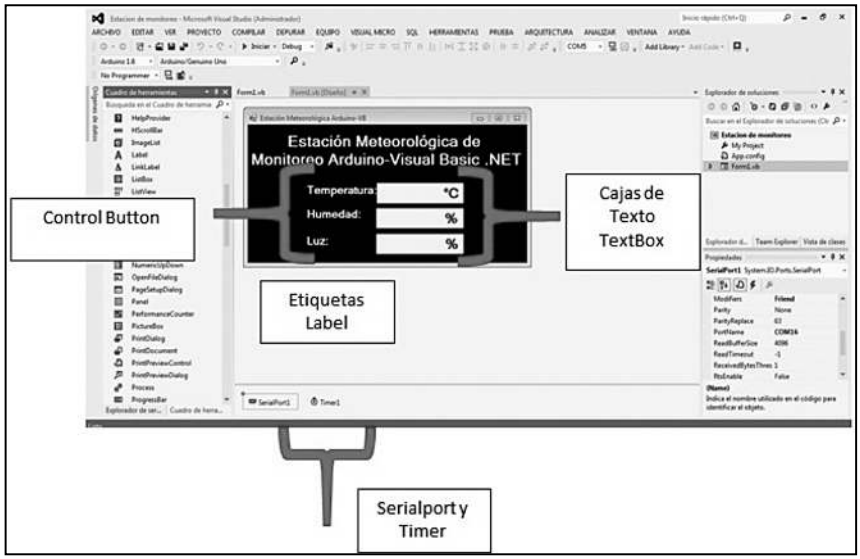


Figura 3.16 Controles comunes de Visual Basic .NET que se utilizarán en el proyecto.

El control SerialPort del entorno de Visual Basic .NET permite establecer una comunicación serial con cualquier dispositivo, en este caso, la computadora. La figura 3.17 ubica el control SerialPort.

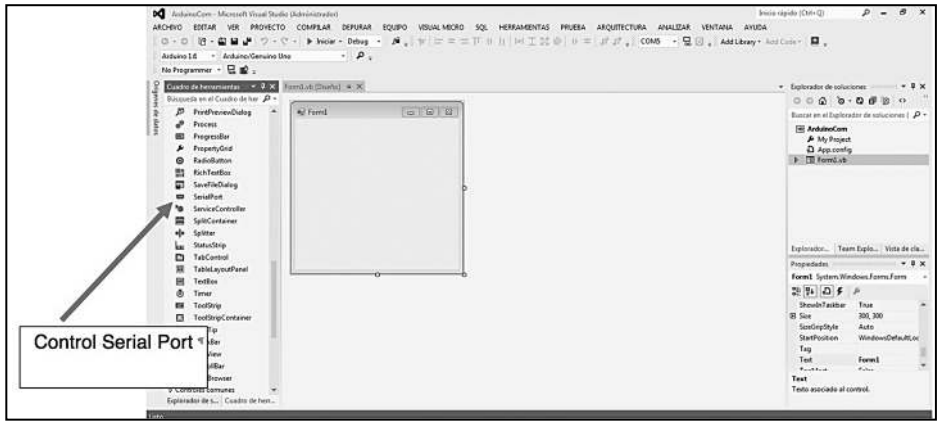


Figura 3.17 El control SerialPort en el menú Herramientas.

Es necesario arrastrar el control SerialPort al formulario, como se muestra en la figura 3.18.

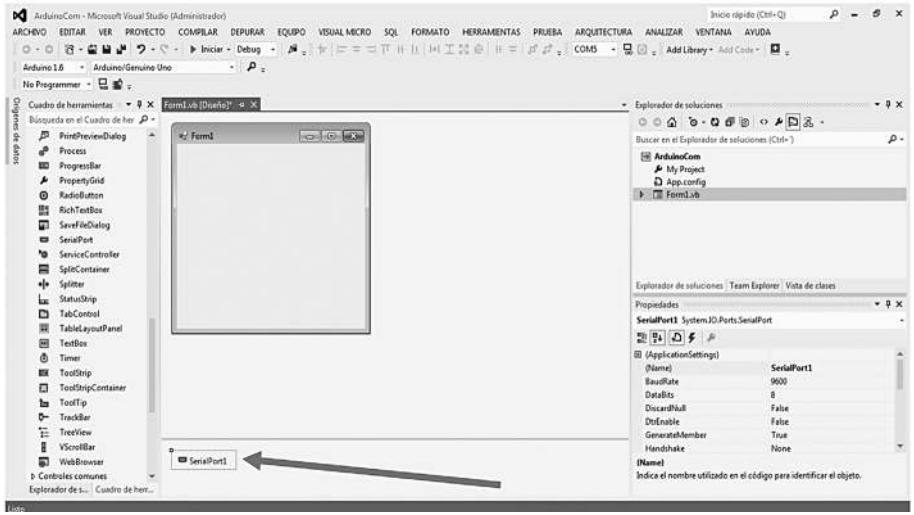


Figura 3.18 Arrastrar el control SerialPort al formulario.

Una vez que el control SerialPort está en el formulario, es posible ver las propiedades del control. Se requiere que dichas propiedades queden configuradas tal como se muestra en la figura 3.19.

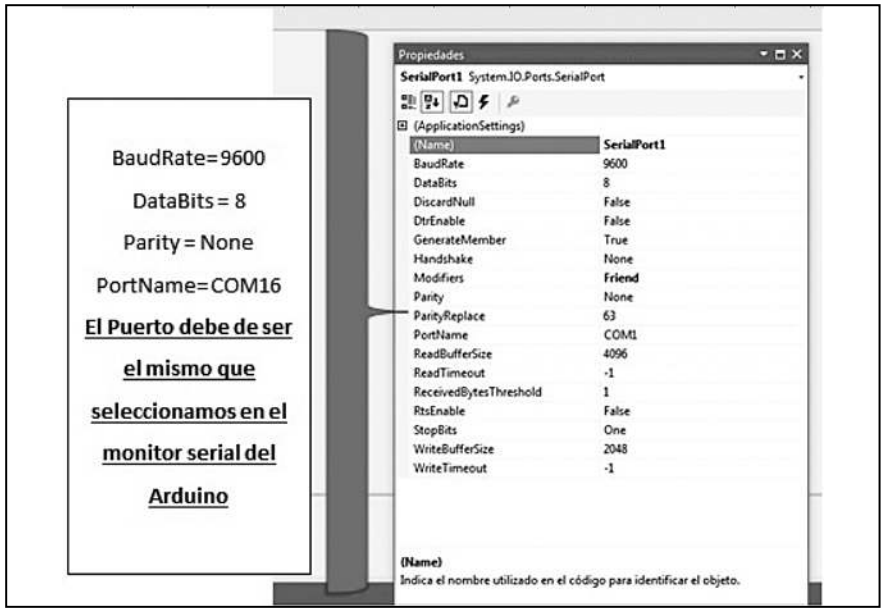


Figura 3.19 Configurar las propiedades del control SerialPort.

El control Timer permitirá establecer un tiempo de lectura para que el búfer no se sature y, en consecuencia, el envío y recepción de los datos estén sincronizados. En las propiedades del objeto Timer, en la propiedad Interval, se debe especificar el valor de 2000 o, mínimo, 1900 milisegundos. No se recomienda que sea menor, ya que los cambios en los valores serán muy rápidos y no se podrán visualizar correctamente. (Ver figura 3.20.)

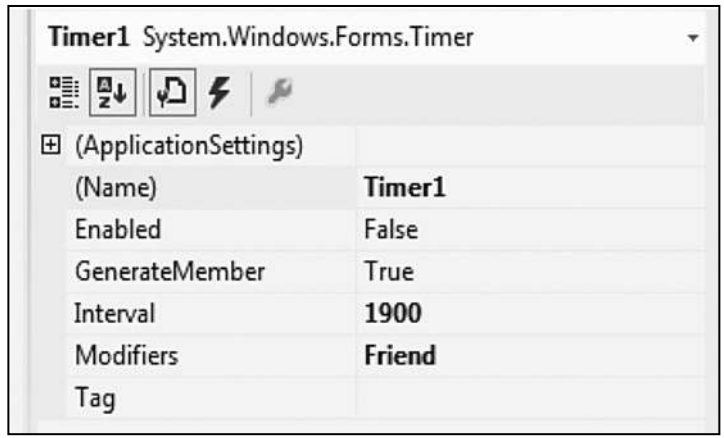


Figura 3.20 Configuración del control Timer.

3.6.2 Código para abrir el puerto y lectura de los valores enviados desde la Arduino

Para leer los datos enviados desde la tarjeta Arduino en el tiempo establecido, hay que abrir el puerto y habilitar el control Timer, como se muestra en la figura 3.21.

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        SerialPort1.Open()
        Timer1.Enabled = True
    End Sub
```

Figura 3.21 Habilitando el control Timer.

En el objeto Timer se debe establecer un código siguiendo estos pasos:

1) Declarar las variables que representan cada dato enviado.
2) Con el comando Serialport.ReadExisting, leer la cadena enviada y guardar en la variable "cadena", declarada al inicio del programa.
3) Con la función Mid, dividir cada dato de la cadena para extraer los datos de cada sensor y asignarles una caja de texto a cada uno, para que se puedan mostrar en pantalla.
4) Como resultado, los datos leídos se mostrarán en pantalla en las cajas de texto.

El código debe establecerse tal como está en la figura 3.22.

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Dim cadena1 As String
        Dim cadena2 As String
        Dim cadena3 As String

        cadena = SerialPort1.ReadExisting()

        cadena1 = Mid(cadena, 1, 5)
        cadena2 = Mid(cadena, 7, 5)
        cadena3 = Mid(cadena, 13, 5)

        txttemp.Text = cadena1.Trim
        txthum.Text = cadena2.Trim
        txtluz.Text = cadena3.Trim
    End Sub
```

Figura 3.22 Código del control Timer.

3.6.3 Resultados del sistema de los sensores en el sistema de monitoreo en tiempo real

Si se elaboró con éxito este proyecto, tendremos como resultado una estación meteorológica creada en Visual Basic .NET con entorno de programación Arduino que muestra

datos de temperatura, humedad y luz en tiempo real. La figura 3.23 muestra cómo debe verse en pantalla.

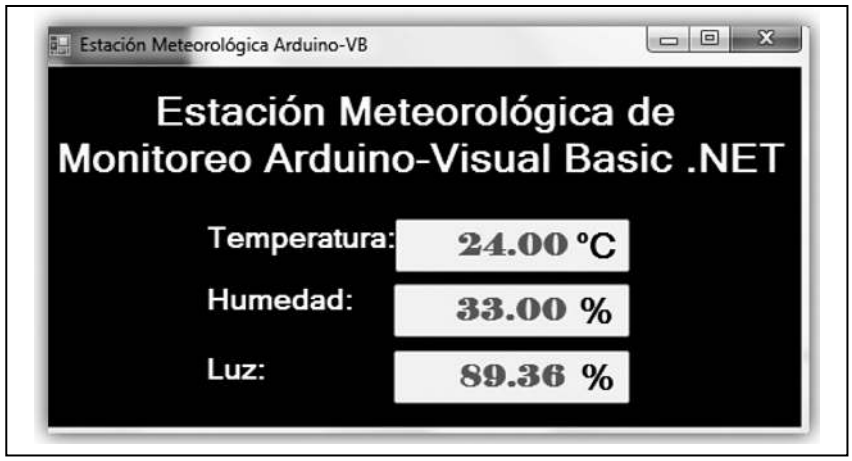


Figura 3.23 Pantalla de la estación meteorológica con Visual Basic .NET y el entorno de programación de Arduino.

3.7 Resumen

En este capítulo se explicó cómo construir un diseño de automatización simple del hogar: el tiempo en pantalla LCD para formar una estación con base en Arduino. Para ello, se interconectaron sensores de humedad y temperatura digitales con Arduino. También se describió cómo enviar las lecturas de dichos sensores a una pantalla, con el fin de verlos de forma gráfica.

Los conocimientos adquiridos en este diseño pueden servir para construir proyectos más emocionantes. Por ejemplo, se podrían conectar más sensores a los proyectos y mostrar sus mediciones en la pantalla LCD, o bien, se puede conectar un sensor de presión barométrica. También se pueden mantener los mismos sensores, pero representar gráficamente los datos medidos en una pantalla OLED.

3.8 Problemas

1. Diseñe un programa que permita enviar una letra y desplegar su valor en la pantalla LCD.
2. Conecte un botón a la tarjeta Arduino, tras lo que se debe mostrar en Visual Basic .NET que el botón se presionó.
3. Realice un programa en Visual Basic para el envío de mensajes desde Arduino: se activó salida tres, se prendió salida ocho, etcétera.

4. Montar una interfaz con un sensor de movimiento PIR para monitoreo desde Visual Basic.
5. Diseñe un formulario con el fin de monitorear un sensor de temperatura LM35DZ.
6. Desarrolle una etapa de control de temperatura: si el valor sobrepasa los 20 grados, enviar una señal de control al Arduino para encender una secuencia de leds conectados a la tarjeta.
7. Conecte tres potenciómetros a la tarjeta Arduino y monitoréelos desde Visual Basic .NET.
8. Conectar un sensor de presión barométrica y monitorear su estatus.
9. Agregue un sensor de corriente para el monitoreo de ésta.
10. Añada un sensor de gotas de lluvia para evaluar la cantidad de agua.

Capítulo 4

Detección de presencia inalámbrica con módulos XBee (sensores inalámbricos)

Objetivos

En este capítulo se aprenderá a trabajar con los módulos de comunicación XBee para realizar una alarma inalámbrica. A lo largo del proyecto se explicará cómo integrar las tecnologías de Arduino y Visual Basic .NET. Asimismo, se aprenderá a conectar un sensor de movimiento PIR y a realizar la interfaz de comunicación, con el fin de interconectar los módulos de comunicación con el software de ésta.

4.1 Introducción

En este capítulo se construirá un proyecto con un sensor de domótica muy común: el sensor de movimiento (PIR). Ejemplos de éstos se ven comúnmente en las esquinas superiores en algunas habitaciones de las casas. Son pequeños módulos de plástico blanco que cambian a un color rojo cuando se camina delante de ellos. En este proyecto, sin embargo, en lugar de utilizar las tecnologías propietarias de estos módulos, se utilizará el sistema de Arduino. También, para la comunicación, se utilizarán los módulos XBee, que son módulos de radio de baja potencia de uso generalizado con la plataforma Arduino. Estos módulos se basan en el protocolo ZigBee, que también se utiliza en muchos sistemas de automatización.

Los aspectos más importantes de este capítulo son: familiarizarse con los sensores de movimiento PIR y los módulos XBee, probar el sensor de movimiento, establecer la comunicación entre éste y el puerto serial en el momento en que se detecta movimiento y realizar una interfaz en Visual Basic .NET para monitorear los sensores.

4.2 Requerimientos de software y hardware

En este proyecto, se necesitarán placas Arduino, sensores de movimiento PIR, algunos módulos XBee y un escudo (shield) XBee. Los componentes dependerán del número de sensores que se quieren tener en el sistema. Por cada sensor se requieren los siguientes componentes:

- Arduino UNO
- Sensor de movimiento PIR
- Dos módulos XBee Serie 2
- Shield XBee con relevadores
- XBee explorer de Sparkfun, para programar los módulos XBee

El sensor de movimiento debe tener tres pines: dos para la fuente de alimentación y uno para la señal. También debe utilizar un nivel de tensión de 5V, para ser compatible con la tarjeta Arduino, que también funciona a 5V.

En cuanto al módulo XBee, se requiere un módulo XBee Serie 2, 1 mW, con una antena de rastreo (lo que significa que no requiere ninguna antena externa), la cual se deberá conectar al módulo, con lo cual, éste tendrá un alcance de aproximadamente 100 metros, sin obstáculos.

También tendrá que conectarse el módulo XBee a la placa Arduino. Para ello, cada uno de los módulos del sensor de movimiento usará un Shield Arduino XBee. Sólo se requiere hacer las conexiones entre el módulo XBee y la placa Arduino.

Para comunicarse con estos módulos XBee desde la computadora se necesita otro módulo XBee, también Serie 2, 1 mW, con antena, conectado al explorador XBee de Sparkfun, que es básicamente una placa de interfaz USB donde se puede conectar cualquier módulo XBee.

En cuanto al software, es necesario tener la última versión del IDE de Arduino instalado en la computadora. Para este proyecto se utilizó 1.6.5 Arduino IDE.

Para configurar los módulos XBee, se requiere tener ya instalado el software XCTU. Éste se puede encontrar en la siguiente URL:

http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/xctu

4.3 Configuracíón del hardware

Para conectar el hardware se debe usar un escudo XBee. (Ver figura 4.1.)

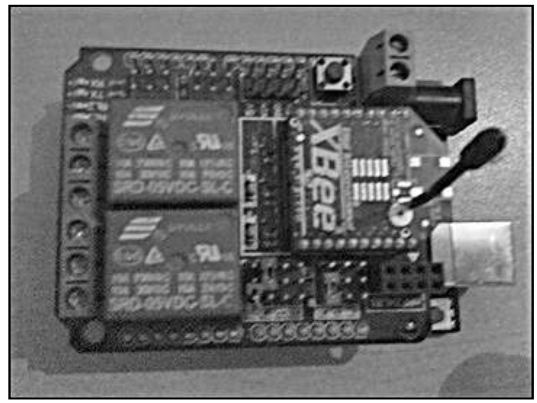


Figura 4.1 El escudo XBee.

Puesto que en éste y en el siguiente capítulo se trabajará con el escudo XBee, se recomienda familiarizarse con sus partes, las cuales se describen en el diagrama de la figura 4.2.

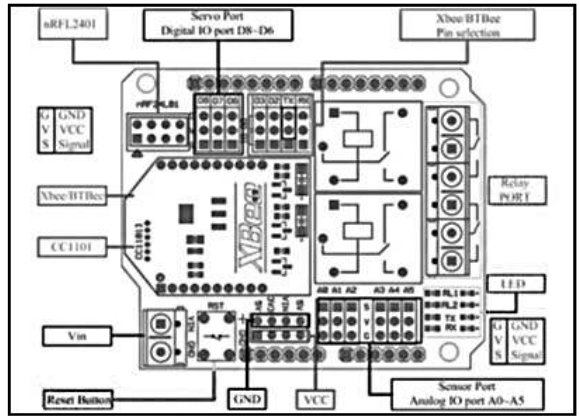


Figura 4.2 Diagrama del módulo XBee.

Para fines prácticos, con el fin de conectar el sensor de movimiento se utilizará la entrada de señal 8, la señal de voltaje de 5 volts y tierra GND. (Ver figura 4.2.)

El sensor de movimiento tiene tres pines: VCC (para lo positivo y fuente de alimentación), GND (que corresponde al nivel de tensión de referencia) y SIG (que se convertirá en un estado ALTO digital, en cuanto detecte cualquier movimiento). Conectar el VCC al pin del escudo +5V; luego, conectar los pines GND y SIG en la entrada digital de cualquier pin del escudo, como se muestra en la figura 4.3 (nótese que en este ejemplo se utiliza el pin 8, aunque se puede utilizar cualquier pin digital).

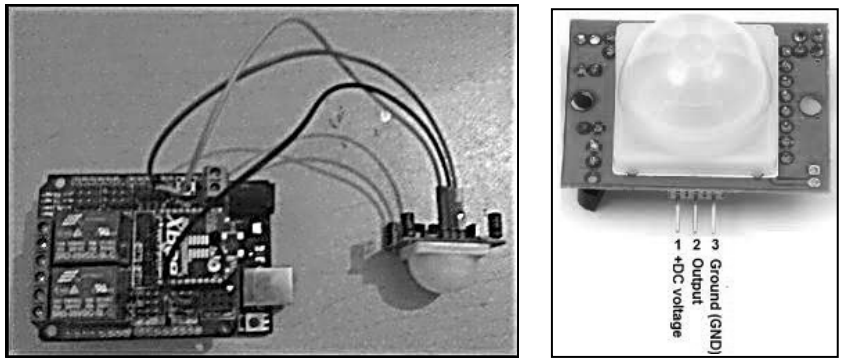


Figura 4.3 Conexión del shield XBee con el sensor de movimiento.

Para poder subir un boceto (sketch), se debe establecer correctamente un puente en el tablero. Sobre el escudo XBee hay un pequeño interruptor (cerca del módulo XBee), ya sea para activar la conexión directamente a la placa Arduino (interfaz en serie), con la cual ya no se puede subir bocetos, o bien, desactivar la conexión. Puesto que se debe subir el boceto Arduino primero, hay que cambiar el interruptor a Dline.

Es importante considerar que la tarjeta tiene pines con jumpers. Para poder descargar el boceto, éstos deben quitarse, con el objetivo de poder descargar el programa mediante los D0 y D1. Después de descargar el boceto, poner nuevamente los jumpers sobre D0 y D1 para que la tarjeta Arduino pueda establecer comunicación con el módulo XBee. (Ver figura 4.4.)

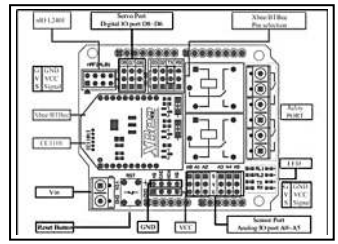


Figura 4.4 Interruptor en Dline del escudo XBee.

También se debe conectar la placa del explorador XBee. Basta con insertar el módulo XBee a la placa, como se muestra en la figura 4.5.

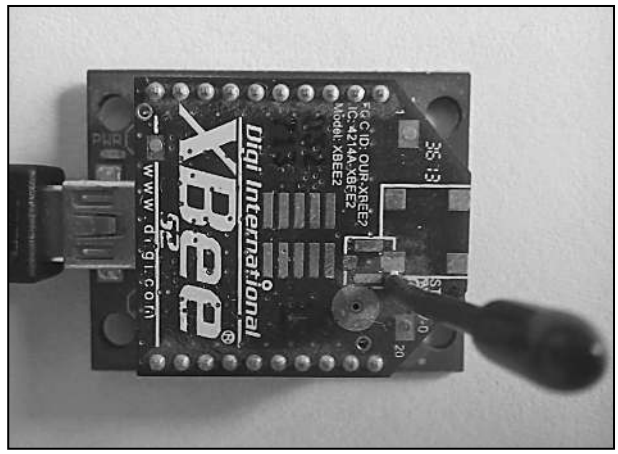


Figura 4.5 Insertar el módulo XBee en la placa.

Una vez que todos los componentes XBee están conectados, el siguiente paso es conectar la tarjeta Arduino y el explorador a la computadora a través de cables USB.

Si se quiere incluir más sensores de movimiento, será necesario repetir el procedimiento por cada sensor que se instale. Es decir, por cada sensor se requiere un escudo XBee, un sensor de movimiento y un módulo XBee S2; asimismo, el explorador USB debe conectarse a la computadora.

4.4 Establecer la interfaz del el sensor PIR con Arduino

En esta sección se describrirá la prueba de comunicación entre la tarjeta Arduino y el sensor de movimiento. A continuación se describe el programa para probar el sensor.

```
// Prueba de sensor de movimiento
int sensor = 8;

void setup() {
  Serial.begin(9600);
  pinMode(sensor,INPUT);
}
void loop(){
   // lectura del sensor
  int estado = digitalRead(sensor);
  Serial.print("Detector de sensor:");
  Serial.println(estado);
  delay(100);
}
```

```
Detector de sensor: 1
Detector de sensor: 1
Detector de sensor: 1
Detector de sensor: 1
Detector de sensor: 1
Detector de sensor: 0
```

Figura 4.6 Prueba en pantalla del sensor PIR.

4.5 Programación del módulo XBee

En esta sección se explicará cómo preparar el módulo XBee. Primeramente, se debe insertar el módulo XBee S2 en el explorador USB y conectarlo a la computadora. Después, hay que abrir la interfaz como se muestra en la figura 4.6. Finalmente, si es correcto, se debe agregar un dispositivo, en este caso el sensor, y esperar a que el programa lo detecte. Es importante tomar en cuenta que el módulo que se va conectar al explorador y a la computadora es el dispositivo que recibirá la información de los sensores de movimiento. Éste es el módulo que establece la comunicación; es decir, a él van a llegar las detecciones de cada sensor (si hay más de uno). En otras palabras, el Coordinator AT es el punto central de la comunicación. (Nótese en la figura 4.6 que el ID PAN ID se configuró con el número 66.)

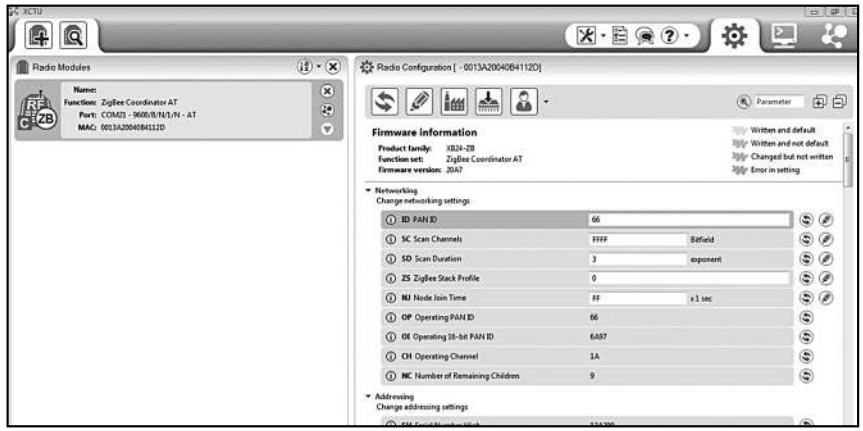


Figura 4.6 Interfaz con el módulo XBee.

Para configurar el módulo XBee que se va a conectar al escudo hay que utilizar la configuración End Device AT con el mismo ID PAN ID con que se configuró el dispositivo Coordinator AT; de esta manera se establece la comunicación entre los módulos inalámbricos. Asimismo, si se requiere de más sensores, habrá que configurar cada

módulo al que se le va a conectar el sensor como End Device, para que se establezca la red de comunicación y se puedan enviar los datos al punto central (Coordinator AT).

4.5.1 Código del detector de presencia inalámbrico

Para configurar la detección de presencia, se utiliza la función loop; en ella se lee el estado de la señal donde está conectado el sensor de movimiento PIR. Ahí hay que incluir la siguiente condición (recordar el control "if" de la sección 1.2.8): si el sensor detecta un estado alto, enviar por el puerto serial la palabra "alto"; si lee un estado bajo, enviar la palabra "bajo". El formato de instrucción es el siguiente:

```
if (sensor_state == HIGH){
    Serial.println("alto");
  }
 if (sensor_state == LOW){
   Serial.println("bajo");
  }
  delay(500);
```

4.6 Creación de la interfaz gráfica del detector de presencia

Para establecer comunicación con el detector de movimiento, en el programa abrir la conexión con el puerto serial; dentro de la función Timer se lee el valor recibido. El puerto serial recibe el valor enviado por Arduino y con la función de Cadena se toma el valor alto. El formato de instrucción es el siguiente:

```
Dim cadena1 As String
cadena = SerialPort1.ReadExisting()
cadena1 = Mid(cadena, 1, 4)
TextBox1.Text = cadena1.Trim
```

Notar que la instrucción debe establecerse de tal manera que el valor de la cadena sea igual a cuatro caracteres para poder comparar este valor. Así, si el programa lee un valor alto, se prende el óvalo en color verde; cuando no detecta un valor alto, se prende en color rojo. El formato de instrucción es el siguiente:

```
If cadena1 = "alto" Then
            OvalShape1.BackColor = Color.Green

        Else
            OvalShape1.BackColor = Color.Red
          End If
```

El código final quedaría como se establece a continuación:

```
Public Class Form1
    Dim cadena As String

    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        SerialPort1.Open()
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As Even-
tArgs) Handles Timer1.Tick
        Dim cadena1 As String
        cadena = SerialPort1.ReadExisting()
        cadena1 = Mid(cadena, 1, 4)
        TextBox1.Text = cadena1.Trim

        If cadena1 = "alto" Then
            OvalShape1.BackColor = Color.Green

        Else
            OvalShape1.BackColor = Color.Red
          End If
    End Sub
End Class
```

La figura 4.7 muestra cuando el sensor inalámbrico no detecta movimiento y la figura 4.8 muestra cuando el sensor inalámbrico sí detecta movimiento.

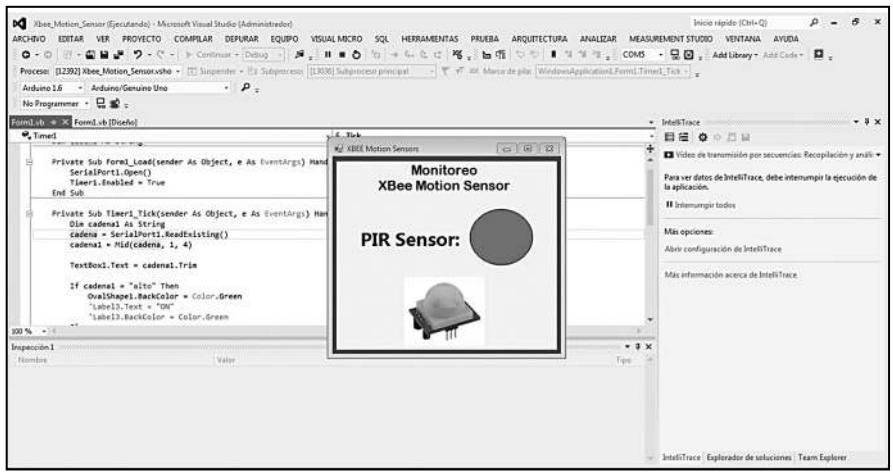


Figura 4.7 El sensor no detecta movimiento.

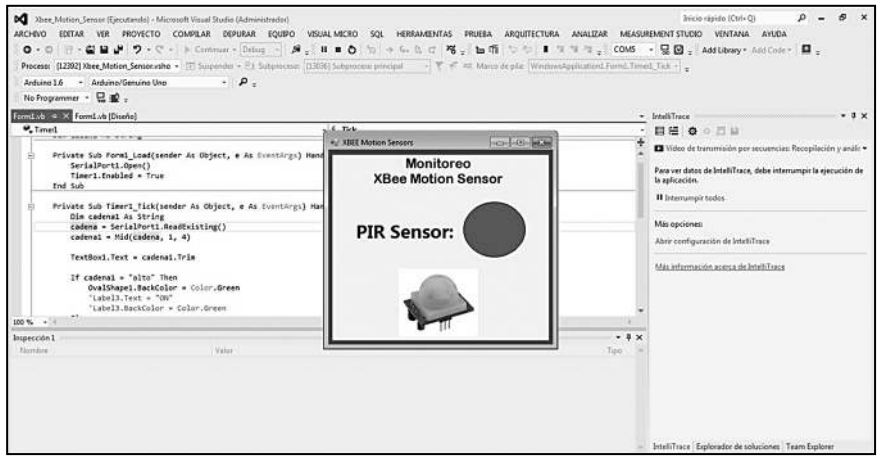


Figura 4.8 El sensor sí detecta movimiento.

4.7 Otros ejemplos con esta misma aplicación

Este mismo procedimiento se puede aplicar para un sensor magnético Reed Switch para detectar cuando una puerta o ventana está abierta o cerrada. (Ver figuras 4.9 y 4.10.)

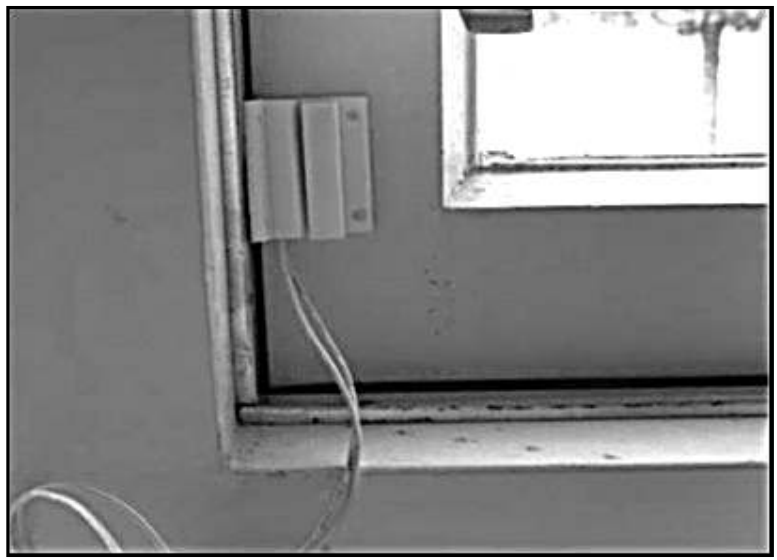


Figura 4.9 Sensor Reed Switch instalado en la ventana.

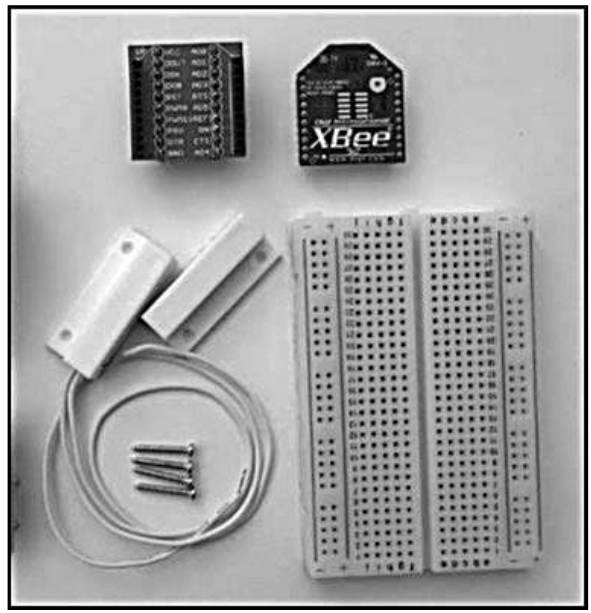


Figura 4.10 Componentes para detectar si una ventana está abierta o cerrada.

El sensor Reed Switch da como salida un "0" cuando detecta el campo magnético y "1" cuando el campo está lejos. De esta manera se puede condicionar cuando la ventana está abierta o cerrada.

Para establecer esta instrucción con el programa de Arduino se debe seguir el procedimiento a continuación. La entrada del sensor pin 8 se declara como entrada "PULL-UP", para que Arduino pueda leer correctamente la señal, que se comporta como un contacto normalmente abierto.

El formato de instrucción es el siguiente:

```
void setup() {
  pinMode(sensor, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  estado = digitalRead(sensor);

  if (estado == LOW){
    Serial.println("puerta CERRADA");
  }
if (estado == HIGH){
    Serial.println("puerta ABIERTA");
  }
 }
```

Las figuras 4.11 y 4.12 muestran cómo Arduino lee e interpreta los datos del sensor Reed Switch.

Figura 4.11 La puerta está abierta.

Figura 4.12 La puerta está cerrada.

4.8 Resumen

En este capítulo se aprendió a conectar el sensor de movimiento PIR, a establecer comunicación entre éste y Arduino y a implementar un sensor inalámbrico a través del módulo XBee. Por otro lado, se explicó cómo usar y aplicar la herramienta de Visual Studio 2012 para realizar las interfaces HMI y poder visualizar y monitorear el sensor inalámbrico de movimiento. Esto mismo se puede aplicar en otro tipo de proyectos; se puede añadir cualquier tipo de sensor y hacer una red de sensores, por ejemplo, sensores de movimiento en cada cuarto, o bien, sensores de humedad, temperatura, luz, gas, etcétera.

En el siguiente capítulo se elaborarán nuevos proyectos de automatización y control que también utilizan módulos XBee.

4.9 Problemas

1. Realice una interfaz de monitoreo para un sensor de temperatura LM35DZ.
2. Conecte un sensor de temperatura y humedad DHT11 y monitoree los valores de forma inalámbrica.
3. Elaborar desde Visual Basic la interfaz para un timbre de forma inalámbrica con monitoreo.
4. Diseñe la interfaz para el control de leds en una secuencia y en secuencia inversa de forma inalámbrica.
5. Planee el control de un semáforo simple de forma inalámbrica, considere tiempos de encendido y apagado.
6. Realice el control de un semáforo crucero, considere programación de tiempos de encendido y apagado.
7. Lleve a cabo un contador de personas de forma inalámbrica; realice la interfaz de conteo desde Visual Basic.
8. Haga el monitoreo de forma inalámbrica de un sensor de presión barométrica.
9. Elabore un medidor de corriente inalámbrico con interfaz y monitoreo.
10. Realice el monitoreo de sensores inalámbricos, LM35DZ, DHT11, fotorresistencia, presión barométrica, diseño de interfaz de monitoreo.
11. Diseñe un sistema de monitoreo con sensor PIR; cuando detecte movimiento, enviar un activación de forma inalámbrica; además realice el monitoreo y control desde Visual Basic .NET.

Capítulo 5

Control de las luces desde una interfaz HMI

Objetivos

En este capítulo se explicará cómo conectar un módulo de relevador a la tarjeta Arduino, con el fin de aprender a controlar inalámbricamente dispositivos de luces desde una interfaz HMI.

5.1 Introducción

En el capítulo 4 se realizó una interfaz en lenguaje de programación Visual Basic .NET a través de la cual se adquirían datos provenientes de las señales de sensores para mostrar e interpretar su funcionamiento y monitorearlo desde una pantalla central. Tales conocimientos servirán como base para el proyecto de este capítulo: controlar las luces desde una interfaz HMI, para lo cual se explicará cómo hacer lo siguiente:

- Probar los relevadores de forma inalámbrica activándolos desde la interfaz de comunicación Serial de Arduino.
- Utilizar el módulo de relevadores con XBee y la tarjeta Explorer conectada a la computadora a través del puerto serial de la PC para controlar de forma inalámbrica los relevadores.
- Controlar los relevadores de forma inalámbrica desde la interfaz HMI.
- Desarrollar una interfaz para página web con la finalidad de controlar de forma remota los relevadores utilizando el servidor web IIS (*Internet Information Server*) y la aplicación ASP.NET.
- Utilizar otra manera de comunicar nuestra aplicación de escritorio para control de forma remota a través de servicios web.

5.2 Requerimientos de software y hardware

Para este proyecto se utilizarán los siguientes materiales:

- Tarjeta Arduino UNO
- Módulo XBee Serie 2
- Shield XBee con relevadores
- XBee explorer para programar los módulos XBee

El software requerido para este proyecto es el mismo que se utilizó para el capítulo anterior, que debe tener las siguientes configuraciones:

- Configurar los módulos XBee: uno de ellos como coordinador y el otro como dispositivo final.
- Probar la conexión entre ambos dispositivos.
- Realizar el programa requerido para lectura de las señales enviadas desde el dispositivo coordinador.

5.3 Configuración del hardware

Recordemos que el escudo XBee cuenta con dos relevadores para conectar cargas de potencia, como se muestra en la figura 5.1.

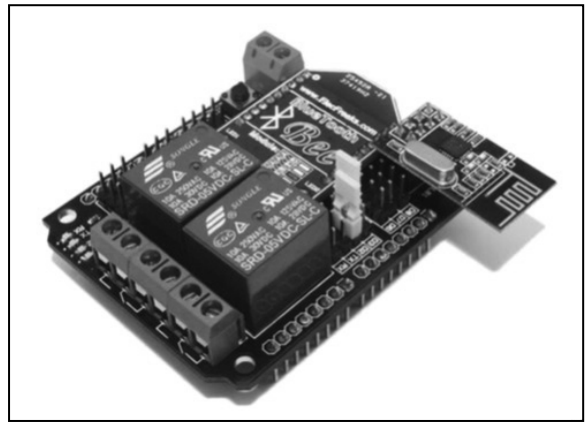


Figura 5.1 El escudo XBee y sus dos relevadores.

5.3.1 Conexión de la tarjeta Arduino con los relevadores

Asimismo, los relevadores deberán conectarse a las salidas digitales D4 y D5 del Arduino; éstos nos permitirán controlar las salidas de potencia que vamos a utilizar en este proyecto. (Ver figura 5.2.)

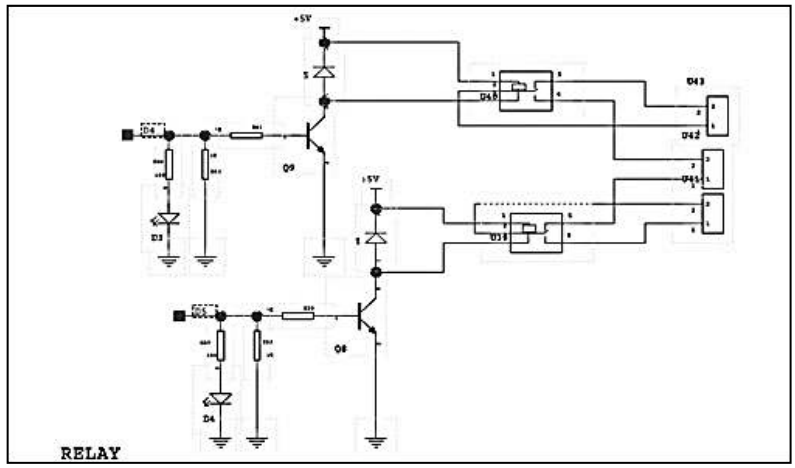


Figura 5.2 Diagrama de hardware de los relevadores.

5.3.2 Conexión del foco al relevador

Para conectar el foco hay que conectar el común del relevador (ver conexión COM de la figura 5.3) a la clavija de la luz y la conexión. Normalmente abierto (ver conexión NO de la figura 5.3) se conecta al foco.

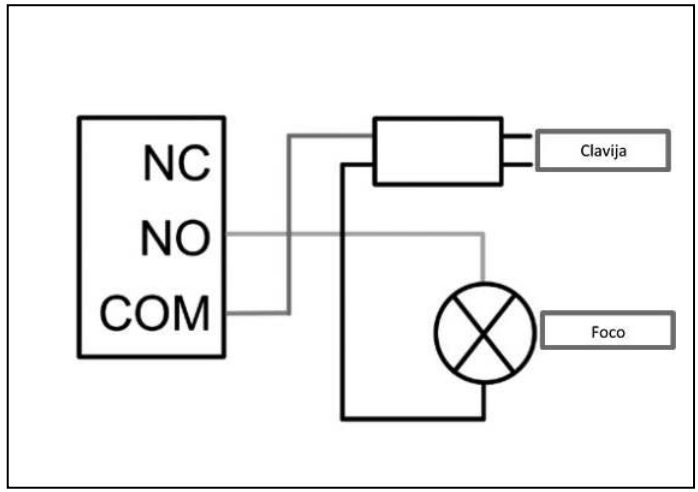


Figura 5.3 Conexiones del relevador.

La figura 5.4 demuestra cómo debe verse la conexión del foco con el relevador del escudo XBee con la Arduino.

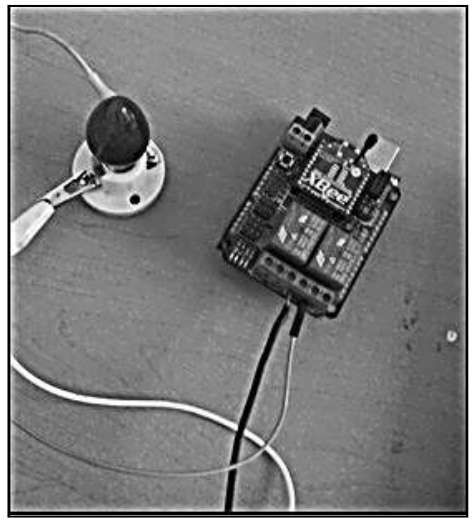


Figura 5.4 Foto de la conexión del foco con el relevador del escudo.

5.4 Prueba de los relevadores

Es importante considerar que, para que ambos dispositivos establezcan comunicación, el dispositivo nodo final envía un valor cada segundo; en este caso, un valor que se va

incrementando. Este valor enviado permite que el dispositivo coordinador se mantenga conectado y esté recibiendo los valores para poder controlar lo que se le pide.

A continuación se muestra la programación del dispositivo nodo final, en donde se configuran las variables del programa con la siguiente instrucción:

```
int luz1 = 4;
int luz2 = 5;
char leer;
int contador = 0;
```

También se configura la comunicación serial y la velocidad a 9600:

```
void setup()
{
  Serial.begin(9600);     //Inicia comunicación Serial
  pinMode(luz1,OUTPUT);   //Configurar luz como una salida
  pinMode(luz2,OUTPUT);   //Configurar luz como una salida
}
```

Para que el dispositivo final se mantenga conectado al dispositivo coordinador, se envía un valor por el monitor serial con la siguiente instrucción:

```
void loop()
{
 contador++;
 Serial.println(contador);
 delay(100);
```

Para activar la lectura del valor enviado, programar con la siguiente instrucción:

```
leer = Serial.read();
```

También hay que crear las condiciones para encender los relevadores:

```
 //condición para luz 1
    if (leer == 'a')
    {
     digitalWrite(luz1,HIGH);
    }

    if (leer == 'b')
    {
     digitalWrite(luz1,LOW);
    }

//condición para luz 2
    if (leer == 'c')
    {
     digitalWrite(luz2,HIGH);
    }

    if (leer == 'd')
    {
     digitalWrite(luz2,LOW);
    }
 }
```

Si todo es correcto, para probar la comunicación y el envío de los datos entre los dispositivos, la pantalla de comunicación serial debe verse similar a la figura 5.5. Se puede capturar una letra "a" para encender el primer relevador y si se presiona la letra "b" se apagará el mismo relevador. Se puede seguir el mismo proceso para encender el relevador 2.

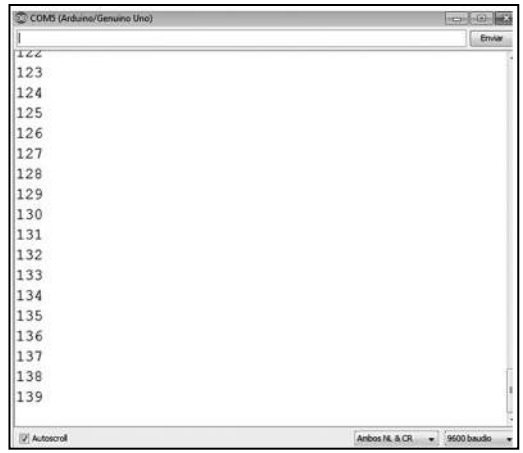


Figura 5.5 Pantalla de comunicación serial entre la tarjeta Arduino y la luz.

Asimismo, en la figura 5.6 se pueden observar las pruebas de comunicación para activar los relevadores desde la interfaz de comunicación serial de Arduino.

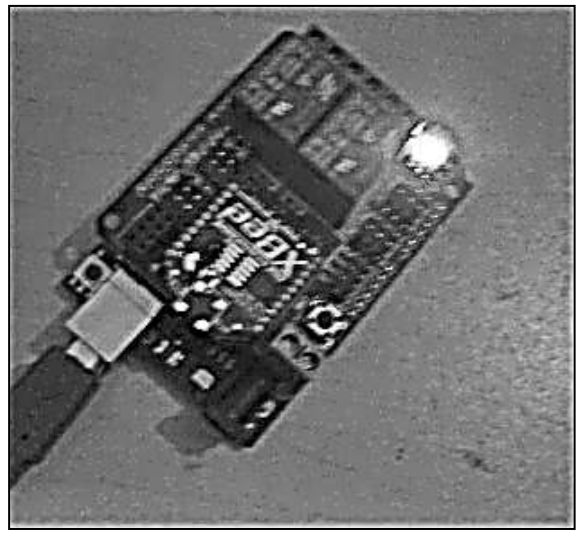


Figura 5.6 Pruebas de comunicación.

5.5 Creación de la interfaz gráfica para control de los relevadores

La pantalla de la interfaz contiene dos botones para controlar cada relevador: uno para el encendido y el otro para el apagado. (Ver figura 5.7.)

Figura 5.7 Pantalla de la interfaz.

La configuración del código debe hacerse como se muestra a continuación:

1) Se abre el puerto de comunicación.
2) Se envían por el puerto serial los caracteres que serán leídos en la placa Arduino ("a", "b", "c" y "d").
3) El dispositivo nodo final que recibe los caracteres tiene la condición de que si recibe el carácter correspondiente a la letra "a", por ejemplo, debe encender la salida digital 4 del Arduino; de tal forma, hace lo mismo con las otras letras que se definan en las condiciones.

El formato de instrucción es el siguiente:

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        SerialPort1.Open()
    End Sub

    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        SerialPort1.Write("a")
    End Sub

    Private Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
```

```
        SerialPort1.Write("b")
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        SerialPort1.Write("c")
    End Sub

    Private Sub Button4_Click(sender As Object, e As
EventArgs) Handles Button4.Click
        SerialPort1.Write("d")
    End Sub

End Class
```

5.6 Prueba de la interfaz de comunicación

La idea del proyecto es controlar las luces mediante una página web. Para ello, se explicará cómo realizar lo siguiente:

- Conexión de la tarjeta Arduino al servidor web a través del puerto Serial de la PC.
- Instalación del servidor de página web de Microsoft IIS (Internet Information Server) en la computadora de la que va a ser servidor.
- Creación de la aplicación web ASP.NET en Visual Basic .NET.
- Elaboración de los controles para el control de los relevadores.
- Control de la tarjeta Arduino a través de la tecnología de servicios web.

Es importante considerar que la tarjeta Arduino estará conectada directamente al servidor web de forma serial, es decir, no estará utilizando el puerto Ethernet como medio de comunicación. Esto resulta muy interesante, ya que se están añadiendo los recursos de la plataforma .NET a la tecnología de Arduino. De tal forma, el equipo que se conectará a la red será la PC, que fungirá como servidor; asimismo, desde un cliente web remoto se controlará la tarjeta Arduino. (Ver la figura 5.8.)

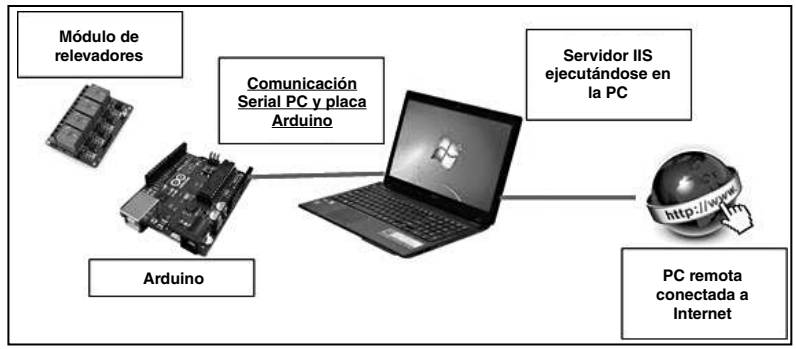


Figura 5.8 Topología del control de la placa Arduino desde una página web.

5.6.1 Aplicación web ASP.NET para control mediante comunicación serial

Para crear la aplicación de la página web que controlará los relevadores de forma remota se deben realizar los siguientes pasos:

1) En el menú Nuevo, seleccionar un proyecto de tipo Sitio web. (Ver figura 5.9.)

Figura 5.9 Selección de proyecto para sitios web.

2) Como se muestra en la figura 5.10, se despliega un menú de diferentes sitios. Hay que seleccionar un proyecto de tipo ASP.NET en blanco. Luego, hay que nombrar la ubicación como HTTP y nombrar al proyecto (ver el recuadro inferior de la figura 5.10). Al final, hacer clic en el botón Aceptar.

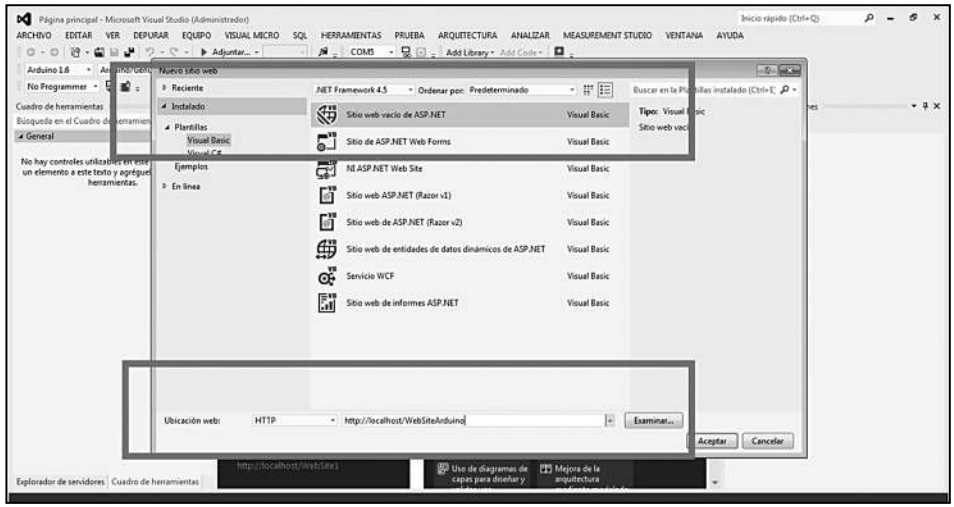


Figura 5.10 Selección y nombramiento del sitio web.

3) Agregar una página nueva (ver figura 5.11). Para acceder a este menú se da clic con el botón derecho sobre el nombre del proyecto, se selecciona Agregar y se elige la opción de Agregar nuevo elemento. Hacemos clic en en el menú Agregar nuevo elemento de tipo página WEB ASPX. Formularios WEB de tipo ASPX.

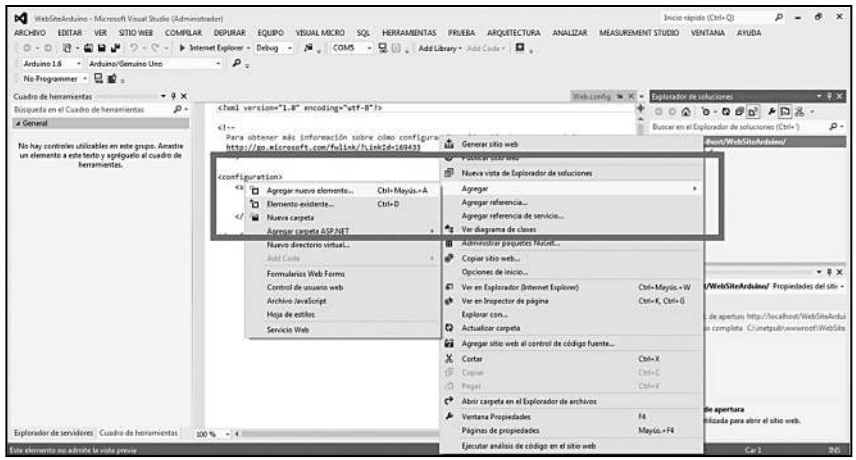


Figura 5.11 Agregar la nueva página.

Cabe mencionar que el nombre se deja como aparece: .aspx (ver en la figura 5.12 el recuadro y la parte inferior de la sección Nombre). Luego se tiene que hacer clic en Agregar.

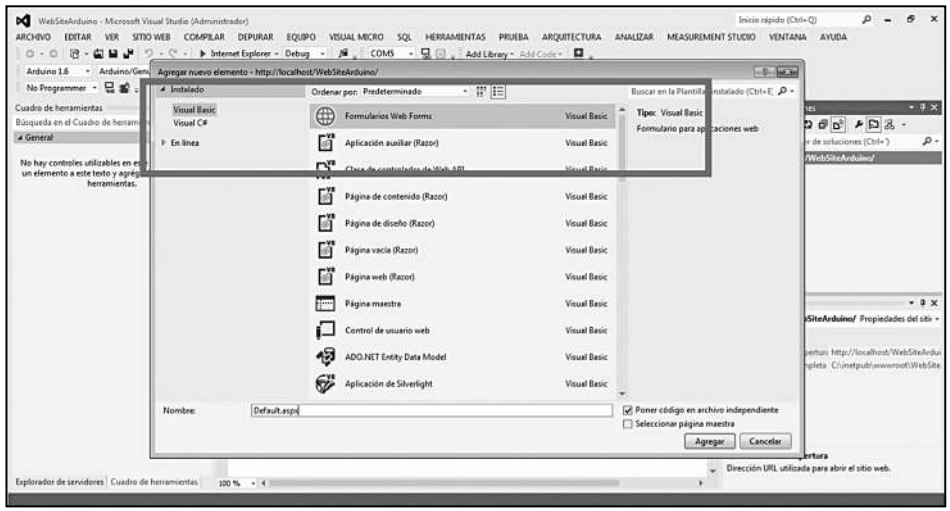


Figura 5.12 Nombrar la página.

4) Una vez que se realizó la interfaz, hacer clic en el botón Explorer para ejecutar la página web. Este botón aparece en el menu de herramientas al lado del botón DEBUG. La pantalla de depuración aparece porque es la primera vez que se ejecuta la página web y se está habilitando el archivo WebConfig. Hacer clic en el botón de Aceptar. (Ver figura 5.13.)

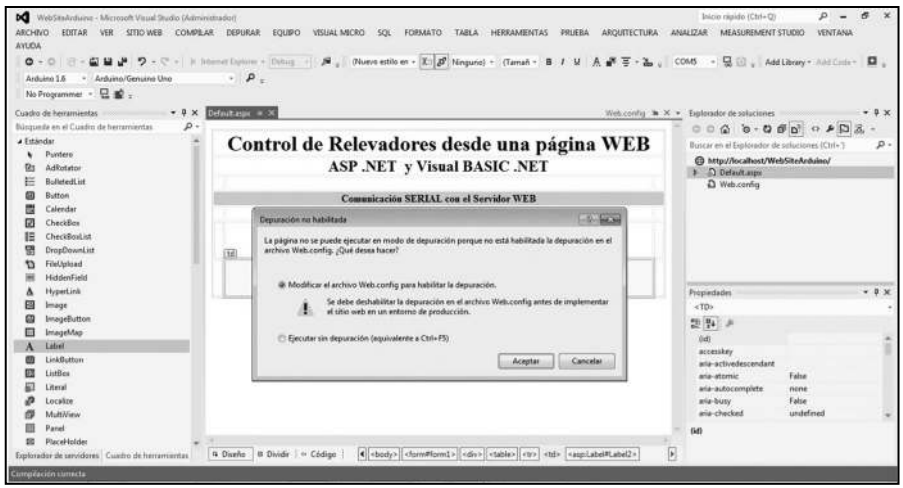


Figura 5.13 Ejecutar la página web.

La figura 5.14 muestra cómo debe verse la interfaz.

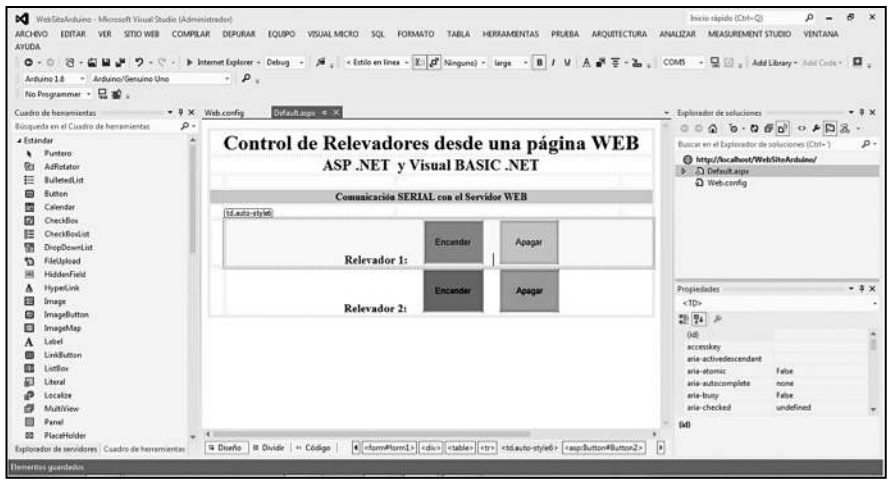


Figura 5.14 Interfaz para el control.

Ahora bien, la página web debe verse como se muestra en la figura 5.15.

Figura 5.15 Pantalla de la página web.

El formato del código de la aplicación es el siguiente:

```
Imports System.IO.Ports
Partial Class _Default
Inherits System.Web.UI.Page

    Protected Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim serialPort As New SerialPort("COM8")
        serialPort.BaudRate = 9600
        serialPort.Open()
        serialPort.Write("1")
        serialPort.Close()
    End Sub

    Protected Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Dim serialPort As New SerialPort("COM8")
        serialPort.BaudRate = 9600
        serialPort.Open()
        serialPort.Write("2")
        serialPort.Close()
    End Sub

    Protected Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
        Dim serialPort As New SerialPort("COM8")
        serialPort.BaudRate = 9600
        serialPort.Open()
        serialPort.Write("3")
        serialPort.Close()
    End Sub

    Protected Sub Button4_Click(sender As Object, e As
EventArgs) Handles Button4.Click
```

```
        Dim serialPort As New SerialPort("COM8")
        serialPort.BaudRate = 9600
        serialPort.Open()
        serialPort.Write("4")
        serialPort.Close()
    End Sub
End Class
```

5) Después de haber probado el código, la placa Arduino y el foco deben encenderse como lo demuestra la figura 5.16.

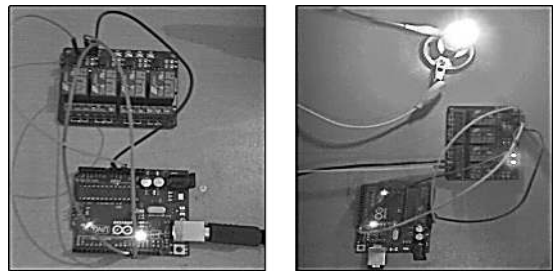


Figura 5.16 Control de encendido desde un sitio web.

Control mediante un servicio web

En el diagrama de la figura 5.17 se explica el esquema de un servicio web y los elementos que intervienen en él: aplicación, servidor y comunicación Serial.

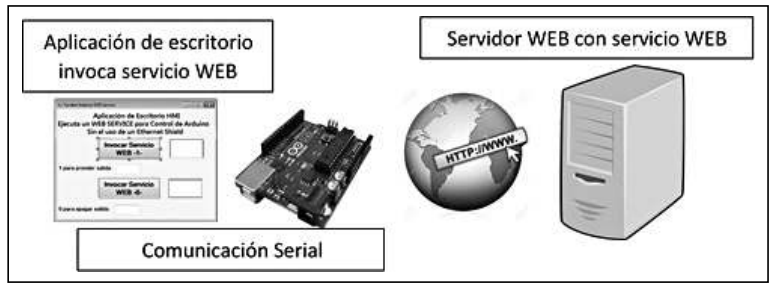


Figura 5.17 Elementos de un servicio web.

- El proceso empieza con la aplicación cliente, que invocará el servicio y enviará el dato para control de las salidas conectadas a la Arduino. Posteriormente, el servicio web ejecutará las funciones que invoca el cliente. En este proyecto se enviará el valor cero (0) o bien el uno (1), que el servicio web procesará; luego, regresará esos valores a la aplicación cliente para que sean enviados por el puerto serial. Por último, el servidor web ejecutará los servicios configurados en la aplicación cliente.

5.6.2 Creación del sitio web en Visual Basic.NET

En esta sección se presentan los pasos para crear la aplicación con servicios web utilizando Visual Basic .Net.

1) En Visual Studio, se elige un nuevo proyecto de sitio web. (Ver figura 5.18.)

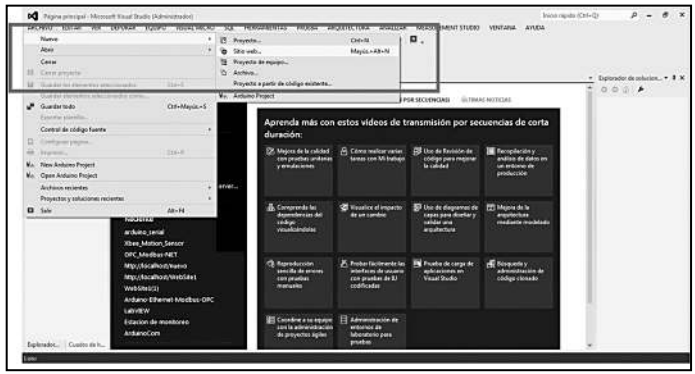


Figura 5.18 Iniciar un nuevo sitio web.

2) Después se crea la aplicación ASP en localhost, se elige la Ubicación web y se selecciona HTTP para que el proyecto se cree como tipo localhost. Esto permitirá que se agregue dentro del servidor. La ruta que se debe de colocar es la siguiente: http://locallhost/nombre_de_la_carpeta. Como se muestra en la figura 5.19.

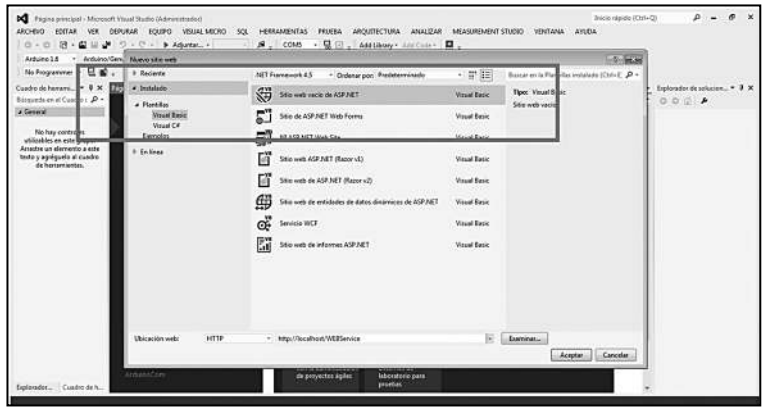


Figura 5.19 Creación de la aplicación ASP.

3) A continuación debe agregarse el formulario web, como lo muestra la figura 5.20. Se debe hacer clic con el botón derecho en el proyecto, se selecciona la opción de Agregar nuevo elemento y Agregar un formulario web de tipo aspx, en donde debe darse clic al comando Agregar nuevo elemento.

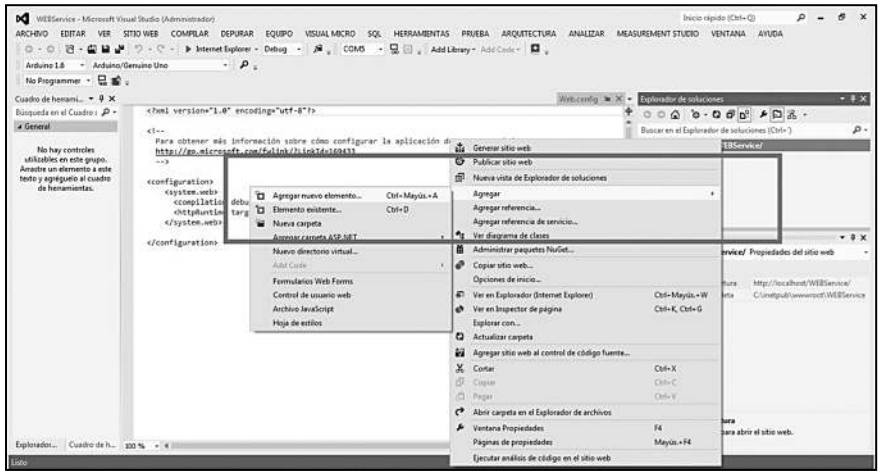


Figura 5.20 Añadir el formulario web. Comando Agregar nuevo elemento.

Después, hay que elegir el comando Formularios Web Forms. (Ver figura 5.21.)

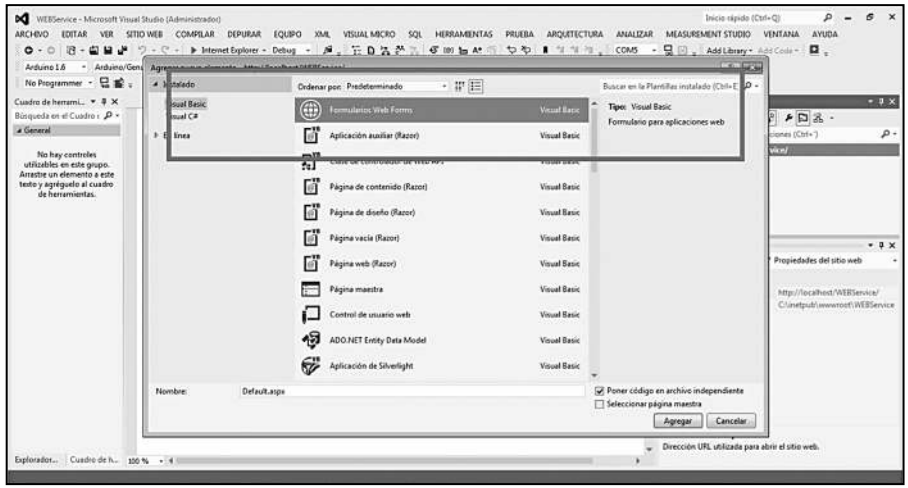


Figura 5.21 Agregar formulario de Web Forms.

4) Enseguida, crear el elemento Servicio web. Hacer clic con el botón derecho en el proyecto, después nuevo elemento de tipo Servicio web. El nombre del servicio web puede ser el mismo; en este caso dejamos el predeterminado y la extensión no se debe cambiar. La extensión es de tipo .asmx. (Ver figura 5.22.)

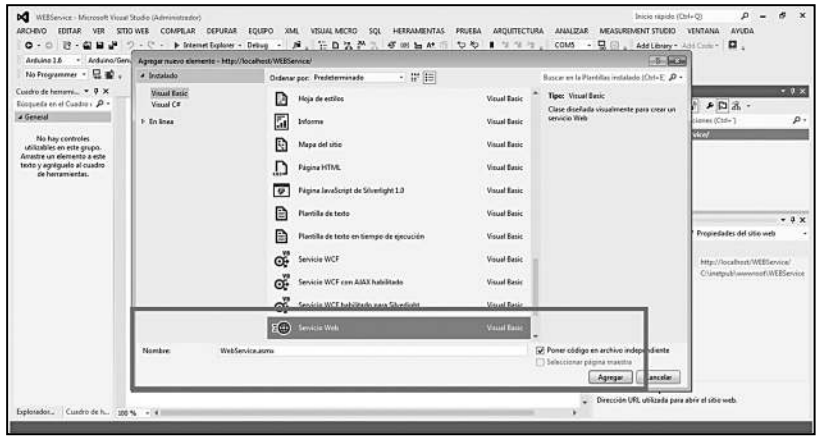


Figura 5.22 Agregar el elemento de servicio web.

El servicio web debe verse como lo muestra la figura 5.23.

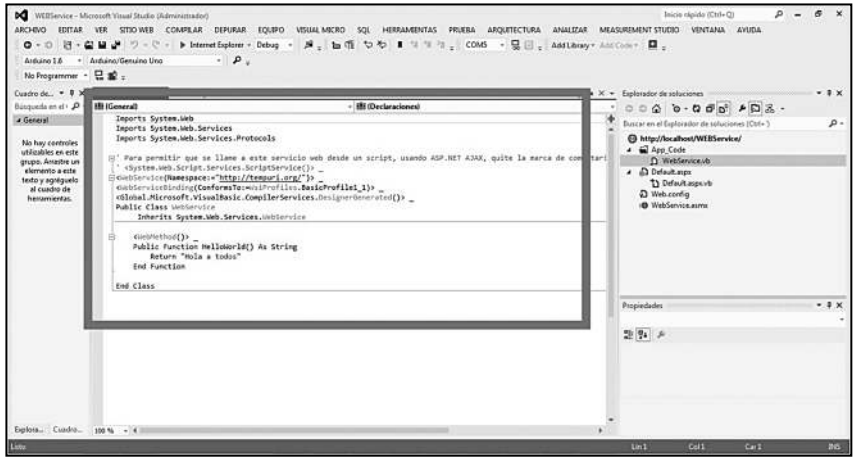


Figura 5.23 Servicio de web creado.

5) Ahora, ya se puede crear la primera función dentro de Web Method con el siguiente formato de instrucción:

```
<WebMethod()> _
    Public Function control(ByVal dato As Double) As Dou-
ble
        Dim d1 As Double
        d1 = dato
        Return d1
    End Function
```

La figura 5.24 muestra los dos métodos web que se crearon para este proyecto.

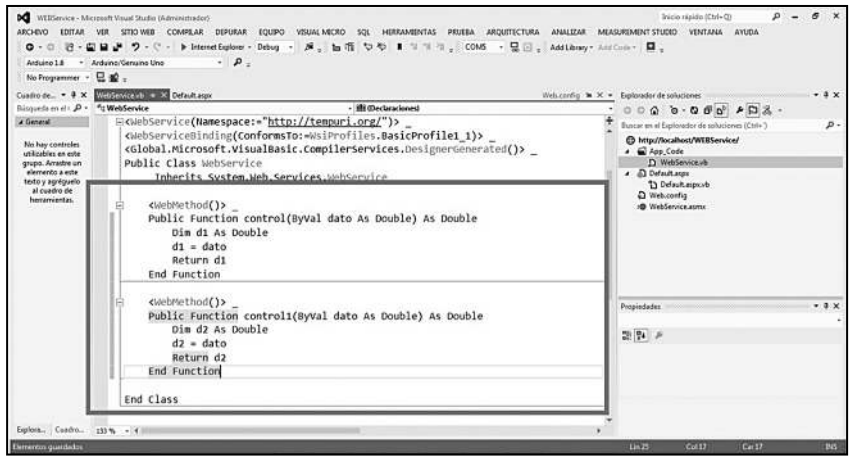


Figura 5.24 Los dos métodos web.

6) Ahora hay que generar el sitio web. Se hace clic con el botón derecho sobre el proyecto web y aparece un menú, se selecciona la opción de Generar sitio web. (Ver figura 5.25.)

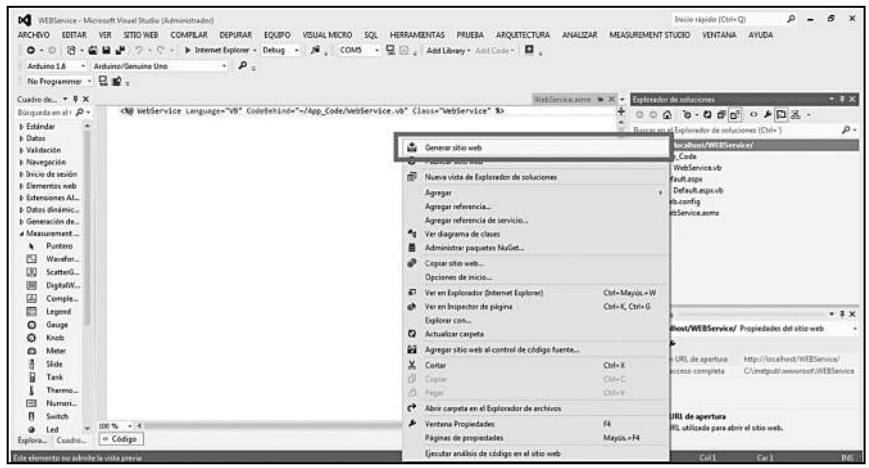


Figura 5.25 Generar el sitio web.

Después, el sitio web debe establecerse como página de inicio dando clic a Establecer como página principal; para ello ubicar primeramente el nombre del servicio web, hacer clic con el botón derecho sobre el nombre del servicio web y seleccionar Establecer como pagina principal. (Ver figura 5.26.)

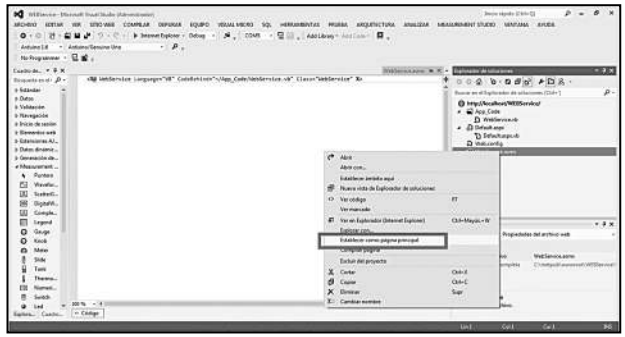


Figura 5.26 Establecer el sitio como página de inicio.

7) Posteriormente, se debe ejecutar el servicio web: seleccionar el servicio web creado previamente, después, en el menú de Herramientas seleccionar el navegador a través del cual se va a ejecutar el servicio web. O tambien se puede ejecutar haciendo clic en el menú Depuración y seleccionando Iniciar depuración. (Ver figura 5.27.)

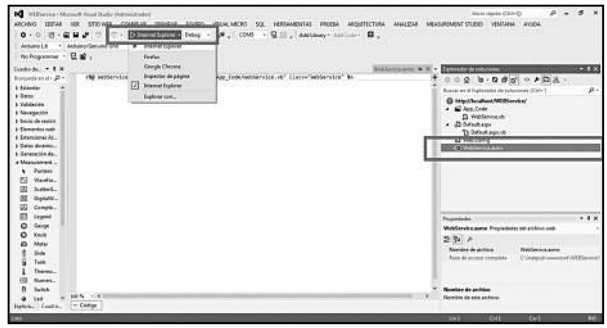


Figura 5.27 Ejecutar el servicio web.

Si sale un anuncio de "Depuración no habilitada", elegir "Modificar el archivo Web. config para habilitar la depuración" y luego dar clic en Aceptar. (Ver figura 5.28.)

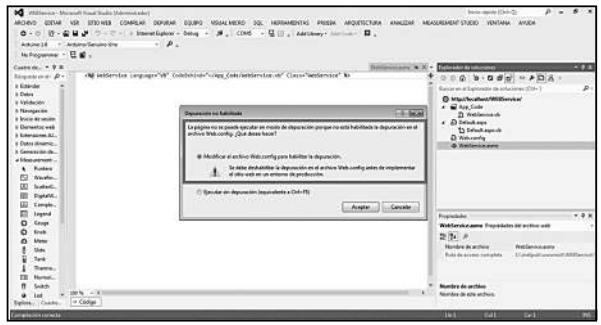


Figura 5.28 Habilitar la depuración.

8) Una vez ejecutado el servicio web hay que hacer una prueba. En la página aparecen los dos nombres de los servicios web creados: control y control 1. Para ejecutar el servicio web, y hacer la prueba, hacemos clic sobre el nombre del servicio web. (Ver figura 5.29.)

Figura 5.29 Realizar la prueba del servicio web.

Las figuras 5.30 y 5.31 muestran los resultados que deben verse en pantalla cuando se invocan los métodos del servicio web creado.

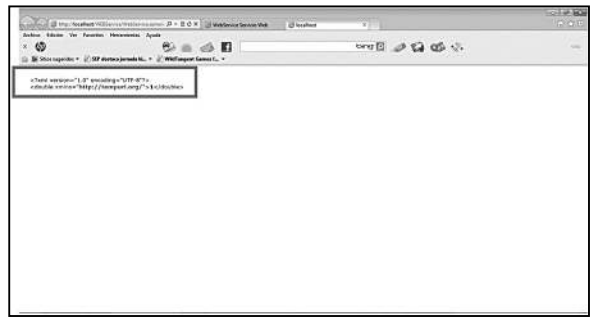


Figura 5.30 Método web 1.

Figura 5.31 Método web 2.

9) Posteriormente, se debe configurar la referencia del servicio web. Para ello, copiar la ruta del servidor: http://localhost/WEBService/WebService.asmx

10) El siguiente paso es crear una aplicación de escritorio que invocará el servicio web que se encuentra en el servidor. Para ello, hay que agregar la referencia de servicio web. Para agregar la referencia del servicio web en la aplicación de escritorio se da clic con el botón derecho sobre el nombre del proyecto y se selecciona Agregar referencia de servicio. Aparecerá una ventana donde se solicita la ubicación del nombre del servicio, es decir, es necesario escribir la ruta donde se encuentra: http://localhost/WEBService/WebService.asmx. (Ver figuras 5.32, 5.33 y 5.34.)

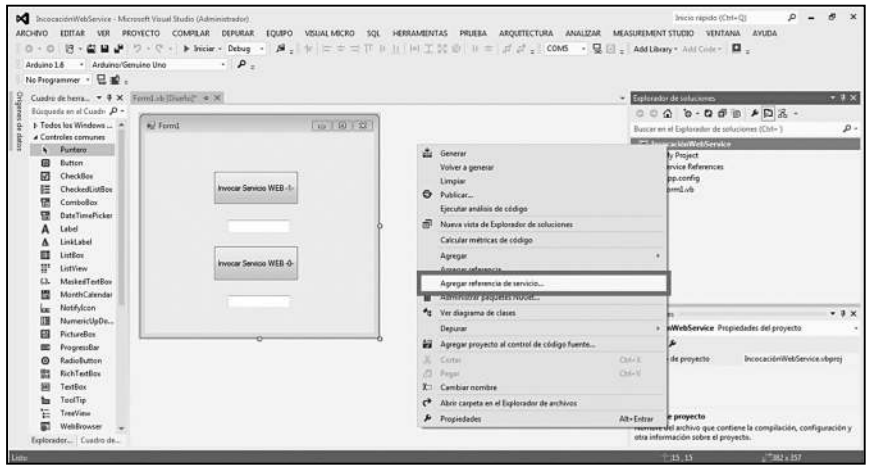


Figura 5.32 Agregar la referencia de servicio web.

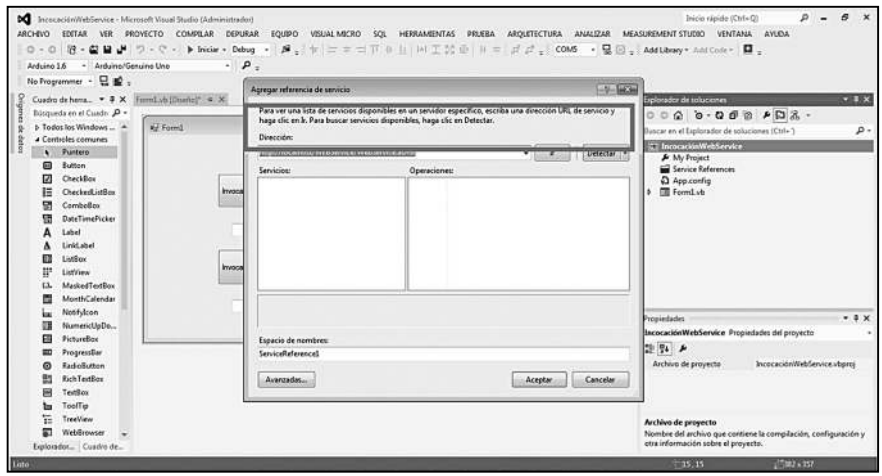


Figura 5.33 Iniciar la detección de la URL del servicio web.

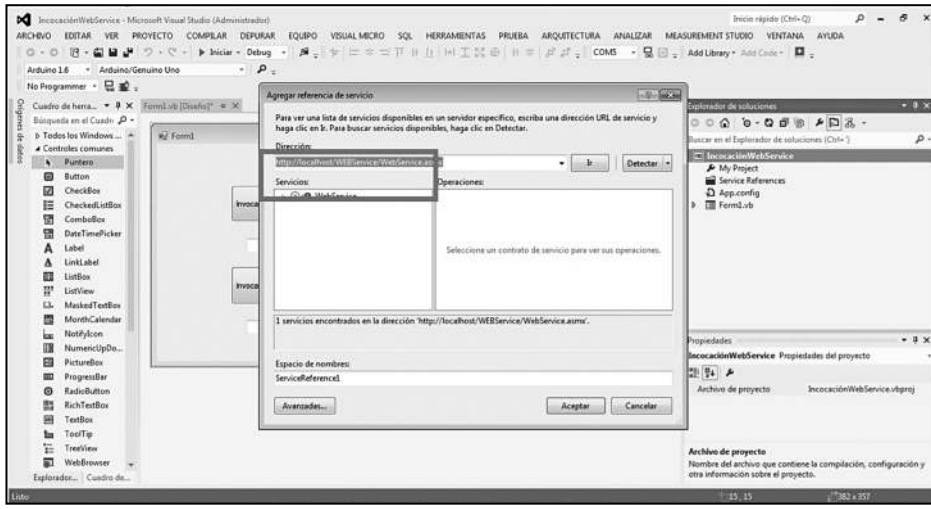


Figura 5.34 Se elige el servicio web que se quiere referenciar.

Una vez que se establece el servicio web, aparecen las funciones de éste. (Ver figura 5.35.)

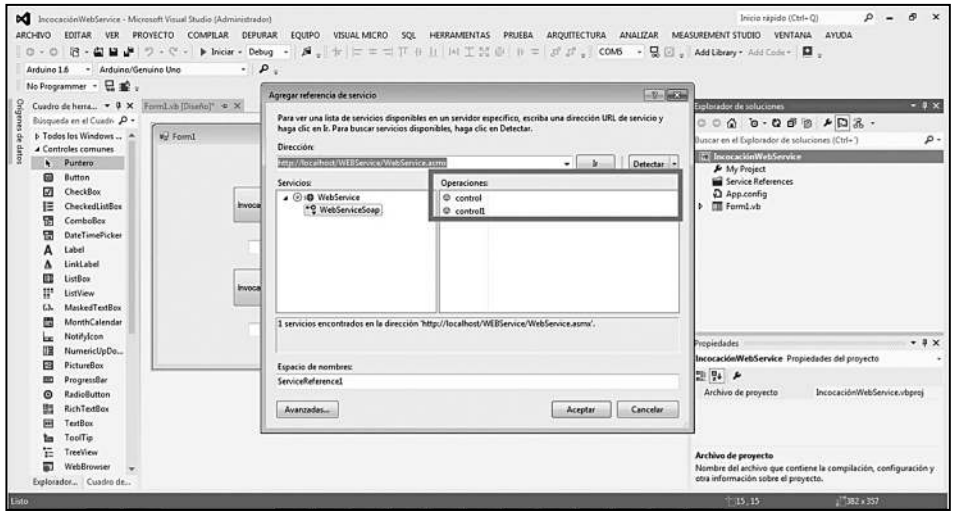


Figura 5.35 Funciones del servicio web.

Después, se debe incluir el nombre de la referencia web creada. Éste debe capturare en donde dice Espacio de nombres. (Ver figura 5.36.)

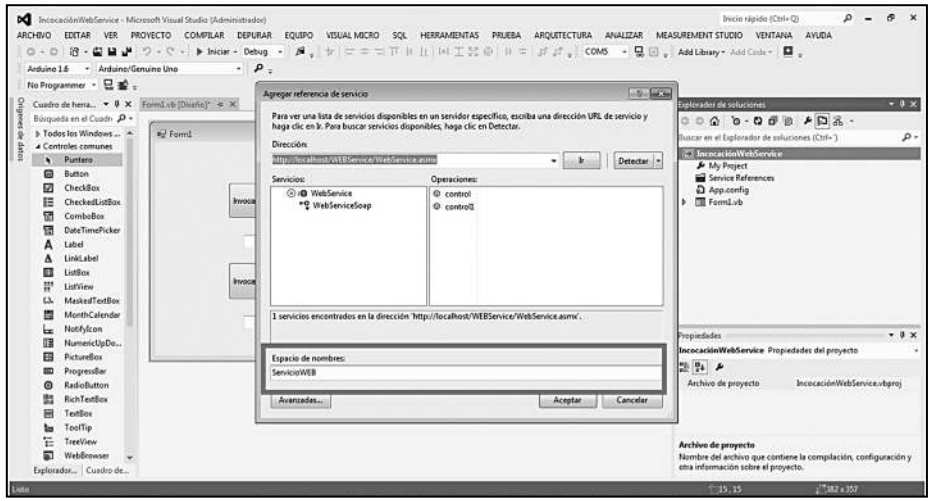


Figura 5.36 Agregar el nombre de la referencia web creada.

La referencia web creada debe aparecer en la aplicación de escritorio. (Ver figura 5.37.)

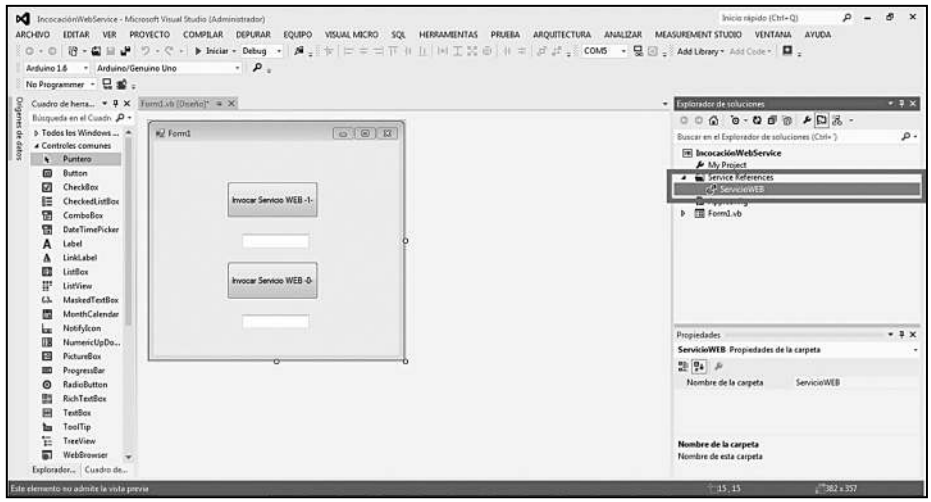


Figura 5.37 Se muestra la referencia web creada en la aplicación de escritorio.

La figura 5.38 muestra cómo debe verse la aplicación de escritorio.

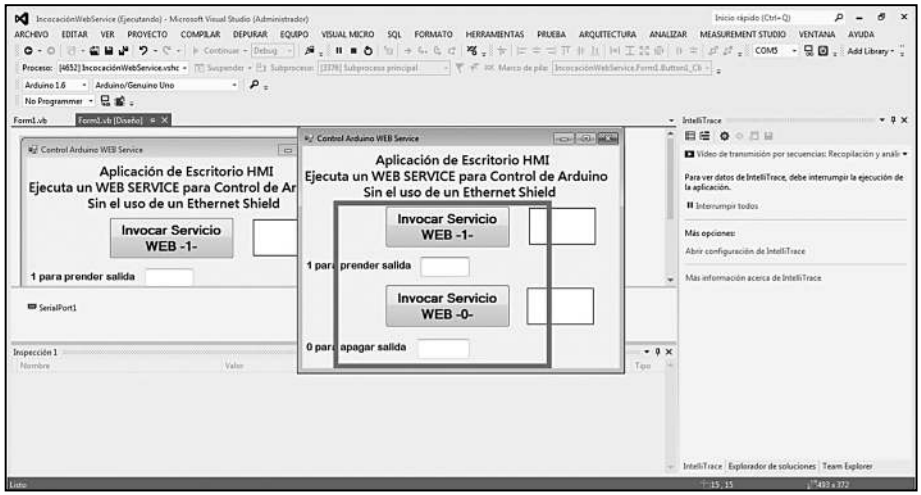


Figura 5.38 Así debe verse la aplicación de escritorio.

De tal forma, cuando se envía el dato "1" para invocar el servicio web, se activa el servicio web 1. (Ver figura 5.39.)

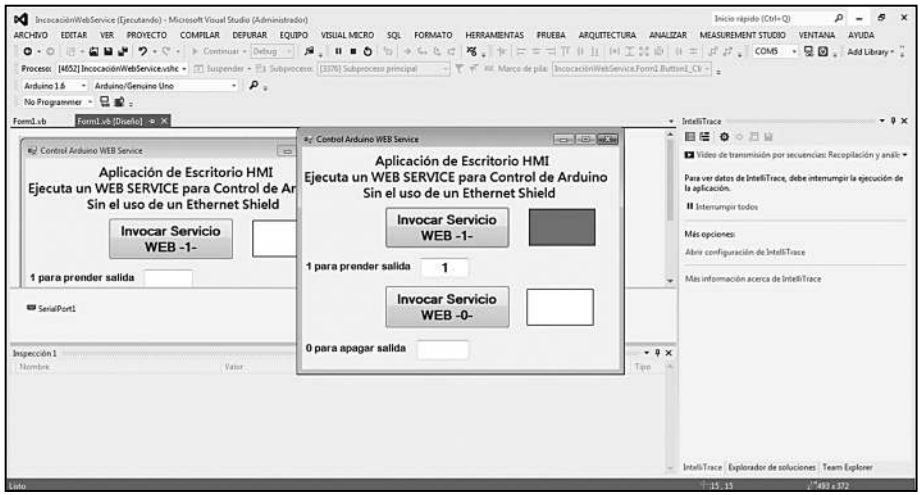


Figura 5.39 Se invoca el servicio del control de encendido.

Cuando se envía el dato "0" para invocar el servicio web, se activa el servicio web 0. (Ver figura 5.40.)

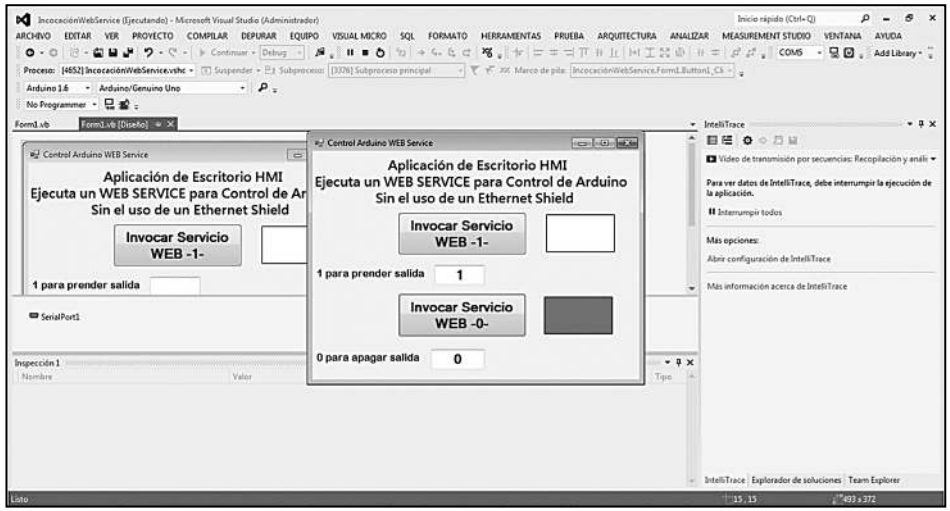


Figura 5.40 Se invoca el servicio del control de apagado.

El código de la aplicación de escritorio que invoca el servicio web debe tener el siguiente formato:

```
Public Class Form1
    Dim x As New ServicioWEB.WebServiceSoapClient

    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim dato As Double = TextBox1.Text
        Dim dato_enviar As Double
        dato_enviar = x.control(dato)
        SerialPort1.Write(dato_enviar)
        RectangleShape1.BackColor = Color.Red
        RectangleShape3.BackColor = Color.White
    End Sub

    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        SerialPort1.Open()
    End Sub

    Private Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Dim dato As Double = TextBox2.Text
        Dim dato_enviar2 As Double
        dato_enviar2 = x.control1(dato)
        SerialPort1.Write(dato_enviar2)
        RectangleShape1.BackColor = Color.White
        RectangleShape3.BackColor = Color.Red
    End Sub
End Class
```

5.7 Resumen

En este capítulo se aprendió a controlar la tarjeta Arduino de forma inalámbrica conectando un módulo de relevadores a través de XBee y de una aplicación HMI. Asimismo, lo interesante es que se logró controlar la Arduino desde una página web utilizando la tecnología ASP.NET y sin utilizar un módulo Ethernet Shield.

Por otro lado, se conectó la Arduino de forma serial al servidor de páginas web directamente, lo cual permitió que desde una página web, en cualquier navegador, se pudieran encender o apagar las luces de forma remota. También se explicó cómo crear una aplicación de un servicio web para ejecutar aplicaciones remotas, es decir, se aprendió a integrar varias tecnologías. Todos estos conocimientos se utilizan para crear aplicaciones remotas que se ejecutan desde cualquier lugar, a través de diferentes clientes.

En el siguiente capítulo se seguirán utilizando las aplicaciones de control para aprender a controlar el movimiento de un motor y su velocidad, así como a controlar un servomotor.

5.8 Problemas

1. Diseñar una interfaz de tipo web con ASP.NET que permita activar cinco focos a través de un temporizador y encenderlos en secuencia.
2. Elabore el control de las salidas a través de la programación de un temporizador programado desde la interfaz de .NET; establezca una página web utilizando Radiobuttons y seleccione el tiempo del temporizador.
3. Realice la activación de secuencia de leds desde una página web.
4. Diseñe el control desde una página web del encendido y apagado de una salida de un foco.
5. Active desde una página web el funcionamiento de un semáforo simple.
6. Realice el ejercicio 5 para un semáforo crucero.
7. Haga la interfaz de una casa domótica diseñada en ASP.NET (diseño de la interfaz y los controles e imágenes) para control de las luces de una casa desde una página web.
8. Realizar la aplicación de escritorio para monitoreo de sensores y activación de salidas.
9. Lleve a cabo un sistema de monitoreo, conecte varios Arduinos y monitoree desde una misma página web varios dispositivos al mismo tiempo.

10. Diseñe una aplicación de escritorio para envío y activación de señales de salida y creación de servicio web en un servidor para monitoreo remoto, utilice sensores como temperatura y humedad, sensores de luz, de presión barométrica y de movimiento.

Capítulo 6

Control de un motor de corriente directa

Objetivos

En este capítulo se utilizarán los conocimientos adquiridos anteriormente sobre aplicaciones de control con el fin de aprender a conectar un motor de corriente directa a la tarjeta Arduino, a través de un puente H, para controlar el giro y la velocidad. También, se explicará cómo llevar a cabo el proceso para realizar la interfaz requerida para el control del dispositivo.

6.1 Introducción

El proyecto de este capítulo es controlar un motor de corriente directa conectado a la tarjeta Arduino y un servomotor. El control de dichos dispositivos se aborda con frecuencia en el área de mecatrónica. Para poder realizar este tipo de control se debe saber cómo controlar:

- El motor a la tarjeta Arduino a través de un circuito puente H (L293D).
- El giro del motor desde la interfaz de software.
- El movimiento de un servomotor.
- Los grados de movimiento del servomotor desde la interfaz HMI.

6.2 Requerimientos de software y hardware

Los dispositivos a utilizar en este proyecto son:

- Tarjeta Arduino UNO.
- Puente H L293D circuito integrado.
- Motor de corriente directa.
- Protoboard.
- Cables macho-macho.
- Servomotor.

El software requerido para Arduino, versión 1.6.5, es el programa Visual Basic .NET 2012; se utilizará la misma interfaz de programación que en capítulos anteriores; esta vez, con el objetivo de implementar los controles para manipular el giro y la velocidad del motor desde una pantalla gráfica.

6.3 Configuración del hardware

Para hacer la interfaz entre la Arduino y el motor se requiere de un circuito integrado que permita controlar el giro del motor y su velocidad, para ello, se utilizará el puente H. Las conexiones de éste se describen en la figura 6.1.

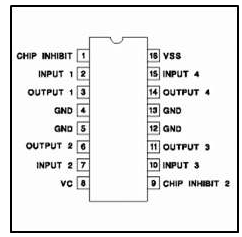


Figura 6.1 Diagrama de conexión del puente H L293D.

Asimismo, el puente H debe conectarse con la tarjeta Arduino de la siguiente manera (tomar como referencia la figura 6.1):

- El pin 1 del puente H se conecta a VCC (+5v).
- El pin 2 del puente H se conecta a la salida digital 7 de la Arduino.
- El pin 7 del puente H se conecta a la salida digital 8 de la Arduino.
- Los pines 3 y 6 del puente H se conectan al motor.
- Los pines 4 y 5 del puente H se conectan a tierra (GND).
- El pin 8 del puente H se conecta a VCC (+5v).
- El pin 16 del puente H se conecta a VCC(+5v).

La imagen de la figura 6.2 muestra cómo debe verse la conexión.

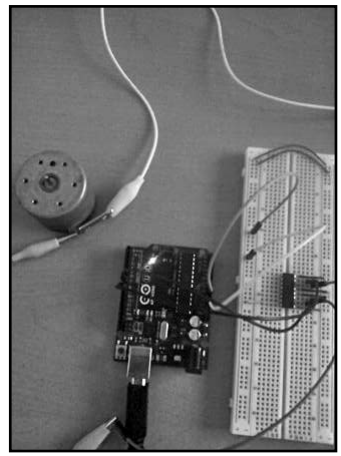


Figura 6.2 Conexión del puente H con la tarjeta Arduino.

En la figura 6.3 se aprecian los elementos que interactuarán en este proyecto.

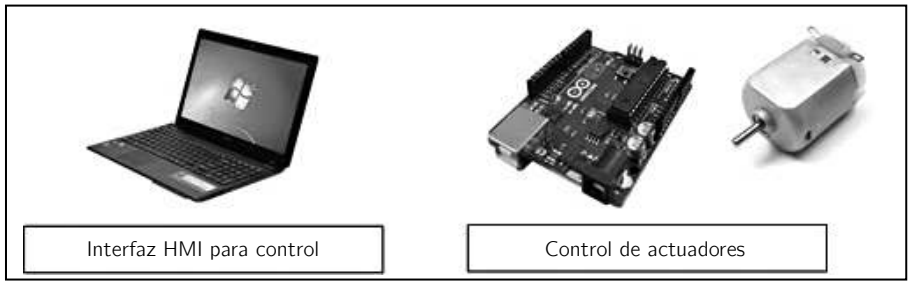


Figura 6.3 Topología de control de actuadores.

6.4 Prueba del motor

Para probar el motor desde la interfaz de comunicación Serial hay que establecer los números de acuerdo con el movimiento que se desee, en este caso, "1" para mover hacia adelante; "2", hacia atrás y "3", para detener el motor.

El formato de instrucción es:

```
int dato;

void setup() {
  Serial.begin(9600);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);

}

void loop() {
  if (Serial.available()>0){
    dato = Serial.read();
```

Para mover el motor hacia adelante:

```
    if (dato == '1'){
      digitalWrite(7,HIGH);
      digitalWrite(8,LOW);
    }
```

Para mover el motor hacia atrás:

```
    if (dato == '2'){
      digitalWrite(7,LOW);
      digitalWrite(8,HIGH);
  }
```

Para detener el motor:

```
     if (dato == '3'){
      digitalWrite(7,LOW);
      digitalWrite(8,LOW);
  }
}
}
```

De tal forma, estos botones aparecen en la pantalla de la interfaz. (Ver figura 6.4.)

Figura 6.4 Pantalla de la interfaz y control.

6.5 Control del giro y la velocidad de un servomotor

En este apartado se explicará cómo controlar el movimiento de un servomotor desde una interfaz de Visual Basic .NET. Es decir, desde una pantalla gráfica en Visual Basic se podrá mover el servomotor de 0 a 180 grados. (Ver figura 6.5.)

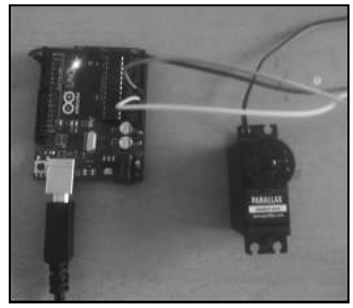


Figura 6.5 El servomotor conectado a la tarjeta Arduino.

En este tipo de proyecto es importante considerar que la placa Arduino acepta caracteres byte por byte desde el puerto serial. Por lo tanto, para que la Arduino pueda recibir más de un dato a la vez, es necesario enviar los datos en forma concatenada para que la tarjeta los pueda interpretar como un solo valor recibido. De tal forma, en Arduino se leerán los datos enviados y se pondrán en un arreglo; después de leerse, se juntarán

todos los datos para recibir valores en un rango de 0 a 180, los cuales son aceptados por la función "servo" para escribir el valor en el pin número 10, que está configurado como una señal PWM (modulación por ancho de pulso). A continuación se muestran los códigos que realiza la tarjeta Arduino al recibir los caracteres:

```
#include <Servo.h>
Servo myservo;
int datos[3] = {0,0,0};
int posicion = 0;

void setup()
{
  Serial.begin(9600);
  myservo.attach(10);
  myservo.write(2);
}

 void loop()
{
    if (Serial.available()){
    for(int i=0; i<4; i++)
    {
      datos[i]= 0;
    }
    int i=0;
    delay(100);
    while (Serial.available() > 0)
    {
      datos[i] = Serial.read() - 48;
      i++;
    }
```

Después se concatenan los valores leídos del puerto serial y se evalúan para que se encuentren dentro del rango entre 0 y 180 grados:

```
    posicion = (100 * datos[0]) + (10 * datos[1]) + da-
tos[2];
     if((posicion >= 0) && (posicion <= 180))
    {
     if(posicion <= 1)
    {
      posicion = 2;
    }
    myservo.write(posicion);
    Serial.println(posicion);
    }
     posicion = 0;
  }
  delay(50);
}
```

6.6 Creación de la pantalla de control

A continuación se explican los pasos y sus códigos para crear la pantalla de control del motor.

Figura 6.6 Pantalla de control posición 0°.

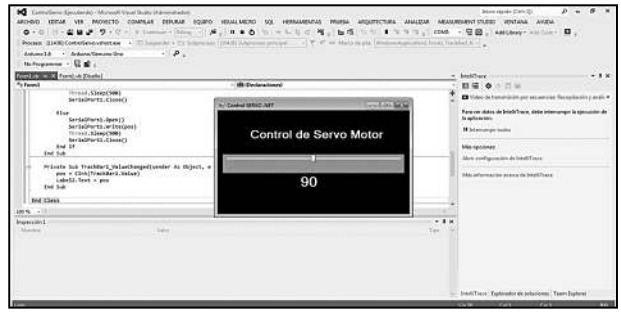


Figura 6.7 Pantalla posición 90°.

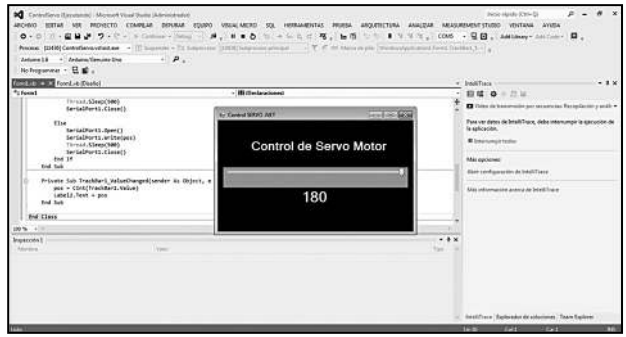


Figura 6.8 Pantalla posición 180°.

1) Declarar las librerías del puerto serial con la siguiente instrucción.

```
Imports System.IO
Imports System.IO.Ports
Imports System.Threading
Public Class Form1
    Dim pos As Integer = 0
```

Es importante declarar el puerto serial que se está ejecutando.

```
    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        SerialPort1.Close()
        SerialPort1.PortName = "COM5"
        SerialPort1.BaudRate = 9600
        SerialPort1.DataBits = 8
        SerialPort1.Parity = Parity.None
        SerialPort1.StopBits = StopBits.One
        SerialPort1.Handshake = Handshake.None
        SerialPort1.Encoding = System.Text.Encoding.De-
fault
    End Sub
```

2) Enviar los datos concatenados. De acuerdo con las condiciones presentadas, se envía el dato por el puerto serial en forma concatenada. Para no saturar el puerto serial y no enviar todos los datos al mismo tiempo, se hace por partes: primero las unidades (cifras de un dígito, entre 0 y 9), luego las decenas (cifras de dos dígitos, entre 10 y 99) y al final las centenas (cifras de tres dígitos, entre 100 y 180), como lo muestran los códigos más abajo.

```
    Private Sub TrackBar1_MouseUp(sender As Object, e As
MouseEventArgs) Handles TrackBar1.MouseUp
```

Instrucciones para especificar las condiciones para enviar datos solamente de 0 a 9 (un dígito):

```
        If pos < 10 Then
            SerialPort1.Open()
            SerialPort1.Write(0)
            SerialPort1.Write(0)
            SerialPort1.Write(pos)
            Thread.Sleep(500)
            SerialPort1.Close()
```

Instrucciones para especificar las condiciones para enviar datos de 10 a 99 (dos dígitos):

```
        ElseIf pos < 100 Then
            SerialPort1.Open()
            SerialPort1.Write(0)
```

```
            SerialPort1.Write(pos)
            Thread.Sleep(500)
            SerialPort1.Close()
```

Instrucciones para especificar las condiciones para enviar datos de 100 a 180 (tres dígitos):

```
        Else
            SerialPort1.Open()
            SerialPort1.Write(pos)
            Thread.Sleep(500)
            SerialPort1.Close()
        End If
    End Sub
```

Cuando cambia el valor se muestra lo siguiente en la pantalla:

```
    Private Sub TrackBar1_ValueChanged(sender As Object,
e As EventArgs) Handles TrackBar1.ValueChanged
        pos = CInt(TrackBar1.Value)
        Label2.Text = pos
    End Sub
End Class
```

6.7 Resumen

En este proyecto se aprendió a controlar un motor de corriente directa desde una interfaz gráfica. En proyectos similares, se podrían adaptar otros controles para manipular la velocidad del motor o hacer el control de forma automática o de forma manual.

Un aspecto importante en este proyecto fue la manipulación del movimiento del servomotor para moverlo desde la interfaz gráfica a través de un control de movimiento. También se pueden añadir otros controles para dirigir más servomotores; por ejemplo, en un brazo robótico y desde la interfaz realizar el control del dispositivo. Además se podría capturar el valor en grados en la caja de texto para realizar el movimiento por medio de otros controles, como TextBox u otros.

En los siguientes proyectos habrá nuevas oportunidades para aplicar este tipo de controles de forma inalámbrica.

6.8 Problemas

1. Realice una interfaz que permita controlar el giro de un motor de corriente directa, movimiento por tiempos; además, programe los temporizadores en el código de Arduino.
2. Diseñe una interfaz de control para controlar un motor de corriente directa por tiempos desde (se puede utilizar un comboox para seleccionar los tiempos de activación).
3. Monitorear el consumo de corriente con sensor de corriente y controlar el giro del motor.
4. Programe el control de una lavadora industrial: movimientos, ciclos de lavado, tiempos; se pueden seleccionar varios programas desde la misma interfaz.
5. Controlar un elevador digital con sensores interruptores de límite; control y monitoreo desde la misma interfaz.
6. Diseñe una interfaz de control de apertura y cierre de una puerta.
7. Lleve a cabo el control de la apertura de una ventana y de su velocidad y giro.
8. Realice una interfaz de control de un termómetro digital con sensor de temperatura y servomotor.
9. Realizar interfaz de control para servomotor con botones; cada botón mueve el servo cierta cantidad de grados.
10. Haga una interfaz de control de motor a pasos de tipo manual desde la que se lleve el monitoreo y el control.
11. Elabore una interfaz de control de un servomotor y teclee en una caja de texto la cantidad de grados.
12. Diseñe un sistema de control de cinco servomotores para prototipo robótico.

Capítulo 7

Sistema de alarma inalámbrica

Objetivos

En este capítulo se explicará cómo activar una alarma mediante una comunicación inalámbrica. Asimismo, se realizará el diseño de la interfaz web para monitorear el sistema de la alarma.

7.1 Introducción

En capítulos anteriores se explicó cómo establecer la comunicación serial, así como algunas aplicaciones de esta manera de transmisión. En este capítulo se aprenderá a realizar un sistema de alarma completamente inalámbrico (módulos y comunicación serial). Para ello, en este proyecto se explicará cómo:

- Conectar de los módulos inalámbricos transmisor-receptor.
- Diseñar una interfaz con la tarjeta Arduino para el envío y recepción de datos.
- Probar el sensor de movimiento PIR en la Arduino y en el módulo transmisor.
- Diseñar la interfaz del receptor y cómo leer la señal.
- Establecer una comunicación entre los dos módulos y probar el mensaje enviado.
- Diseñar la interfaz HMI para activar la alarma desde Visual Basic .NET.

7.2 Requerimientos de software y hardware

Con respecto al hardware, se requiere de lo siguiente:

- Dos tarjetas Arduino UNO.
- Un módulo transmisor de 433 Mhz.
- Un módulo receptor de 433 Mhz.
- Un sensor de movimiento PIR.
- Una bocina (buzzer).
- Un protoboard.
- Cables macho-macho.

En cuanto al software, en este proyecto se utilizará la librería virtual Wire, la cual se puede descargar del siguiente vínculo: http://www.airspayce.com/mikem/arduino/VirtualWire/

7.3 Configuración del hardware

Cada módulo (transmisor y receptor) se debe de conectar a una tarjeta Arduino UNO, como se aprecia en la figura 7.1.

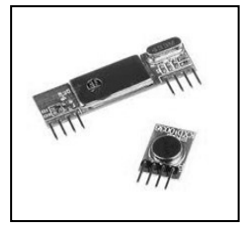


Figura 7.1 Conexión del módulo receptor conectado a la tarjeta Arduino.

Por otro lado, se debe conectar el transmisor con el sensor de movimiento PIR. (Ver figura 7.2.)

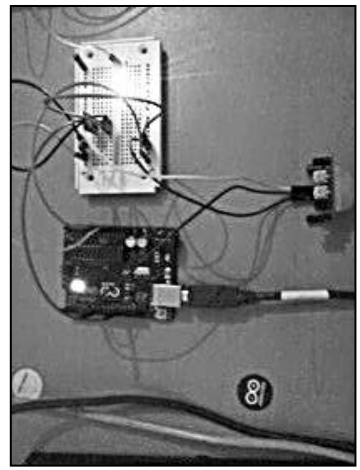


Figura 7.2 Conexión del transmisor con el sensor de movimiento PIR.

También, conectar el receptor con el módulo y la bocina, como se aprecia en la figura 7.3.

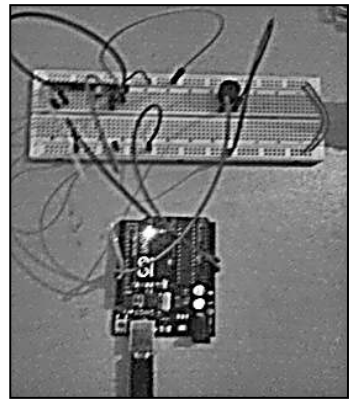


Figura 7.3 Conexiones de la tarjeta Arduino con la bocina.

7.3.1 Conexiones de los módulos transmisor y receptor

A continuación se explican las conexiones de los módulos transmisor y receptor. (Ver figura 7.4.)

Módulo transmisor:

- El pin 1 se conecta a tierra (GND).
- El pin 2 (el pin de datos) se conecta al pin 12 de la Arduino.

- El pin 3 se conecta a +5V.
- El pin de datos del sensor de movimiento se conecta al pin 7 de la Arduino.
- El pin 4 es el cable de la antena.

Módulo receptor:

- El pin 1 se conecta a tierra (GND).
- El pin 2 (el pin de datos) se conecta al pin 11 de la Arduino.
- El pin 3 no se conecta.
- Los pines 4 y 5 se conectan a +5V.
- Los pines 6 y 7 se conectan a tierra (GND).
- El pin 8 es el cable de la antena.
- La bocina se conecta al pin 10 de la Arduino.

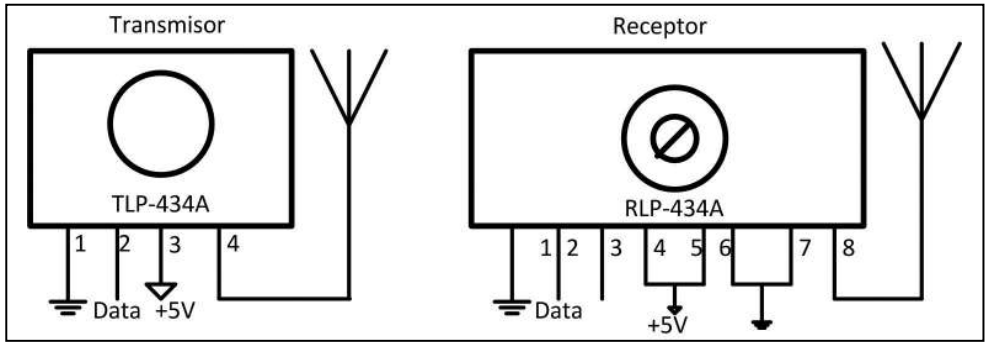


Figura 7.4 Diagrama de conexión de los módulos transmisor y receptor.

En la figura 7.5 se puede apreciar la interacción de los elementos en este proyecto.

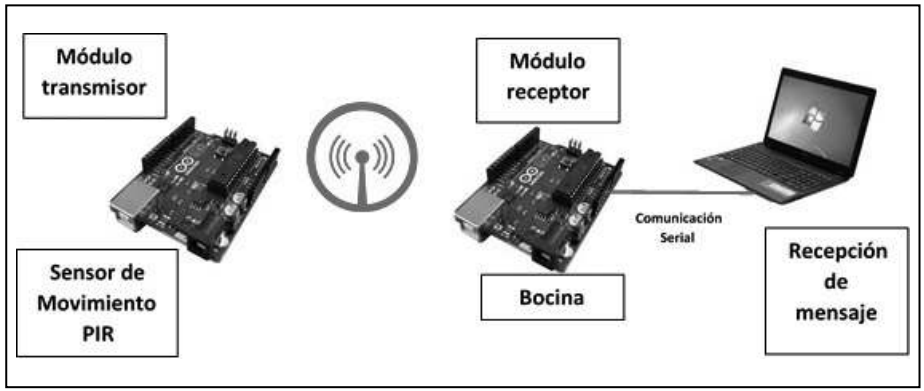


Figura 7.5 Topología de la aplicación en este proyecto.

7.4 Comunicación serial inalámbrica

Para establecer la comunicación serial se debe utilizar la librería Virtual Wire (#include <VirtualWire.h>), que se debe cargar al transmisor.

Por otro lado, deben definirse los pines de envío y recepción con las siguientes instrucciones:

```
const int led_pin = 13;
const int transmit_pin = 12;
const int sensor_pin = 7;

int sensor_value;
void setup()
{
vw_set_tx_pin(transmit_pin);
vw_setup(2000);

pinMode(led_pin, OUTPUT);
pinMode(sensor_pin,INPUT);
}

void loop()
{
sensor_value = digitalRead(sensor_pin);
```

Si no se detecta presencia, se debe declarar un arreglo carácter por carácter, siguiendo el siguiente formato:

```
char msg[3] = {'o','f','f'};
```

Para enviar el mensaje de que hay presencia, se debe utilizar el control de condición (if) con el siguiente formato:

```
if (sensor_value == 1){
msg[0] = 'o';
msg[1] = 'n';
msg[2] = '#';
}
digitalWrite(led_pin, HIGH);
```

La instrucción para enviar mensaje es:

```
vw_send((uint8_t *)msg, 3);


vw_wait_tx();
digitalWrite(led_pin, LOW);
delay(1000);
}
```

En cuanto al receptor, se debe cargar la misma librería (#include <VirtualWire.h>) con el siguiente formato:

```
const int led_pin = 13;
Las instrucciones para los pines de envío y recepción de
datos son:
const int transmit_pin = 12;
const int receive_pin = 11;
int bocina = 10;
int dato;
void setup()
{
pinMode(bocina,OUTPUT);
delay(1000);
Serial.begin(9600);
vw_set_rx_pin(receive_pin);
vw_setup(2000);
```

La instrucción para inicializar la comunicación es:

```
vw_rx_start();
pinMode(led_pin, OUTPUT);
}

void loop()
{
```

Las instrucciones para establecer los arreglos para el envío del mensaje y el tamaño de éste son:

```
uint8_t buf[VW_MAX_MESSAGE_LEN];
uint8_t buflen = VW_MAX_MESSAGE_LEN;
Las instrucciones para verificar que el mensaje llegó co-
rrectamente son:
if (vw_get_message(buf, &buflen))
{
```

Las instrucciones para encender el led y así indicar que el mensaje llegó correctamente son:

```
digitalWrite(led_pin, HIGH);
```

Es necesario imprimir el mensaje por el puerto serial con el siguiente formato de instrucción:

```
for (int i = 0; i < buflen; i++)
{
Serial.print(char(buf[i]));
}
Serial.println();
```

```
digitalWrite(led_pin, LOW);
}
```

También hay que establecer las condiciones para activar el buzzer desde Visual Basic. Para ello, se utiliza este formato:

```
if (Serial.available()>0){
   dato = Serial.read();

   if (dato == '1'){
    digitalWrite(bocina,HIGH);
    delay(3000);
   }
  if (dato == '0'){
    digitalWrite(bocina,LOW);
   }
}
}
```

7.5 Prueba de los módulos de comunicación transmisor-receptor

A continuación se describirán los resultados que deben surgir después de descargar ambos programas. Es importante considerar que el módulo transmisor se puede conectar a un eliminador de 9 a 12 voltios, para que la Arduino se quede trabajando de forma independiente. Por otro lado, el módulo receptor debe estar conectado a la computadora para poder probar el envío y recepción del mensaje y para poder ver los caracteres enviados en la pantalla del monitor serial.

La figura 7.6 muestra cómo se ve la pantalla cuando el receptor recibe un cero; es decir, no detecta presencia.

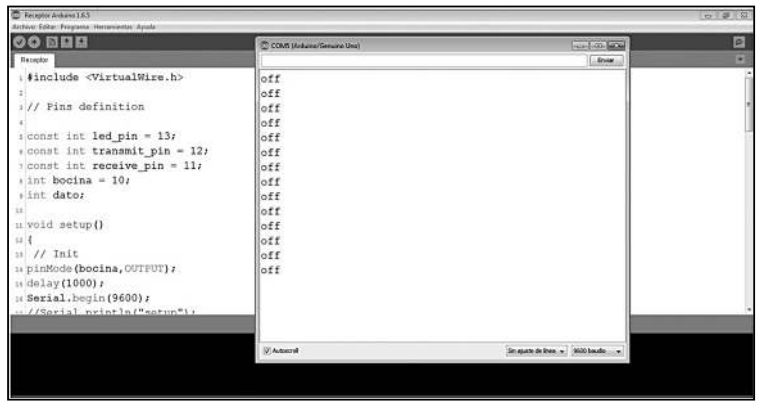


Figura 7.6 Pantalla cuando no hay detección de presencia.

La figura 7.7 muestra cómo se ve la pantalla cuando el receptor recibe un uno; es decir, sí detecta presencia.

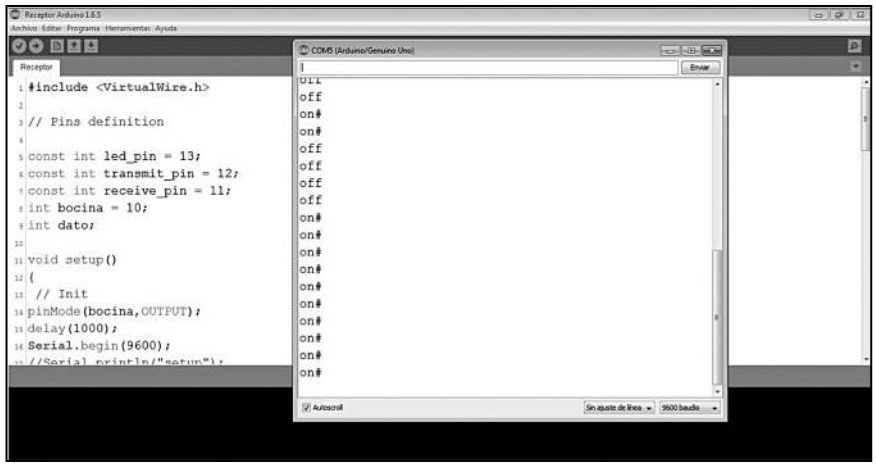


Figura 7.7 Pantalla cuando hay detección de presencia.

7.6 Interfaz gráfica de monitoreo

Cuando no se detecte presencia, el sensor enviará los caracteres "off", que se podrán apreciar en pantalla, como lo muestra la figura 7.8.

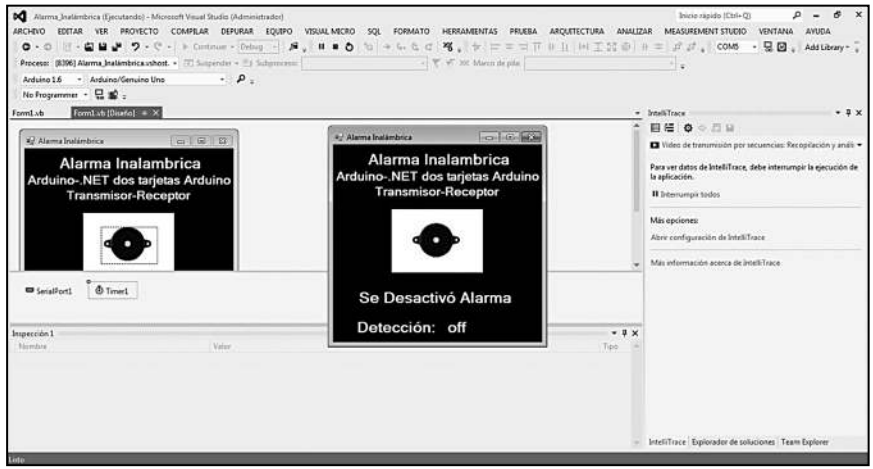


Figura 7.8 El sensor envía los caracteres "off".

Cuando el sensor detecta presencia, envía los caracteres "on", que se podrán apreciar en pantalla, como lo muestra la figura 7.9.

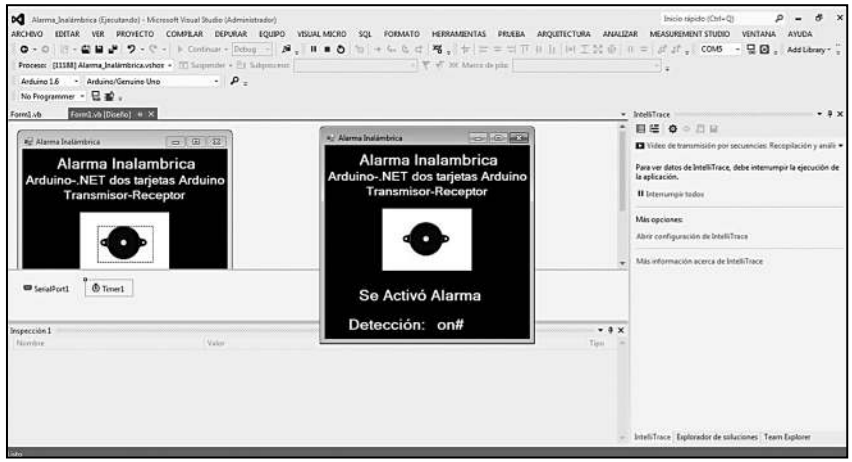


Figura 7.9 El sensor envía los caracteres "on".

Las instrucciones son las siguientes:

```
Public Class Form1

    Dim cadena As String

    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        SerialPort1.Open()
        Timer1.Enabled = True

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As Even-
tArgs) Handles Timer1.Tick
        Dim cadena1 As String
        cadena = SerialPort1.ReadExisting()
        cadena1 = Mid(cadena, 1, 3)
        Label1.Text = cadena1.Trim
```

Las instrucciones para encender la bocina, con lo cual se indica que se ha detectado movimiento, son las siguientes:

```
        If cadena1 = "on#" Then
            SerialPort1.Write("1")
            Label4.Text = "  Se Activó Alarma"
        Else
            SerialPort1.Write("0")
            Label4.Text = "Se Desactivó Alarma"
        End If

    End Sub
End Class
```

7.7 Resumen

En este capítulo se describió cómo enviar datos de forma inalámbrica utilizando módulos transmisor-receptor, cada uno conectado a una tarjeta Arduino. A este proyecto se podría agregar un módulo GSM para enviar un mensaje de texto cuando detecte presencia por medio del sensor. Los pasos para ello se describirán en los próximos capítulos. A este proyecto también se le podría añadir un sensor de temperatura en el transmisor, con el fin de leer la temperatura en el receptor y mostrarla en una pantalla LCD o en la misma interfaz HMI. Además, puesto que en capítulos anteriores se aprendió a realizar una interfaz en una página web, también se podría aplicar en este proyecto para monitorear el estado de la alarma.

En el siguiente capítulo se describirán los pasos para realizar una estación de registro de datos para guardar las lecturas de los sensores en una base de datos y mostrarlos en pantalla a través de un reporte.

7.8 Problemas

1. Realizar un timbre inalámbrico, conectar un botón pulsador para activar.
2. Lleve a cabo una estación de monitoreo de sensores de movimiento.
3. Elabore un monitoreo inalámbrico de temperatura y humedad.
4. Haga un monitoreo inalámbrico de corriente.
5. Diseñar una activación de secuencia de luces de forma inalámbrica.
6. Elabore una apertura de chapa de forma inalámbrica.
7. Llevar a cabo un sistema de monitoreo de varios sensores de movimiento en cada cuarto.
8. Diseñe una casa inteligente con control inalámbrico de puertas y ventanas, activación de luces, sensores inalámbricos y timbre inalámbrico.

Capítulo 8

Estación de registro de datos

Objetivos

En este capítulo se realizará una estación de registro de datos, en la cual se integrarán los conocimientos de los proyectos realizados previamente. Además, se aprenderá a realizar un registro de datos de forma local, un registro en una base de datos (datalogger) y un reporte en Excel para consulta.

8.1 Introducción

En el capítulo 3 se describió cómo realizar una aplicación de una estación meteorológica a través de un sensor de temperatura, humedad y el sensor de luz. Estos datos se mostraban en una pantalla de interfaz como pantalla de monitoreo. Actualmente, en los procesos industriales, resulta de vital importancia que las variables que deben monitorearse se registren de forma digital. Esto se hace con el fin de tener almacenado el momento en el que ocurrió un suceso, se activó una alarma dentro del proceso o si un valor registrado sobrepasa un límite programado. Debido a esto, en este capítulo se aprenderá a realizar un software para registrar los datos, primeramente de manera local, guardando las lecturas de los sensores en una memoria Micro SD (ver figura 8.1) y luego en una base de datos..

Figura 8.1 Memoria Micro SD para guardar las lecturas de los sensores.

En este proyecto se describirá cómo:

- Leer los sensores.
- Mostrar información en pantalla.
- Abrir y crear un archivo en la memoria SD.
- Guardar los datos de los sensores.

Para insertar los datos leídos en una base de datos se realizará el mismo proceso, pero además se aprenderá a:

- Crear una base de datos en Microsoft SQL server 2012.
- Crear tabla con los campos Id, temperatura, humedad y fecha/hora.
- Conectarse con la base de datos.
- Insertar datos.
- Cerrar conexión.
- Mostrar datos en pantalla.

8.2 Requerimientos de software y hardware

Para este proyecto se utilizará el siguiente software:

- Entorno de trabajo de Arduino IDE.
- Microsoft Visual Basic .NET 2012.
- Manejador de base de datos Microsoft SQL Server 2012.
- Máquina virtual opcional para cargar servidor de base de datos.
- Librerías DHT.H y SPI.h.
- Librería de Excel para comunicación con .NET.
- Tarjeta Arduino UNO.
- Módulo Ethernet Shield con entrada para Micro SD.
- Memoria Micro SD y adaptador para memorias.

8.3 Configuración del hardware

En este proyecto, el sensor DHT11 se conecta al pin número 7 (GND), la tierra de la tarjeta Arduino UNO y VCC al pin de 5 volts. (Ver figura 8.2.)

Figura 8.2 Conexión de la tarjeta Arduino con el sensor DHT11.

8.4 Guardar los datos localmente mediante el módulo SD

Para la primera parte del proyecto; es decir, guardar de forma local los datos en una memoria SD, se realizará un sketch en el IDE de Arduino de la siguiente manera:

1) Se declaran las librerías con el siguiente formato:

```
#include "DHT.h"
#include <SD.h>
#include <SPI.h>
```

2) Se declaran los pines del sensor:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

3) Se define el pin de selección de la memoria:

```
const int chipSelect = 4;
DHT dht(DHTPIN, DHTTYPE);
void setup() {
     Serial.begin(9600);
     Serial.println("Inicializando Tarjeta SD...");
```

4) Se verifica que la memoria se haya detectado:

```
  if (!SD.begin(chipSelect)) {
    Serial.println("Tarjeta no presente");
     return;
  }
  Serial.println("Memoria SD inicializada.");

   dht.begin();

 }
void loop() {

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  String temp = String((int) t);
  String hum = String((int) h);
```

5) Se concatena el valor de la temperatura y la humedad:

```
  String valor = temp + "," + hum;
```

6) Se crea el archivo (si no existe):

```
File dataFile = SD.open("data.txt", FILE_WRITE);
```

7) Se escribe el dato en el archivo, se cierra el archivo y se muestra el valor guardado en el monitor serial:

```
if (dataFile) {
   dataFile.println(valor);
  dataFile.close();
   Serial.println(valor);
}
else {
  Serial.println("error para abrir data.txt");
}

Cada segundo se repite el proceso de guardado de datos:
delay(1000);
}
```

En la pantalla, los valores guardados de la temperatura y humedad se muestran como en la figura 8.3:

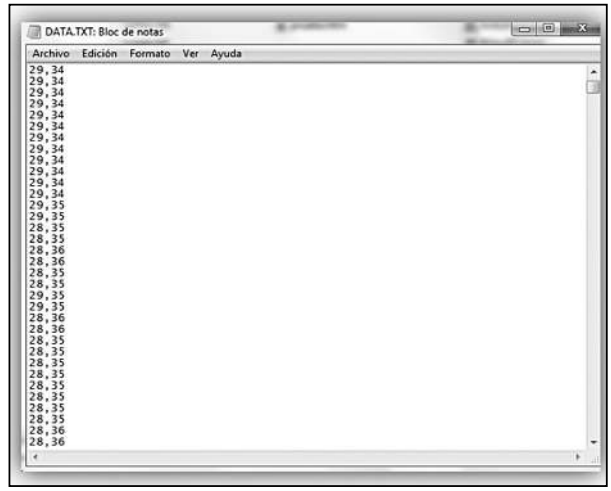


Figura 8.3 Valores de temperatura y humedad en pantalla.

8.5 Servidor de la base de datos

En este proyecto se utilizará el manejador de base de datos Microsoft SQL Server 2012, cargado en una máquina virtual para la comunicación con la aplicación de Visual Basic .NET.

Los pasos para acceder al servidor y crear la base de datos son los siguientes:

1) Acceder a Microsoft SQL Server 2012. (Ver figura 8.4.)

Figura 8.4 Entrando al servidor.

2) Acceder a la pantalla principal y hacer clic en el comando Conectar. (Ver figura 8.5.)

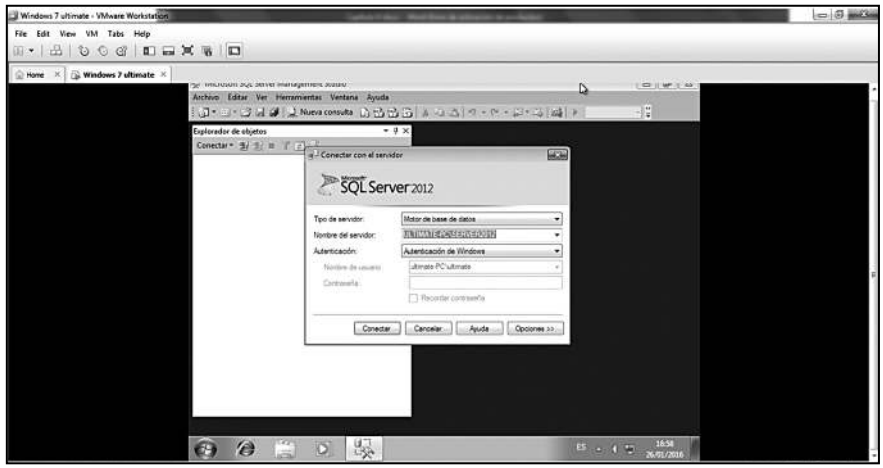


Figura 8.5 Conectando con el servidor.

3) Elegir la aplicación de base de datos y su servidor. (Ver figura 8.6.)

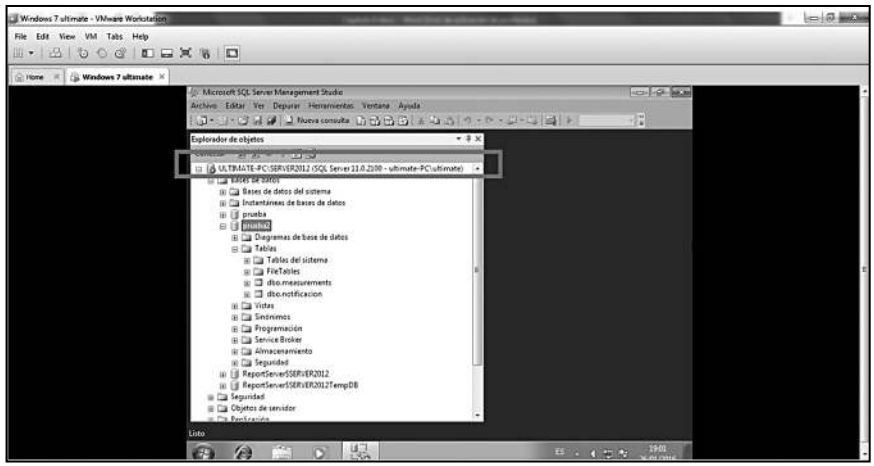


Figura 8.6 Entrando al servidor de la base de datos.

4) Elegir una tabla creada de la aplicación; en este caso, dbo.measurements. (Ver figura 8.7.)

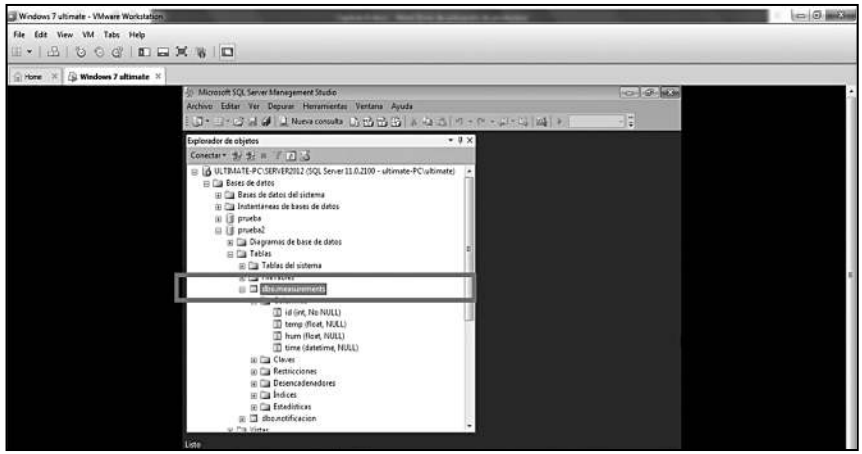


Figura 8.7 Creando una tabla desde la aplicación.

5) Diseñar la tabla "measurements", donde se guardarán los datos. (Ver figura 8.8.)

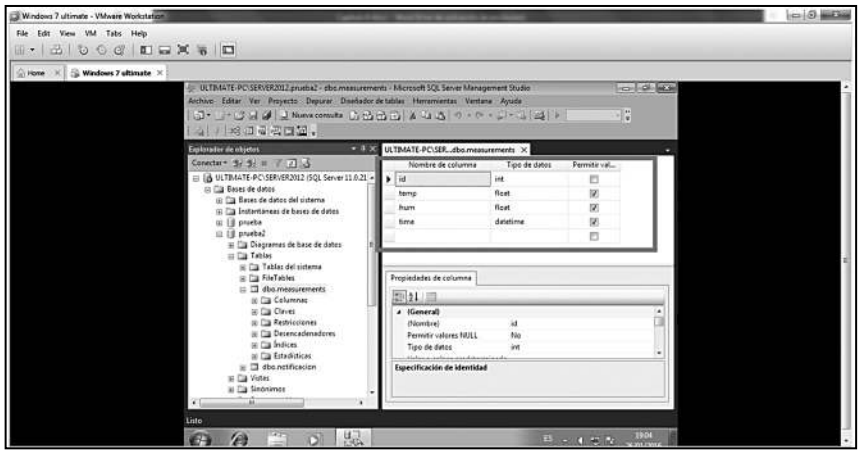


Figura 8.8 Diseñando la tabla con los campos que se quieren registrar.

6) Configurar la propiedad de la columna "id" como "int" y autoincrementable en 1; de tal forma, la lectura se guardará de forma consecutiva. (Ver figura 8.9.)

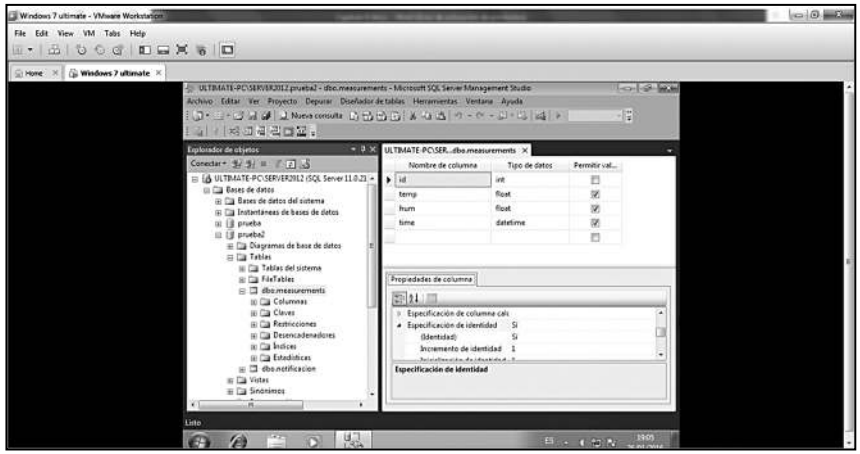


Figura 8.9 Configurando las propiedades de la columna "id".

7) Configurar la propiedad de la columna Time para que muestre el valor actual de la fecha y la hora; de tal forma, quedarán registradas en el momento de dicha lectura. (Ver figura 8.10.)

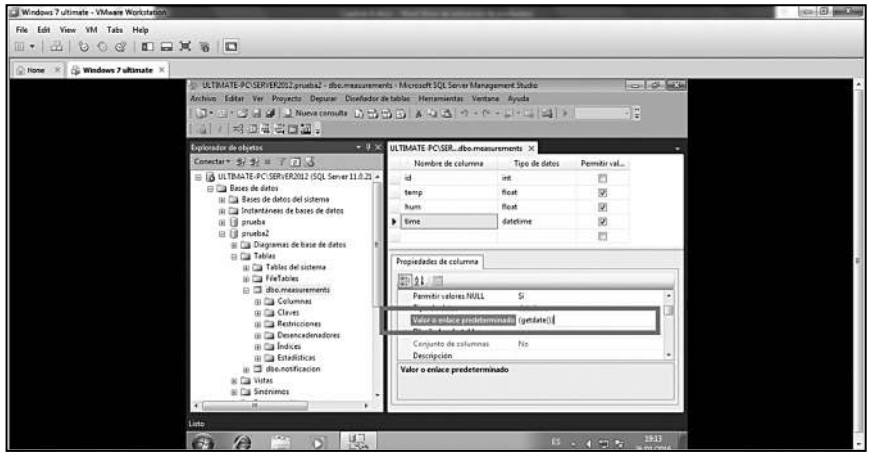


Figura 8.10 Configurando las propiedades de la columna Time.

8.6 Inserción de los datos desde la aplicación

Para insertar los datos desde la aplicación, primeramente se debe realizar la adquisición de los datos y leer los valores según el siguiente formato:

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles timerlectura.Tick
        Dim cadena1 As String
        Dim cadena2 As String
        Dim cadena3 As String
        cadena = SerialPort1.ReadExisting()
        cadena1 = Mid(cadena, 1, 5)
        cadena2 = Mid(cadena, 7, 5)
        cadena3 = Mid(cadena, 13, 5)
        txttemp.Text = cadena1.Trim
        txthum.Text = cadena2.Trim
    End Sub
```

Después, se insertan los datos al ejecutar, cada cierto tiempo, esta función:

```
Private Sub timerinsertardatos_Tick(sender As Object, e
As EventArgs) Handles timerinsertardatos.Tick

        Dim temp As Double = Val(txttemp.Text)
        Dim hum As Double = Val(txthum.Text)

        If Insertar(temp, hum) Then
            lblmensajeguardado.Text = "Lectura guarda-
da..."
        Else
```

```
            MsgBox("Los datos no se guardaron", MsgBoxS-
tyle.Exclamation, "Error de lectura")
        End If
    End Sub
```

Para insertar los datos se utiliza la siguiente función:

```
Public Function Insertar(ByVal temp As Double, ByVal hum
As Double) As Boolean

        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"
        Dim consulta As New SqlCommand("INSERT INTO me-
asurements(temp,hum) VALUES('" & temp & "','" & hum &
"')", conexion)

        Try
            conexion.Open()
            consulta.ExecuteNonQuery()
            conexion.Close()
            Return True

        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Exclamation)
            conexion.Close()
            Return False
        End Try
    End Function
```

Debe salir el siguiente mensaje, el cual corrobora que los datos fueron insertados:

```
Private Sub Timer1_Tick_1(sender As Object, e As Even-
tArgs) Handles Timer1.Tick
        lblmensajeguardado.Text = ""
    End Sub
```

8.6.1 Control para insertar los datos

Para insertar los datos adquiridos en la pantalla se registró un control checkbox. Al estar activado, éste permite que los datos se inserten en la base de datos. Si no se encuentra activado, no se inserta ningún registro.

Para ello, hay que describir la función que realice dicha acción:

```
Private Sub CheckBox1_CheckedChanged(sender As Object, e
As EventArgs) Handles chkguardarDatos.CheckedChanged
        If chkguardarDatos.Checked = True Then
            timerinsertardatos.Enabled = True
            Timer1.Enabled = True
        Else
```

```
            timerinsertardatos.Enabled = False
            lblmensajeguardado.Text = ""
            Timer1.Enabled = False
        End If
    End Sub
```

La figura 8.11 muestra la pantalla de la aplicación que le indica al usuario dónde insertar los datos.

Figura 8.11 Pantalla de interacción con el usuario para ir construyendo la base de datos.

En la figura 8.12 se puede apreciar cómo se han incluido los datos.

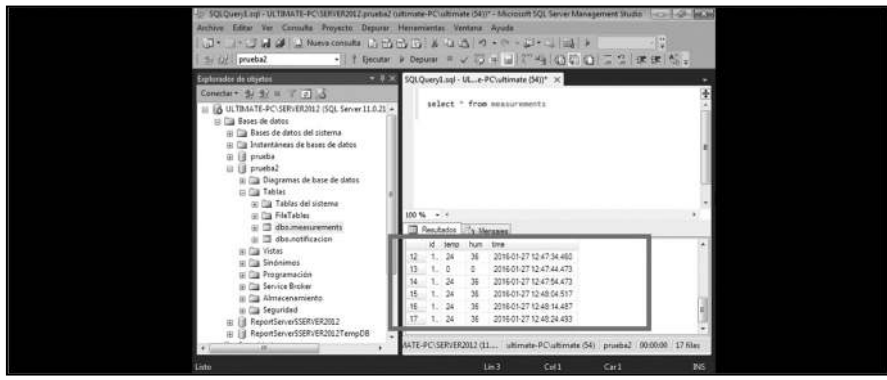


Figura 8.12 Vista previa de la base de datos.

8.7 Pantalla de registro de datos

Para que los registros queden en la base de datos, primeramente se debe crear un proyecto nuevo en Visual Basic .NET 2012 de tipo Escritorio. Ahí se mostrarán los valores adquiridos de los sensores de temperatura y humedad conectados a la tarjeta Arduino.

Para mostrar los datos en el control Datagridview hay que definir una función al inicio de la forma, con ello, cuando se cargue el formulario, se mostrará la información que está almacenada. El formato de instrucción es el siguiente:

```
Dim dato As New DataSet
        dato = Extraerlecturas()
        data.DataSource = dato.Tables("measurements")
```

A continuación se debe presentar el código referente a la consulta cuando se haga clic en el botón de consulta. El formato es el siguiente:

```
Private Sub Button1_Click(sender As Object, e As Even-
tArgs) Handles Button1.Click
        Dim dato As New DataSet
        dato = Extraerlecturas()
        data.DataSource = dato.Tables("measurements")
    End Sub
```

Para poder extraer los datos, se utiliza la siguiente función:

```
Public Function Extraerlecturas() As DataSet

        Dim conjunto As New DataSet
        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"

        Dim consulta As New SqlDataAdapter("select * from
measurements", conexion)
        Try
            conexion.Open()
            consulta.Fill(conjunto, "measurements")
            conexion.Close()
            Return conjunto
        Catch ex As Exception
            conexion.Close()
            Return conjunto
        End Try

    End Function
```

8.7.1 Mostrando los datos insertados

Para mostrar los datos inmediatamente después de que se inserten, hay que hacer una combinación de las funciones de código que se han estudiado. De esta forma, el usuario podrá observar el último registro que se guardó en la base de datos y, de hecho, podrá identificarlo al momento. Más adelante se enlistan los pasos a seguir.

1) El procedimiento empieza agregando unas líneas de código al timerinsertardatos, para que se ejecute en el momento en que el usuario se lo indique a través de la activación del control Guardar datos. La instrucción es:

```
Private Sub timerinsertardatos_Tick(sender As Object, e
As EventArgs) Handles timerinsertardatos.Tick
```

2) Para insertar los datos se utiliza el siguiente formato:

```
        Dim temp As Double = Val(txttemp.Text)
        Dim hum As Double = Val(txthum.Text)

        If Insertar(temp, hum) Then
            lblmensajeguardado.Text = "Lectura guarda-
da..."
        Else
            MsgBox("Los datos no se guardaron", MsgBoxS-
tyle.Exclamation, "Error de lectura")
        End If
```

3) Luego, inmediatamente después de que se insertaron los valores, se extraen los datos para mostrarlos en el datagridview:

```
        Dim dato As New DataSet
        dato = Extraerlecturas()
        data.DataSource = dato.Tables("measurements")
```

4) Esta línea permite que la tabla se posicione en el último registro guardado:

```
data.FirstDisplayedScrollingRowIndex = data.Rows.Count -
1


End Sub
```

La figura 8.13 muestra cómo se ve en pantalla el registro de los datos.

Figura 8.13 Registro de los datos en pantalla.

8.8 Envío de los datos a Excel para graficar los valores registrados

Una vez que se tiene registrada la información en la base de datos, es muy importante generar un reporte según los datos que genere la aplicación. Para ello se debe crear un archivo en formato de Excel; así pueden utilizarse en la posteridad.

A continuación se describen los pasos para exportar los datos registrados a Excel:

1) Agregar una referencia para poder realizar la consulta con Excel. (Ver figuras 8.14 y 8.15.)

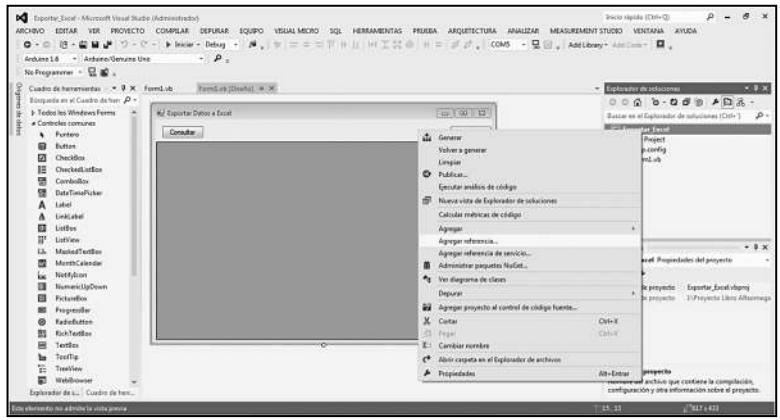


Figura 8.14 Elegir comando Agregar referencia para unir los datos con Excel.

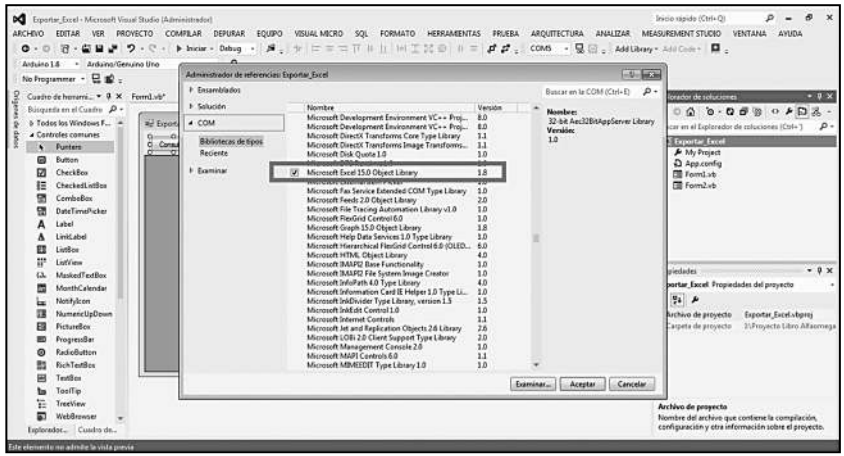


Figura 8.15 Seleccionar la referencia de Excel.

2) Después hay que establecer los códigos de la aplicación. Para los archivos de cabecera se utiliza la siguiente instrucción:

```
Imports System.Data.SqlClient
Imports Microsoft.Office.Interop
```

3) Después se extraen los datos de las consultas:

```
Public Function Extraer() As DataSet

        Dim conjunto As New DataSet
        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"
        Dim consulta As New SqlDataAdapter("select * from
measurements", conexion)

        Try
            conexion.Open()
            consulta.Fill(conjunto, "measurements")
            conexion.Close()
            Return conjunto

        Catch ex As Exception
            conexion.Close()
            Return conjunto
        End Try

    End Function
Public Function Extraer2() As DataSet
```

```
        Dim conjunto As New DataSet
        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"
        Dim consulta As New SqlDataAdapter("select * from
measurements where temp = 21", conexion)
        '>

        Try
            conexion.Open()
            consulta.Fill(conjunto, "measurements")
            conexion.Close()
            Return conjunto

        Catch ex As Exception
            conexion.Close()
            Return conjunto
        End Try

    End Function
```

4) Se establece la función del botón Consulta 1:

```
Private Sub Button1_Click(sender As Object, e As Even-
tArgs) Handles Button1.Click
        Dim dato As New DataSet
        dato = Extraer()
        DataGridView1.DataSource = dato.Tables("measure-
ments")
5) Después, la función del botón Consulta 2:
Private Sub Button3_Click(sender As Object, e As Even-
tArgs) Handles Button3.Click
        Dim dato As New DataSet
        dato = Extraer2()
        DataGridView1.DataSource = dato.Tables("measure-
ments")
    End Sub
```

6) Ahora se pueden exportar los datos a Excel:

```
Public Function GridAExcel(ByVal DGV As DataGridView) As
Boolean

        'Creamos las variables

        Dim exApp As New Excel.Application
        Dim exLibro As Microsoft.Office.Interop.Excel.Wor-
kbook
        Dim exHoja As Microsoft.Office.Interop.Excel.Wor-
ksheet
```

```
        Try

            exLibro = exApp.Workbooks.Add
            exHoja = exLibro.Worksheets.Add()

            'Cuantas columnas y cuantas filas?
            Dim NCol As Integer = DGV.ColumnCount
            Dim NRow As Integer = DGV.RowCount

            'recorremos todas las filas, y por cada fila
todas las columnas
            'y vamos escribiendo.
            For i As Integer = 1 To NCol
                exHoja.Cells.Item(1, i) = DGV.Columns(i -
1).Name.ToString
            Next

            For Fila As Integer = 0 To NRow - 1
                For Col As Integer = 0 To NCol - 1
                    exHoja.Cells.Item(Fila + 2, Col + 1)
= DGV.Rows(Fila).Cells(Col).Value()
                Next

            Next

            'Titulo en negrita, Alineado
            exHoja.Rows.Item(1).Font.Bold = 1
            exHoja.Rows.Item(1).HorizontalAlignment = 3
            exHoja.Columns.AutoFit()

            'para visualizar el libro
            exApp.Application.Visible = True
            exHoja = Nothing
            exLibro = Nothing
            exApp = Nothing


        Catch ex As Exception
            MsgBox(ex.Message, MsgBoxStyle.Critical,
"Error al exportar a Excel")

            Return False
        End Try
        Return True
    End Function
```

Para ello, hay que establecer la función del botón para exportar:

```
Private Sub Button2_Click(sender As Object, e As Even-
tArgs) Handles Button2.Click
        GridAExcel(DataGridView1)
    End Sub
```

La figura 8.16 muestra cómo se ve la interfaz de la aplicación.

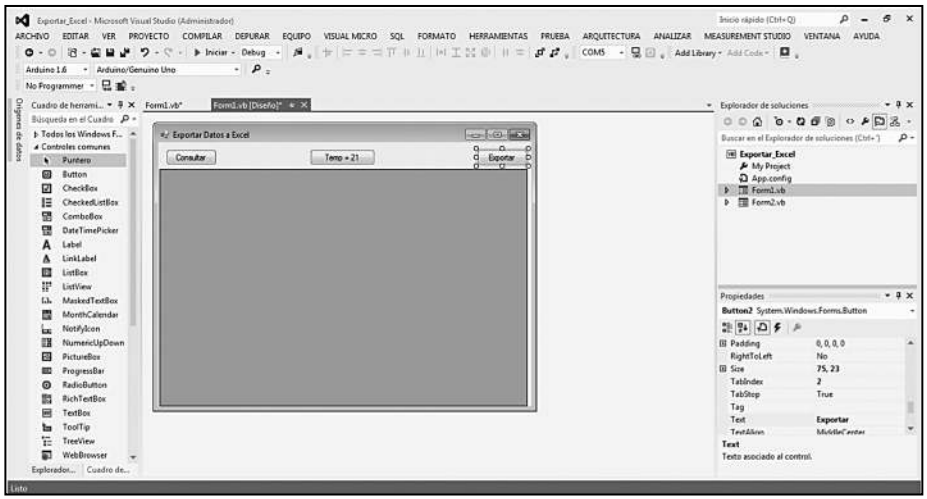


Figura 8.16 Interfaz de la aplicación.

La figura 8.17 muestra cómo se ven en pantalla los resultados de este proyecto.

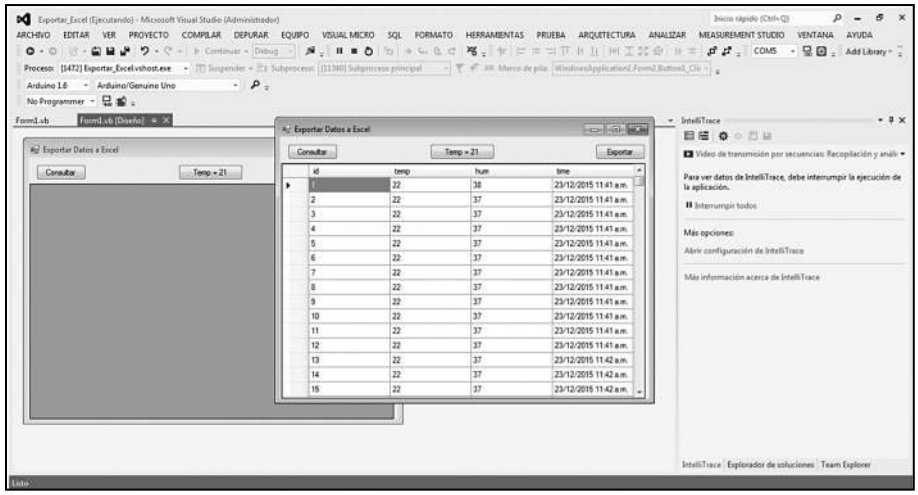


Figura 8.17 Base de datos en Excel con las lecturas de los sensores.

Ahora bien, puesto que los datos quedan registrados en Excel, se pueden hacer análisis de las variables de temperatura; un ejemplo de ello es que se puede consultar la temperatura es igual a 21, como se ve en la figura 8.18. Gracias a esto, se puede generar un reporte en Excel, como se muestra en la figura 8.19.

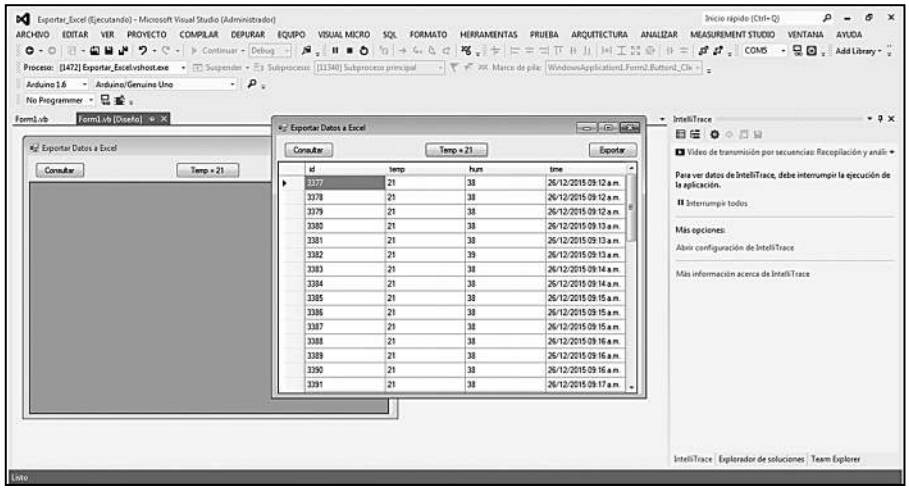


Figura 8.18 Consulta de temperaturas iguales a 21.

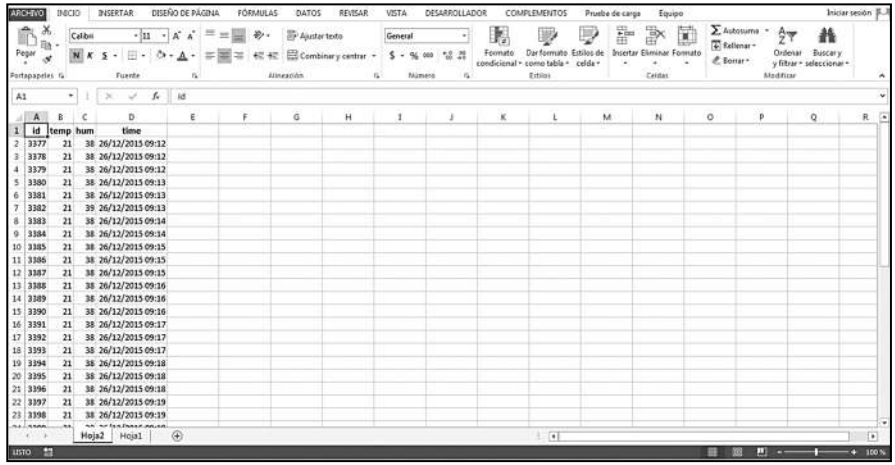


Figura 8.19 Generación del reporte en Excel.

8.9 Resumen

En este capítulo se realizó un proyecto que utiliza herramientas que ofrecen múltiples posibilidades en el desarrollo de software, tales como el acceso a bases de datos y al lenguaje de consultas SQL, conjuntando tanto la tarjeta Arduino como Visual Basic .NET y SQL Server 2012.

En la primera parte del proyecto se describió cómo guardar datos de forma local en una memoria Micro SD, que es un registro de datos en tiempo real. Esto es de gran

utilidad, puesto que se podrían agregar más sensores; por ejemplo, un sensor de presión barométrica o un sensor de presencia. Asimismo, se puede guardar el suceso ocurrido o el valor del sensor.

En la segunda parte del proyecto se conectó la Arduino a una base de datos para generar un registro de lecturas con los campos id, temperatura, humedad y fecha/hora. Este tipo de arreglo de los datos permite generar consultas y reportes de dichas lecturas.

Más aún, estos datos se pueden registrar en un archivo de Excel con el objetivo de utilizar dicha información para futuras consultas y para la toma de decisiones. Sin duda alguna, este tipo de proyectos se pueden aplicar en proyectos reales, a través del complemento de otras herramientas de software y hardware vistas en este libro.

En el siguiente capítulo se describirá cómo conectar la Arduino con Internet a un servidor web utilizando la comunicación Ethernet. Así se podrán enlazar los proyectos a todos los usuarios de Internet.

8.10 Problemas

1. En la memoria SD generar una secuencia de datos guardados y leerlos desde la tarjeta Arduino.
2. Llevar a cabo una activación de secuencias de encendido leyendo las lecturas desde la memoria SD.
3. Realizar un temporizador desde la tarjeta SD.
4. Conectar tres botones a la tarjeta Arduino y guardar el registro del botón que se presionó.
5. Generar una base de datos con el nombre de la columna id, activación y hora/tiempo del registro cada vez que se identifique un suceso de activación se insertará un registro.
6. Crear una interfaz en una Datagridview para mostrar los sucesos por día y fecha.
7. Realizar una consulta de las activaciones que se hicieron con más frecuencia.
8. Generar un reporte de monitoreo en la plataforma de Excel realizando una gráfica de las acciones más repetitivas en los tiempos.
9. Generar una base de datos con los datos de id, temperatura y humedad; y registrar en la base de datos únicamente las temperaturas entre 25 y 30 grados.
10. Generar un reporte en Excel de los datos solicitados y realizar una gráfica comparativa de cada día de registro.

Capítulo 9

Desarrollo de proyectos del Internet de las cosas basados en el Shield Ethernet de Arduino

Objetivos

En este capítulo se realizará una vasta descripción y ejemplificación de usos de Arduino, Visual Basic y aplicaciones web para realizar proyectos en la vida cotidiana de monitoreo y acceso inalámbrico. Se espera que con estos proyectos se desarrolle un mejor instinto de cómo aplicar las diferentes herramientas y aplicaciones y entonces empezar a desarrollar aplicaciones personales utilizando Arduino e Internet, y que tengan un uso real. Las posibilidades son numerosas. Por ejemplo, se pueden desarrollar aplicaciones que registren datos de diferentes tipos de sensores en Internet, que envíen datos a teléfonos celulares o controlen objetos de manera inalámbrica.

9.1 Introducción

En este capítulo se desarrollarán varios proyectos relacionados con el escudo Ethernet de Arduino que se aplican a todo lo que esté conectado: el Internet de las cosas. En la elaboración de estos proyectos se utilizará la tarjeta Arduino, la herramienta de ASP.NET y Visual Basic .NET, cuyo uso se ha estudiado en capítulos anteriores, aunque en éste aumentará la complejidad de cómo se aplicarán estos tres elementos en cada proyecto. Por ejemplo, en los capítulos anteriores se utilizó el puerto serial para comunicarnos con la PC; ahora, en este capítulo, se utilizará el escudo de comunicación de red con el fin de interactuar en una red local y una red remota a través de una página web creada en ASP.NET. Esto permitirá aumentar el rango de control a todo usuario de Internet.

9.2 Requerimientos de software y hardware

Con respecto al hardware, cada proyecto tendrá sus especificaciones, pero en todos ellos se utilizarán una o varias tarjetas Arduino UNO y el Shield Ethernet. (Ver figura 9.1.)

En cuanto al software, se utilizarán principalmente las herramientas de ASP.NET y Visual Basic .NET.

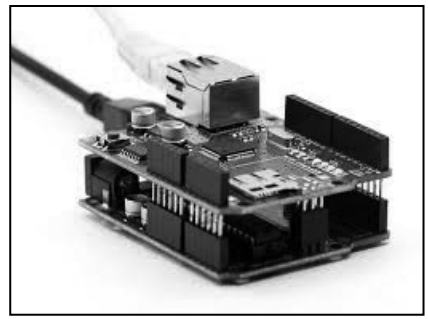


Figura 9.1 El Shield Ethernet.

9.3 Cómo se aprovecha la interacción entre servicios web y Arduino

En esta sección se describirán algunos proyectos en los que se puede aplicar el escudo Ethernet para establecer una comunicación entre servidores web y bases de datos, así como en proyectos donde interactúan las tecnologías web de .NET. Para ello, primero se describirán algunas propiedades de los elementos que participan en este tipo de aplicaciones.

9.3.1 El internet de las cosas

Todas las aplicaciones con este tipo de tecnología funcionan en el entorno del Internet de las cosas. Éste se refiere al mundo conectado; es decir, todos los dispositivos que tengan un microcontrolador como cerebro y una conexión a una red o a Internet, compartiendo información de un punto a otro, teniendo como punto central un servidor web y de bases de datos o corriendo servicios web en la nube, constituyen el **mundo conectado**. Algunos ejemplos de estos dispositivos son los electrodomésticos, como lavadoras, hornos, televisiones, refrigeradores. En otras palabras, se considera el Internet de las cosas a cualquier dispositivo que se requiera controlar de forma remota. De tal forma, entre las tecnologías de software que se utilizan en este tipo de aplicaciones se encuentran las páginas web HTML, los protocolos de comunicación, los servidores web IIS, los servidores de base de datos Microsoft SQL SERVER y otros servicios web. Por otro lado, las tecnologías de Arduino que se utilizan al implementar el Internet de las cosas son el escudo Ethernet, Arduino Wifi Shield, Arduino GSM/GPRS y Arduino GPS.

9.3.2 Usos de los servicios web

El término "servicios web" designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada.

Aunque los servicios web permitan que todos esos dispositivos dinámicos combinen varios servicios en aplicaciones, es necesario construir los servicios primero. Los lenguajes de programación en la Ciencia de la computación están en continua evolución. Se comenzó hace décadas con la idea de una función en la cual, al proporcionar algunos parámetros, ésta ejecuta alguna operación según esos parámetros y retorna un valor basado en los cálculos. Con el tiempo, ese primer concepto evolucionó al grado en que cada objeto, además de poder realizar varias funciones, también establece sus propias variables de datos privados en lugar de basarse en variables externas de todo el sistema, que anteriormente hacía más complejo el desarrollo de aplicaciones.

A medida que las aplicaciones comenzaron a comunicarse, el concepto de interfaces universales definidas para objetos empezó a cobrar importancia para permitir que objetos de otras plataformas se comunicaran entre sí, a pesar de estar escritos en lenguajes de programación distintos y operar en otros sistemas operativos.

Recientemente, los servicios web abordan el concepto de interfaces y comunicaciones definidas en XML para finalmente unir aplicaciones de cualquier tipo, además de dar la libertad de cambiar y evolucionar a lo largo del tiempo, a condición de que estén diseñadas para la interfaz adecuada.

Lo que distingue estos nuevos servicios web de las tecnologías de la generación anterior es la versatilidad de XML. Ésta permite separar la estructura gramatical (sintaxis) del significado gramatical (semántica) y, por lo tanto, la forma en que cada servicio del entorno procesa y entiende estructura y significado. En consecuencia, ahora los objetos se pueden definir como servicios que se comunican con otros servicios en la gramática definida por XML, donde cada servicio traduce y analiza el mensaje de acuerdo con la implementación local y el entorno. Por tanto, una aplicación conectada en red puede estar compuesta por varias entidades con varias construcciones y diseños diferentes, a condición de que cumplan con las reglas definidas por su arquitectura orientada a servicios.

Así, al tener esto en mente, los servicios web permiten que:

- Servicios en cualquier plataforma, escritos en cualquier lenguaje, interactúen entre sí.
- Se conceptualicen funciones de aplicaciones en tareas, lo que lleva al desarrollo y a flujos de trabajo orientados a tareas. Esto posibilita más abstracción del software, que puede ser empleado por usuarios menos técnicos que trabajan con análisis en el ámbito de los negocios.
- Haya un acoplamiento flojo, lo que implica que las interacciones entre las aplicaciones de servicio no se rompan cada vez que haya un cambio en la forma de diseño o implementación de un servicio o más.
- Se adapten las aplicaciones ya existentes a las cambiantes condiciones empresariales y necesidades de clientes.
- Se proporcionen aplicaciones de software ya existentes o ligadas con interfaces de servicio sin cambiar las aplicaciones originales, lo que permite operar totalmente en el entorno de servicios.
- Surjan otras funciones administrativas o de gestión de operaciones como confiabilidad, rendición de cuentas, seguridad, etc., independientemente de la función original, lo que aumenta su versatilidad y utilidad en el entorno de computación empresarial.

9.3.3 Estándares empleados en servicios web

Protocolo SOAP

SOAP (siglas en inglés de *Simple Object Access Protocol*) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Básicamente, SOAP es un paradigma de mensajería de una dirección sin estado, que se puede utilizar para formar protocolos más complejos y completos según las necesidades de las aplicaciones que lo implementen. Puede formar y construir la base de una "pila de protocolos de servicios web", ofreciendo un marco de mensajería básica en el cual los servicios web se puedan construir. Este protocolo está basado en XML y se conforma de tres partes:

1) El **sobre** (*envelope*), que define qué hay en el mensaje y cómo procesarlo.
2) El **conjunto de reglas de codificación**, que expresan instancias de tipos de datos.
3) La **convención**, que representa llamadas a procedimientos y respuestas.

Por otro lado, el protocolo SOAP tiene tres características principales:

1) **Extensibilidad** (seguridad y WS-routing son extensiones aplicadas en el desarrollo).
2) **Neutralidad** (SOAP se puede utilizar sobre cualquier protocolo de transporte, como HTTP, SMTP, TCP o JMS).
3) **Independencia** (SOAP permite cualquier modelo de programación).

Por ejemplo, un mensaje SOAP podría ser enviado a un servicio web para realizar la búsqueda de algún precio en una base de datos, indicando para ello los parámetros necesitados en la consulta. El servicio podría generar un documento en formato XML con el resultado, que puede ser el precio, la localización o sus características. Al tener los datos de respuesta en un formato estandarizado procesable (en inglés llamado *parsable*), éste puede ser integrado directamente en un sitio web o aplicación externa.

La arquitectura SOAP está formada por varias capas de especificación: los patrones de intercambio de mensajes (MEP, por sus siglas en inglés: *Message Exchange Patterns*) para el formato del mensaje, enlaces subyacentes del protocolo de transporte, el modelo de procesamiento de mensajes y la capa de extensibilidad del protocolo. SOAP es el sucesor de XML-RPC, a pesar de que toma el transporte y la neutralidad de la interacción, así como el sobre (*envelope*), encabezado (*header*) o cuerpo (*body*) de otros modelos, probablemente de WDDX.

Arquitectura de software REST

La Transferencia de Estado Representacional o REST (por sus siglas en inglés: *Representational State Transfer*) es un estilo de arquitectura software para sistemas hipermedia distribuidos, como la *World Wide Web*. Los sistemas que siguen los principios REST se llaman con frecuencia "RESTful" y tienen las siguientes características:

- Un **protocolo cliente/servidor sin estado**. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en

HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URL, no son permitidas por REST).

- Un conjunto de **operaciones bien definidas** que se aplican a todos los *recursos* de información. HTTP en sí define un conjunto pequeño de operaciones; las más importantes son post, get, put y delete.
- Una **sintaxis universal** para identificar los recursos. En un sistema REST, cada recurso se puede direccionar únicamente a través de su URL.
- El **uso de hipermedios** tanto para la información de la aplicación como para las transiciones de estado de la aplicación. La representación de este estado en un sistema REST es típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces, sin requerir el uso de registros u otra infraestructura adicional.

9.4 Servicios web aplicados a Arduino

En esta sección se presentan dos ejemplos de servicios web que utilizan las herramientas de Arduino en conjunto con ASP.NET.

9.4.1 Ejemplo de servicio web con SOAP

En la tecnología de ASP.NET se puede crear el servicio web utilizando el estándar SOAP. Se debe utilizar la siguiente estructura del servicio creado, en donde la función Add recibe un valor "num1" y regresa un número decimal sumado en uno, como se puede apreciar en este código:

```
Public Class WebService2
     Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function Add(num1 As Double) As Double
        Return num1 + 1
    End Function

End Class
```

La respuesta del servicio web con HTTP GET es la siguiente (es necesario reemplazar con valores reales los marcadores de posición que aparecen):

```
GET /WEBService/WebService2.asmx/Add?num1=string HTTP/1.1
Host: localhost
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version= "1.0" encoding= "utf-8"?>
<int xmlns="http://tempuri.org/">int</int>
```

En la tarjeta Arduino aparecerá una línea que permitirá invocar al servicio web y que tal dato sea recibido por éste en ASP.NET. En este caso, la línea sería:

```
client.println("GET /WEBService/WebService2.asmx/Ad-
d?num1=23.6");
client.println(" HTTP/1.1");
client.println("Host: 192.168.1.104");
client.println("Connection: close");
client.println("Content-Type: text/xml; charset=UTF-8");
client.println("Content-Length: length");
client.println();
```

La figura 9.2. muestra cómo se ejecuta el servicio web.

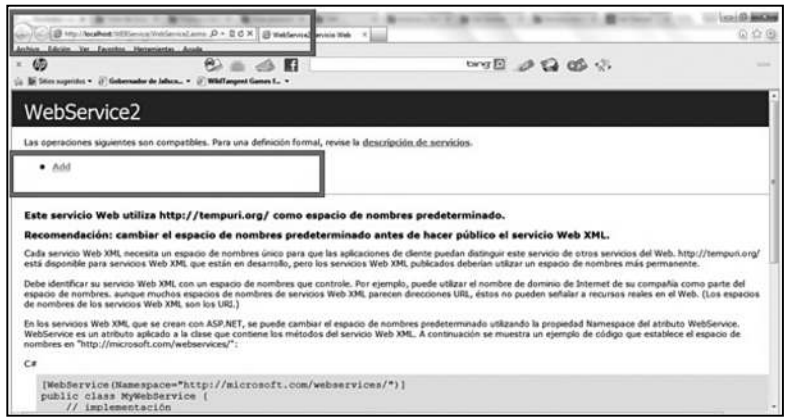


Figura 9.2 Ejecutando el servicio web.

Después, debe capturarse el parámetro. (Ver figura 9.3.)

Figura 9.3 Captura del parámetro.

El servicio web dará la respuesta que se muestra en la figura 9.4.

Figura 9.4 Respuesta del servicio web después de invocarlo.

Ahora, la tarjeta Arduino será la que consumirá al servicio web enviando los parámetros al servidor web. Para ello, antes de enviar datos desde Arduino, primero se hace una prueba del envío del dato por la URL. Se envía la siguiente cadena: http://localhost/WEBService/WebService2.asmx/Add?num1=23.6

Una vez que se conoce la ruta, se envía el dato desde Arduino para invocar el servicio web. La respuesta llega desde el servidor web IIS, como lo muestra la figura 9.5.

Figura 9.5 Respuesta de retorno desde el servidor web IIS.

El código completo es el siguiente:

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x0A, 0xC2 };

IPAddress server(192,168,1,104);

IPAddress ip(192,168,1,150);

EthernetClient client;

void setup() {
Serial.begin(9600);

if (Ethernet.begin(mac) == 0) {
Serial.println("Fallo la conexion con el servidor DHCP");
}

delay(1000);
Serial.println("conectando…");

if (client.connect(server,80)) {
Serial.println("conectado");
client.println("GET /WEBService/WebService2.asmx/Ad-
d?num1=23.6");
client.println(" HTTP/1.1");
client.println("Host: 192.168.1.104");
client.println("Connection: close");
client.println("Content-Type: text/xml; charset=UTF-8");
client.println("Content-Length: length");
client.println();
}
else {
Serial.println("connection failed");
}
}

void loop()
{
if (client.available()) {
char c = client.read();
Serial.print(c);
}

if (!client.connected()) {
Serial.println();
Serial.println("desconectado");
client.stop();

while(true);
}
}
```

9.4.2 Ejemplo de conversión de temperatura a través de un servicio web

En esta sección se explicará un ejemplo de servicios web con Arduino para convertir temperaturas de grados centígrados a grados Fahrenheit. Los primeros pasos son:

1) Conectar el sensor DHT11 al pin 11 del Arduino, la tierra (GND) y VCC +5 volts al Arduino.
2) Adquirir los datos del sensor.
3) Enviar la temperatura leída a través del servicio web.

La cadena de envío de datos por URL desde el navegador es:

```
http://localhost/ArduinoServicioWEB_Temp/ServicioWebAR-
duinoTemp.asmx/conversion?gradosC=24
```

La figura 9.6. muestra cómo se ve en pantalla. El mensaje es:

```
<double xmlns="http://tempuri.org/">75.2</double>
```

Figura 9.6 Captura de pantalla de la URL.

Después, Arduino empieza su labor de consumir el servicio web. Ésta es la cadena de datos:

```
client.println("GET /ArduinoServicioWEB_Temp/ServicioWe-
bARduinoTemp.asmx/conversion?gradosC=" + temp);
client.println(" HTTP/1.1");
client.println("Host: 192.168.1.104");
client.println("Connection: close");
client.println("Content-Type: text/xml; charset=UTF-8");
client.println("Content-Length: length");
client.println();
```

La figura 9.7 muestra cómo se ven los valores recibidos en pantalla.

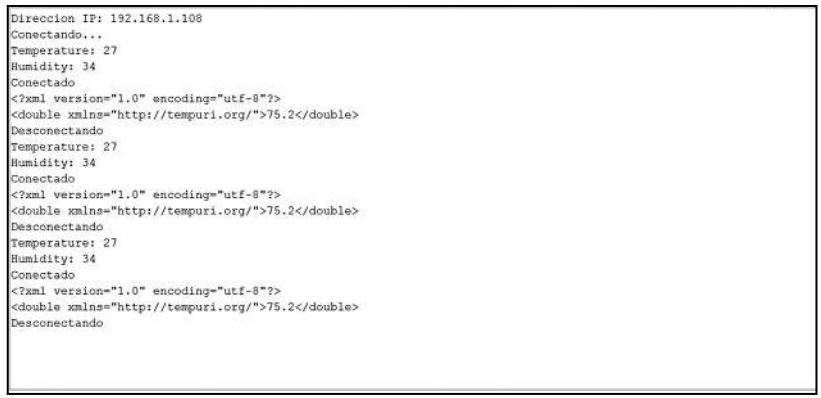


Figura 9.7 Valor de la conversión recibido.

El recuadro de la figura 9.8 muestra los datos que arrojaría el servicio web si se pide convertir 28 grados centígrados a grados Fahrenheit.

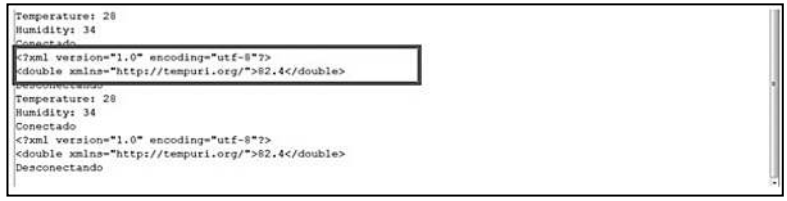


Figura 9.8 El servicio web recibe el parámetro “28 grados centígrados”, realiza la conversión y envía la respuesta a Arduino.

La figura 9.9 muestra cómo se hace la prueba desde la URL, donde se verifica en el navegador que el dato sea el mismo. El mensaje es:

```
<double xmlns="http://tempuri.org/">82.4</double>
```

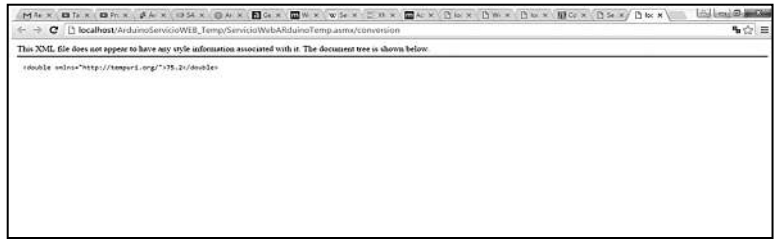


Figura 9.9 Prueba desde la URL.

La figura 9.10 muestra lo que sucede cuando se envía una conversión de 24 grados centígrados:

```
<doublé xmlns="http://tempura.org/">75.2</double>
```

Figura 9.10 Conversión a 24 grados centígrados.

De tal forma, el código completo de la aplicación sería:

```
//Ejemplo Arduino consume Servicio WEB en ASP .NET para
convertir temperatura

#include <SPI.h>
#include <Ethernet.h>
#include <DHT.h>

// Enter a MAC address for your controller below.
byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x0A, 0xC2 };

// Define los pines del sensor
#define DHTPIN 7
#define DHTTYPE DHT11

//Direccion IP del servidor WEB
IPAddress server(192,168,1,104);

//Si el servidor DHPCP no asigna dirección se le asigna
la IP fija
IPAddress ip(192,168,1,150);

//Se inicializa el cliente Arduino
EthernetClient client;
//Instancia del sensor
DHT dht(DHTPIN, DHTTYPE);
```

```
void setup(){
Serial.begin(9600);

// Inicializa la asignación de la IP
if (Ethernet.begin(mac) == 0) {
   Serial.println("No pudo asignar IP el servidor DHCP");
   Ethernet.begin(mac, ip);
}

// Se muestra la dirección IP
  Serial.print("Direccion IP: ");
  Serial.println(Ethernet.localIP());

// Se le da tiempo para que se conecte
delay(1000);
Serial.println("Conectando...");
}

void loop() {
// Lectura de sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Transformando a string
  String temp = String((int) t);
  String hum = String((int) h);

  // Se despliega en el monitor serial
  Serial.println("Temperature: " + temp);
  Serial.println("Humidity: " + hum);

  // Se conecta a servidor WEB IIS
  if (client.connect(server, 80)){
    if (client.connected()){
      Serial.println("Conectado");

// Hace la peticion HTTP
client.println("GET /ArduinoServicioWEB_Temp/ServicioWe-
bARduinoTemp.asmx/conversion?gradosC=" + temp);
client.println(" HTTP/1.1");
client.println("Host: 192.168.1.104");
client.println("Connection: close");
client.println("Content-Type: text/xml; charset=UTF-8");
client.println("Content-Length: length");
client.println();
}
else {
 //si no se pudo conectar
 Serial.println("Fallo la conexion");
    }
   // Lee la respuesta del servidor
```

```
    while (client.connected()){
          while (client.available()) {
          char c = client.read();
          Serial.print(c);
      }
   }
    // Si el servidor se desconecta, el cliente se desco-
necta
    if (!client.connected()) {
      Serial.println();
      Serial.println("Desconectado");
      client.stop();
    }
  }
  // Repite la conexion cada segundo
  delay(1000);
}
```

Ahora bien, el código de servicio web en ASP.NET sería el siguiente:

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols
```

Para permitir que se llame a este servicio web desde un script, usando ASP.NET AJAX, quite la marca de comentario de la línea siguiente:

```
' <System.Web.Script.Services.ScriptService()> _
<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfi-
le1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.Designer-
Generated()> _
Public Class ServicioWebARduinoTemp
     Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function conversion(gradosC As Single) As Single
        Return (gradosC * 1.8) + 32.0
    End Function

End Class
```

En este paso es muy importante incluir en el archivo Web.Config las siguientes líneas que permitirán el acceso mediante peticiones GET y POST, lo cual se refiere a la etiqueta webservices:

```
<?xml version="1.0"?>
<!--
  Para obtener más información sobre cómo configurar la
aplicación de ASP.NET, visite
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
```

```
<configuration>
  <system.web>
      <webServices>
      <protocols>
        <add name="HttpGet"/>
        <add name="HttpPost"/>
      </protocols>
    </webServices>
    <compilation debug="true" strict="false" explicit="-
true" targetFramework="4.5"/>
    <httpRuntime targetFramework="4.5"/>
  </system.web>
 </configuration>
```

Envío de datos a servidor web

Es importante saber que la tarjeta Arduino se puede comunicar a una red local y, por supuesto, a una red como Internet. Sin duda, este tipo de ventajas son de gran potencialidad, ya que la tarjeta podrá interactuar con equipos que también se encuentren conectados a la red. En este punto se explicará cómo conectarse a un servidor de base de datos configurado en ASP.NET.

Primeramente, es fundamental entender que la tarjeta Arduino será el equipo cliente en la red y el servidor remoto será la computadora que tenga configurado el servidor de base de datos y el servidor web IIS. En el capítulo anterior las lecturas de los sensores se guardaron en una memoria Micro SD; ahora los datos se almacenarán en un sistema de base de datos, gracias al cual se podrán tener grandes volúmenes de información de eventos o lecturas de sensores que pudieran ocurrir en un proceso o sistema.

A continuación se presenta el sketch de Arduino para realizar la conexión con el servidor web y la base de datos:

1) Usar las siguientes librerías:

```
#include <SPI.h>
#include <Ethernet.h>
#include "DHT.h"
```

2) Establecer la dirección MAC de la tarjeta Arduino:

```
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xFE, 0x40 };

#define DHTPIN 7
#define DHTTYPE DHT11
```

3) Verificar que la dirección IP de la Arduino y la dirección IP del servidor web estén en la misma red:

```
IPAddress ip(192,168,1,50);

IPAddress server(192,168,1,100);


EthernetClient client;
```

```
DHT dht(DHTPIN, DHTTYPE);

void setup() {
     Serial.begin(9600);
  if (Ethernet.begin(mac) == 0) {
    Serial.println("No se inicializar el servidor DHCP");
    Ethernet.begin(mac, ip);
  }
  Serial.print("IP address: ");
  Serial.println(Ethernet.localIP());

  delay(1000);
  Serial.println("Conectando...");
}
void loop()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  String temp = String((int) t);
  String hum = String((int) h);

  Serial.println("Temperature: " + temp);
  Serial.println("Humidity: " + hum);
```

4) Establecer la conexión con la base de datos, puerto 80:

```
  if (client.connect(server, 80)) {
  if (client.connected()) {
  Serial.println("conectado");
```

5) Realizar la petición al servidor a través del método GET; colocar el nombre de la página web y pasar los parámetros de envío:

```
     client.println("GET /HttpGET/Default.aspx?temp=" +
     temp + "&hum=" + hum + " HTTP/1.1");
     client.println("Host: 192.168.1.100");
     client.println("Connection: close");
     client.println();
    }
    else {
      Serial.println("Fallo la conexión con el servi-
dor");
    }
```

6) Así debe ser la lectura de la respuesta:

```
   while (client.connected()){
         while (client.available()) {
         char c = client.read();
         Serial.print(c);
      }
   }
```

7) Así debe verse cuando el cliente se desconecta:

```
    if (!client.connected()) {
      Serial.println();
      Serial.println("desconectando.");
      client.stop();
    }

  }
```

8) Cada lectura se repite en un determinado tiempo:

```
  delay(10000);
}
```

9) Realizar una consulta de los datos. Para ello, escribir en el navegador la siguiente URL (como se muestra en la figura 9.11:

```
http://localhost/HttpGET/Default.aspx?temp=24&hum=33
```

Figura 9.11 Consulta de datos.

10) Después se escribe el código del script en ASP .NET:

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Dim temp As Double
    Dim hum As Double
```

11) En el evento "load" se leen los valores enviados desde el Arduino; las variables temp y hum se publican y se obtienen los valores con la instrucción Request.QueryString, según el siguiente código:

```
    Protected Sub Page_Load(sender As Object, e As Even-
tArgs) Handles Me.Load

        Dim temp1, hum1 As Double
        temp = Request.QueryString("temp")
        hum = Request.QueryString("hum")

    End Sub
```

Se pueden agregar estas funciones, de la siguiente manera, para que el servidor responda cuando haya recibido la información:

```
Response.Write("Datos Recibidos")
```

12) Se escribe el código para insertar las lecturas en SQL server:

```
Imports System.Data.SqlClient

Dim temp1, hum1 As Double
If Insertar(temp1, hum1) Then
            mensajeguardado.Text = "Lectura guardada..."
        Else
            mensajeguardado.Text = "los datos no se guar-
daron"
        End If
```

La función para insertar las lecturas es:

```
Public Function Insertar(ByVal temp1 As Double, ByVal
hum1 As Double) As Boolean

        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"
        Dim consulta As New SqlCommand("INSERT INTO me-
asurements(temp,hum) VALUES('" & temp & "','" & hum &
"')", conexion)

        Try
            conexion.Open()
            consulta.ExecuteNonQuery()
            conexion.Close()
            Return True

        Catch ex As Exception
            ' MsgBox(ex.Message, MsgBoxStyle.Exclamation)
```

```
            conexion.Close()
            Return False
        End Try

    End Function
```

La función para consultar las lecturas es:

```
Imports System.Data.SqlClient
Imports System.Data

Partial Class Consulta
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click

        Dim dato As New DataSet
        dato = Extraer()
        GridView1.DataSource = dato
        GridView1.DataBind()


    End Sub

Public Function Extraer() As DataSet

        Dim conjunto As New DataSet
        Dim conexion As New SqlConnection

        conexion.ConnectionString = "Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123"

        Dim consulta As New SqlDataAdapter("select * from
measurements", conexion)

        Try
            conexion.Open()
            consulta.Fill(conjunto, "measurements")
            conexion.Close()
            Return conjunto

        Catch ex As Exception
            conexion.Close()
            Return conjunto
        End Try

    End Function
```

Los resultados de la figura 9.12 se refieren a las consultas realizadas de la información.

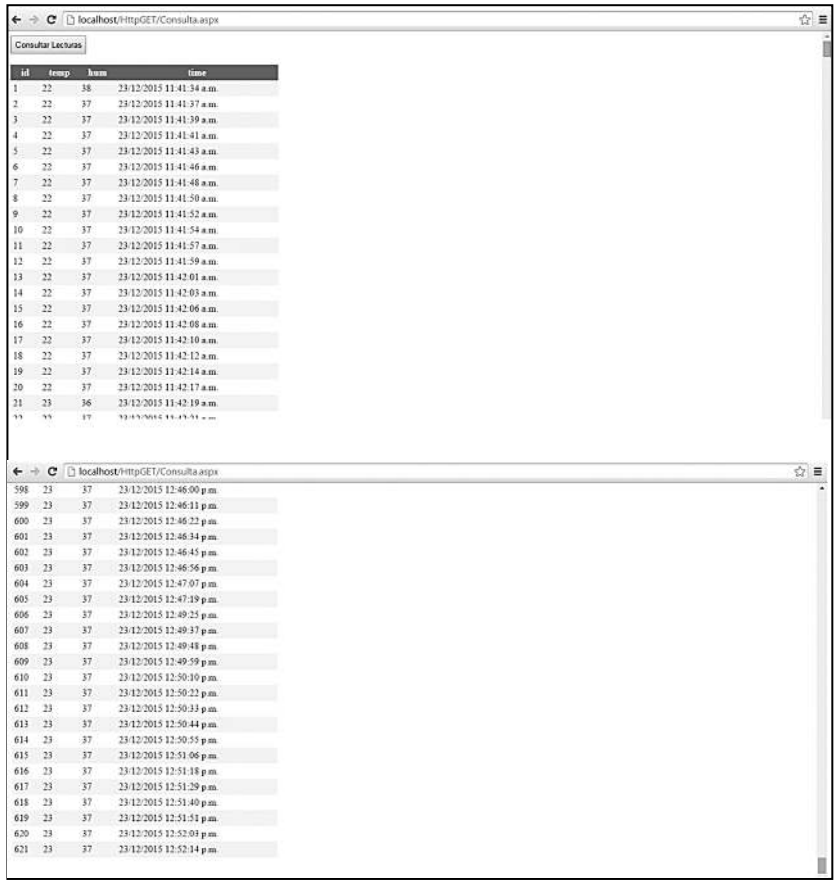


Figura 9.12 Consulta de datos.

Consulta mediante servicios web

Un punto muy importante dentro de las aplicaciones del Internet de las cosas es el acceso a datos a través de una base de datos. En este proyecto de conversión de temperatura la idea es conectarse desde un cliente web para consultar información de una base de datos a través de un servicio web. Esto permitirá crear aplicaciones para generar bases de datos y, a través de los datos guardados, realizar consultas que permitan mandar avisos de posibles eventos que se generen dentro de un sistema o mandar alertas a los dispositivos que se encuentran conectados al cerebro del sistema. En la figura 9.13 podemos ver cómo interactúan el servicio web, la tarjeta Arduino, la nube y ASP.NET.

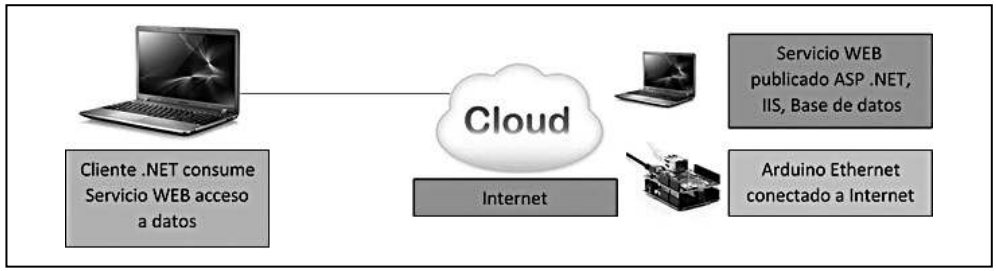


Figura 9.13 Esquema de la conectividad de la aplicación.

Servicio web para acceso a los datos previamente almacenados

En esta sección se describirá cómo establecer el servicio web para tener acceso a la base de datos en Visual Basic.

1) Se crea una aplicación web. (Ver figura 9.14.)

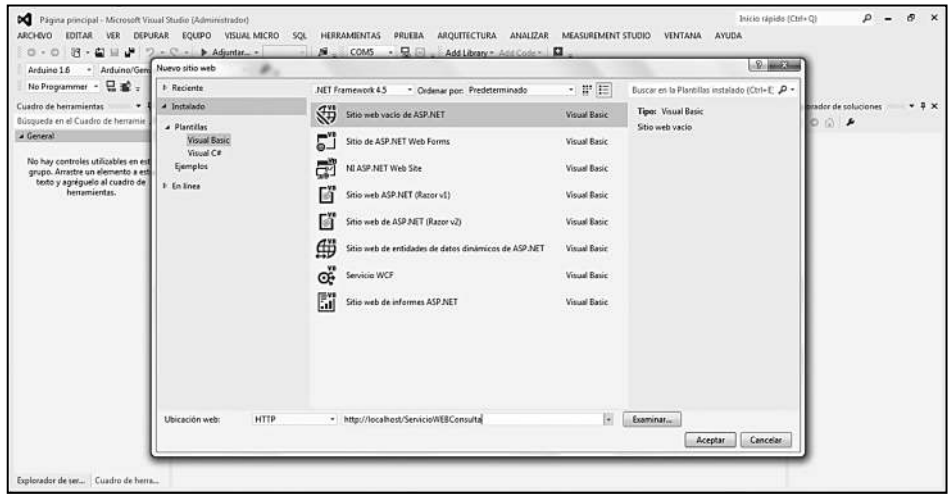


Figura 9.14 Desde las plantillas de Visual Basic se da clic a un nuevo sitio web de ASP.NET.

2) Se crea un servicio agregando un nuevo elemento. (Ver figura 9.15.)

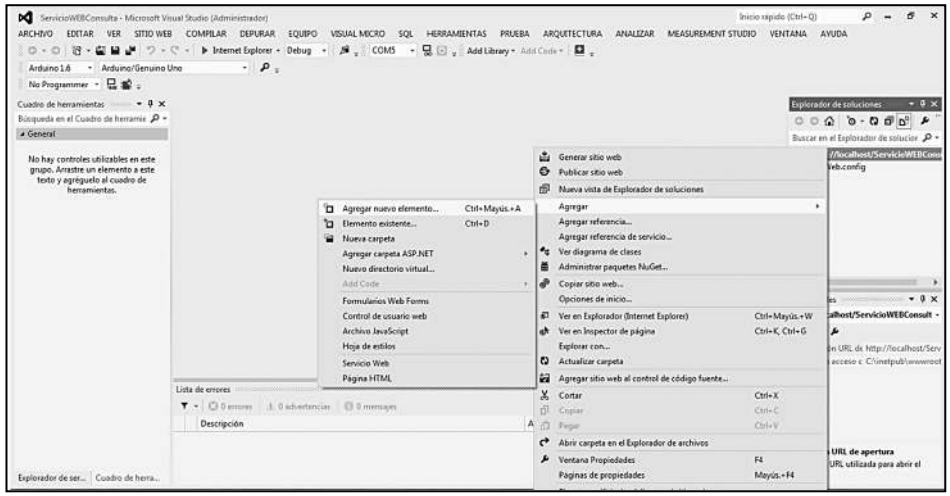


Figura 9.15 En el menú Agregar, dar clic a Agregar nuevo elemento.

3) Seleccionar servicio web y nombrar con la extensión .asmx (ver figura 9.16.)

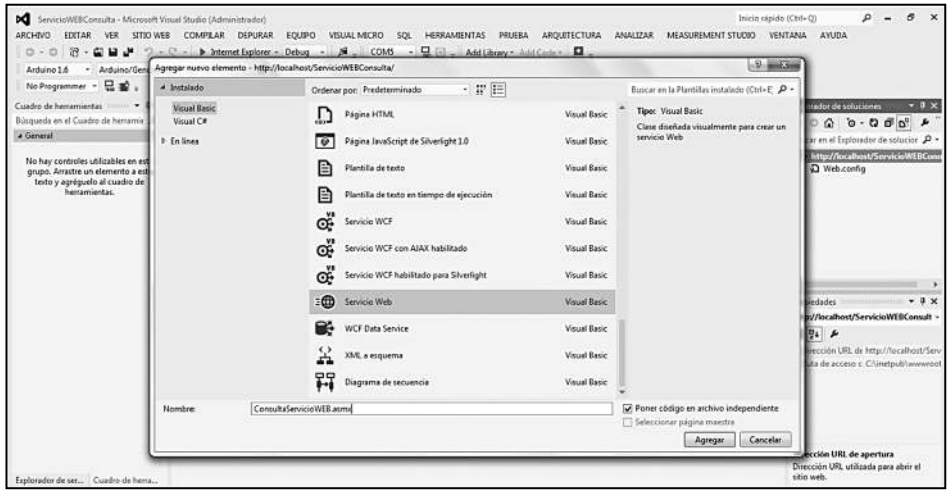


Figura 9.16 Dar clic en Servicio Web.

4) Agregar el método que tendrá el servicio web. (Ver figura 9.17.)

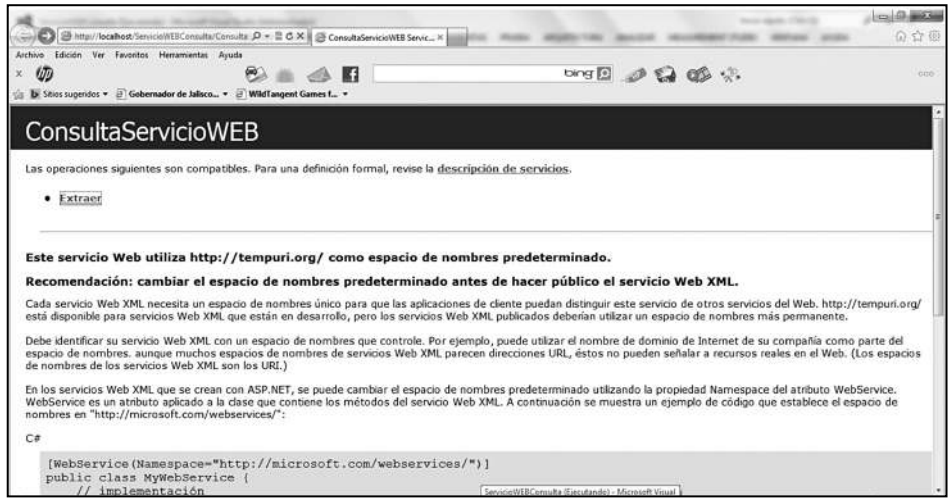


Figura 9.17 Pantalla de ejecución del servicio web.

5) Invocar el servicio web en SOAP, versión 1.1. (Ver figura 9.18.)

Figura 9.18 Invocar el servicio web.

En el localhost estará el archivo en formato XML de los datos. Su estructura se muestra en la figura 9.19.

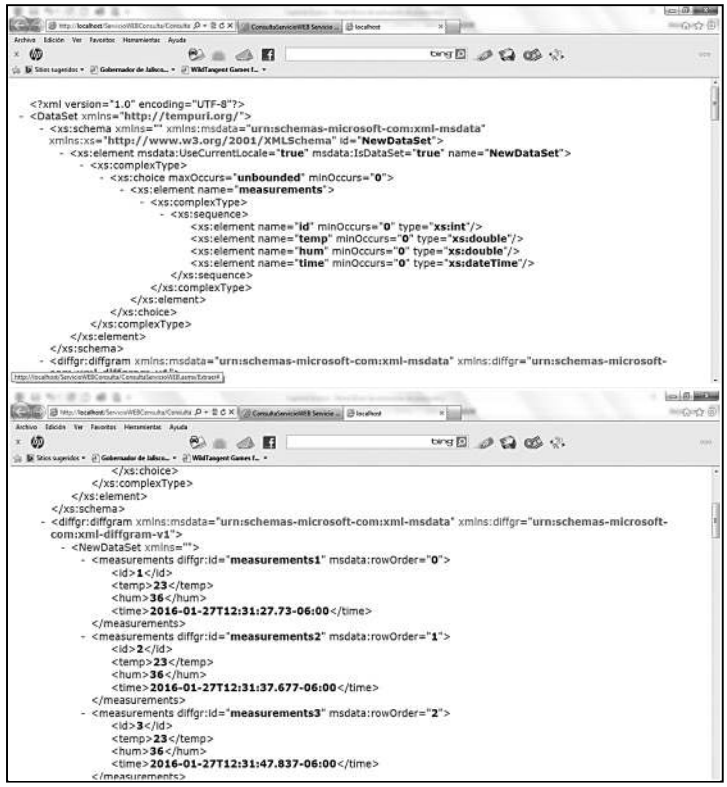


Figura 9.19 Estructura del archivo en XML.

6) Establecer el código del servicio web con el siguiente formato de instrucción:

```
Public Class ConsultaServicioWEB
    Inherits System.Web.Services.WebService
    Dim cadena As New SqlConnection("Data Source=ULTI-
MATE-PC\SERVER2012;Initial Catalog=prueba2;User ID=-
sa;Password=Hol\123")

    <WebMethod()> _
    Public Function Extraer() As DataSet
        Dim datas As New SqlDataAdapter("SELECT * FROM
measurements", Cadena)
        Cadena.open()
        Dim SetData As New DataSet
        datas.Fill(SetData, "measurements")
        cadena.close()
        Return SetData
    End Function
End Class
```

De tal forma, la aplicación cliente consume el servicio web para consultar información.

7) Agregar una referencia web, como se muestra en la figura 9.20.

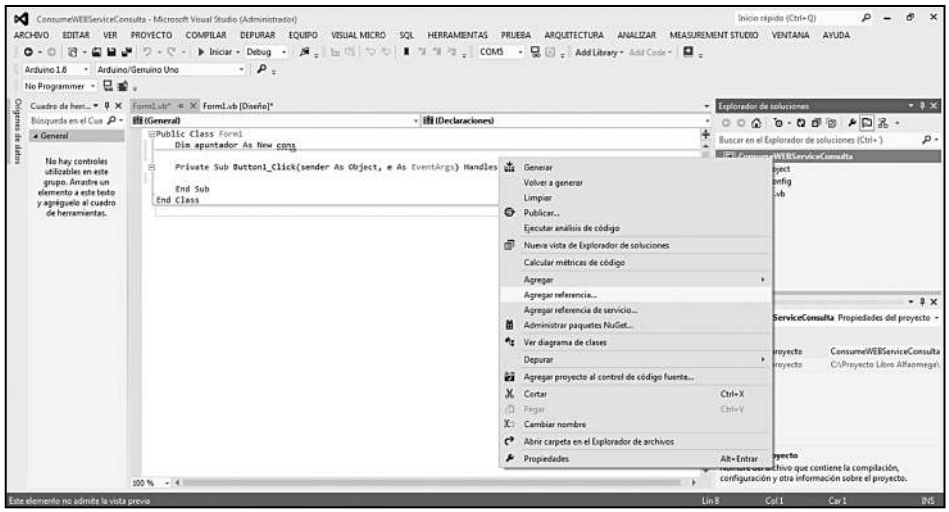


Figura 9.20 Se agrega una referencia a la aplicación cliente.

8) Establecer la dirección del servicio web para agregar la referencia (Ver figura 9.21.)

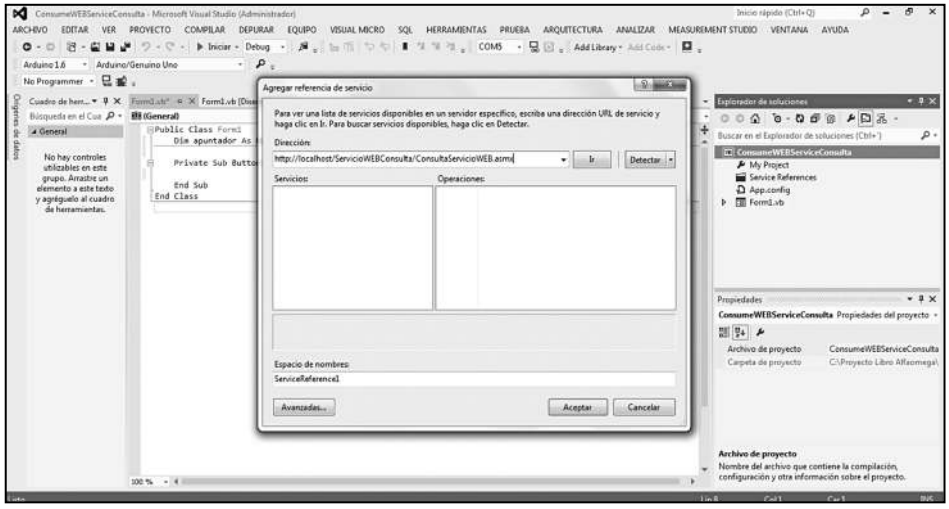


Figura 9.21 Dirección del servicio web para agregar la referencia.

9) Al detectar el servicio web, dar clic en Extraer (Ver figura 9.22.)

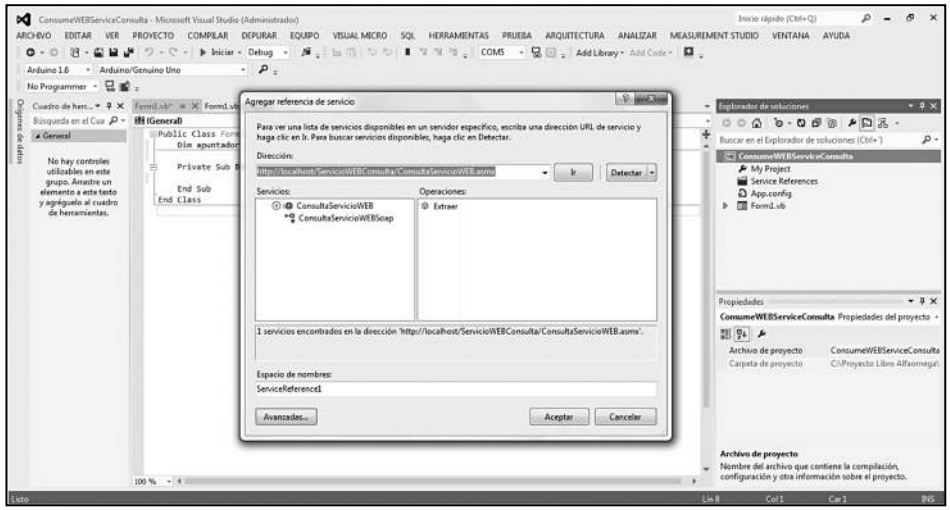


Figura 9.22 Extrayendo el servicio web encontrado.

10) Escribir en el Espacio de nombres el nombre de la referencia; es decir "ServicioConsulta", sin comillas. (Ver figura 9.23.)

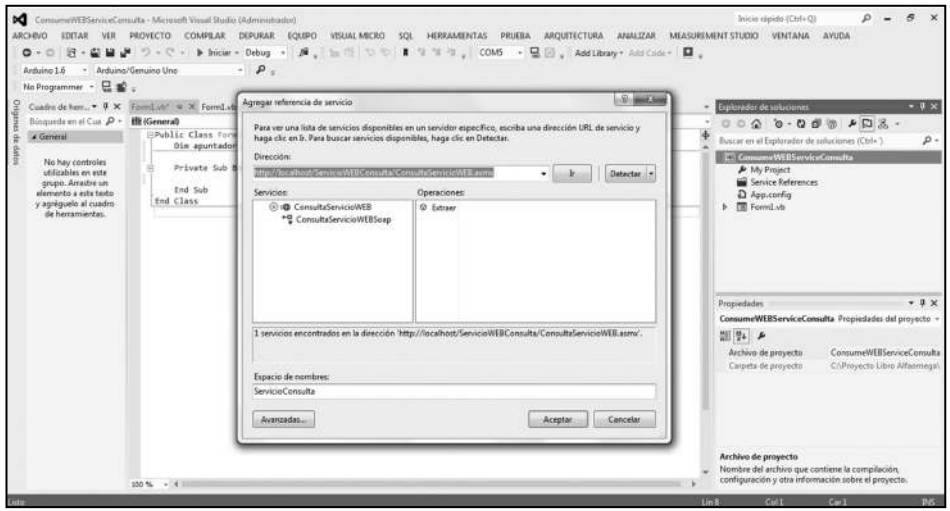


Figura 9.23 Escribir el nombre de la referencia de servicio web.

Si se ha tenido éxito al agregar la referencia, la pantalla debe verse tal como la figura 9.24.

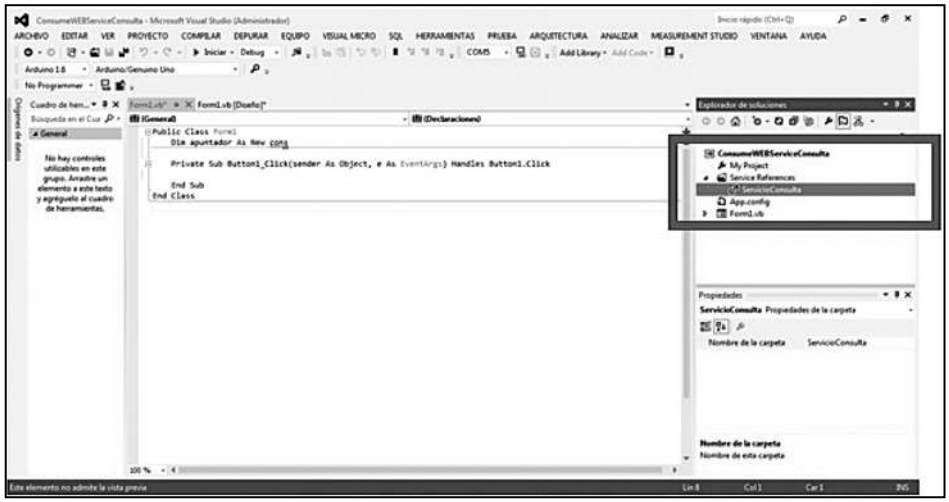


Figura 9.24 Se agregó la referencia al servicio web.

11) A continuación, se presenta el llamado del servicio web desde la aplicación de escritorio. Para ello:

a) Se declara una variable que hace el llamado al servicio web SOAP:

```
Public Class Form1
    Dim consulta As New ServicioConsulta.ConsultaServi-
cioWEBSoapClient
```

b) Se crea el dataset y se manda llamar el método creado a través del llamado del servicio web:

```
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim C1 As New DataSet
        C1 = consulta.Extraer()
```

c) Se llena el datagrid view (vista como hoja de cálculo) con los datos extraídos desde el servicio web. (Ver figura 9.25):

```
        DataGridView1.DataSource = C1.Tables("measure-
ments")
    End Sub
End Class
```

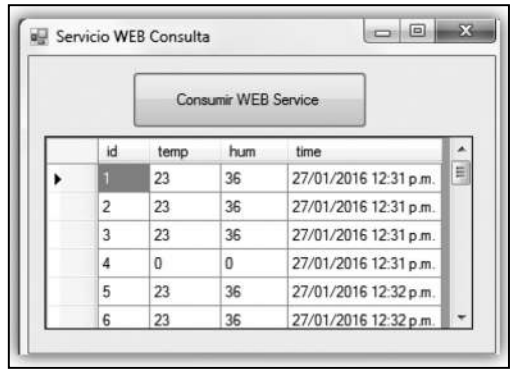


Figura 9.25 Pantalla de datos extraídos desde el servicio web.

Retomando lo estudiado en este ejemplo, hasta ahora se ha explicado una forma en que se pueden aplicar los conocimientos estudiados sobre bases de datos y programación en Visual Basic. NET con ASP.NET. La finalidad es implementar un servicio web que permita realizar aplicaciones de acceso remoto utilizando esta tecnología de servicios web. Ésta permite desarrollar otro tipo de aplicaciones; por ejemplo, guardar los datos de forma local en la tarjeta Arduino, a través de la memoria Micro SD. Es decir, cuando no exista conexión a Internet, los datos se almacenarán de forma local; cuando se esté conectado a Internet, el sistema se podrá sincronizar con la base de datos para almacenar los datos y que la información se mantenga actualizada.

Otra aplicación podría ser tener más de un dispositivo Arduino-Ethernet conectado a un servidor web. Cada uno de ellos estaría enviando datos al servidor, actualizando la información y sincronizando cada dispositivo desde un sitio web.

9.4.3 Comandos de servicios RESTful con Arduino

En esta sección se aprenderá a controlar la tarjeta Arduino Ethernet mediante servicios RESTful desde una interfaz de una página web en ASP.NET.

Para llevar esto a cabo es necesario tener instalada la carpeta aREST.h, de forma que se pueda utilizar esta herramienta directamente desde el software de páginas web.

Más aún, se deben conocer los comandos que permiten establecer comunicación a través de una red. Para ello, se creó el API Arest, que permite enviar comandos a la tarjeta y obtener información, entre otros usos. Los datos obtenidos están en un JSON, el cual permite interactuar con diversas plataformas para poder interpretar tal información y hacer uso de ella. Los comandos son los siguientes:

Comando para configurar la salida 6 como salida:

```
192.168.115.102/mode/6/o
```

Comando para poner en alto la salida 6:

```
192.168.115.102/digital/6/1
```

Comando para leer los datos del Arduino con un ID:

```
192.168.115.102/id
```

Comando para leer una variable que se publique (puede ser la lectura de un sensor de temperatura):

```
192.168.115.102/temperature
```

Éste es el formato que surge como respuesta después de que se llaman los comandos descritos:

```
{"message": "Pin D6 set to 1", "id": "1", "name": "ardui-
no", "connected": true}. El punto forma parte del coman-
do? Si no, quitarlo.
```

En el ejemplo de la siguiente sección se explica cómo leer dicho formato en la página web.

9.4.4 Control del Ethernet Shield con ASP.NET

Es posible implementar el servicio WEB RESTful en una aplicación ASP.NET. Para ello, es importante que se haga a través de la clase WebClient en ASP.NET, porque permite conectarse a un recurso y enviarlo en formato de cadena. El comando para controlar es el siguiente:

```
myWebClient.OpenWrite("http://192.168.1.108/digital/6/1")
```

La instrucción "openwrite" escribe en el recurso establecido, a través de la dirección IP de nuestra tarjeta Arduino, y se le envían los comandos que nos permitirán controlar el dispositivo. La ventaja de esto es que dentro de la misma página web se podrán insertar y controlar todos los dispositivos que se requieran en una misma pantalla.

El código de la aplicación es:

```
Imports System.Net
Imports System.IO

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        Dim myWebClient As New WebClient()
        Dim postStream1 As Stream = myWebClient.OpenWri-
te("http://192.168.1.108/digital/6/1")
        postStream1.Close()
    End Sub

    Protected Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Dim myWebClient As New WebClient()
```

```
        Dim postStream2 As Stream = myWebClient.OpenWri-
te("http://192.168.1.108/digital/6/0")
        postStream2.Close()
    End Sub

    Protected Sub Page_Load(sender As Object, e As Even-
tArgs) Handles Me.Load
        Dim myWebClient As New WebClient()
        Dim postStream3 As Stream = myWebClient.OpenWri-
te("http://192.168.1.108/mode/6/o")
        postStream3.Close()
    End Sub

    Protected Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/id")
        Dim dato As String = lectura
        TextBox1.Text = dato
    End Sub

   Protected Sub Button4_Click(sender As Object, e As
EventArgs) Handles Button4.Click
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/temperature")
        Dim dato As String = lectura
        TextBox2.Text = dato
    End Sub

    Protected Sub Button5_Click(sender As Object, e As
EventArgs) Handles Button5.Click
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/humidity")
        Dim dato As String = lectura
        TextBox3.Text = dato
    End Sub
End Class
```

Por otro lado, el código del sketch en Arduino es:

```
#include <SPI.h>
#include <Ethernet.h>
#include <aREST.h>

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x0A, 0xC2 };

IPAddress ip(192,168,1,150);

EthernetServer server(80);

aREST rest = aREST();
```

```
int temperature;
int humidity;

void setup(void)
{
  Serial.begin(115200);

  temperature = 24;
  humidity = 40;

  rest.variable("temperature",&temperature);
  rest.variable("humidity",&humidity);
  rest.set_id("001");
  rest.set_name("ASP");

    if (Ethernet.begin(mac) == 0) {
    Serial.println("Fallo servidor DHCP");
        Ethernet.begin(mac, ip);
  }
  server.begin();
  Serial.print("IP:");
  Serial.println(Ethernet.localIP());

}
void loop() {

  EthernetClient client = server.available();
  rest.handle(client);

}
```

La figura 9.26 muestra cómo se ve la aplicación en pantalla.

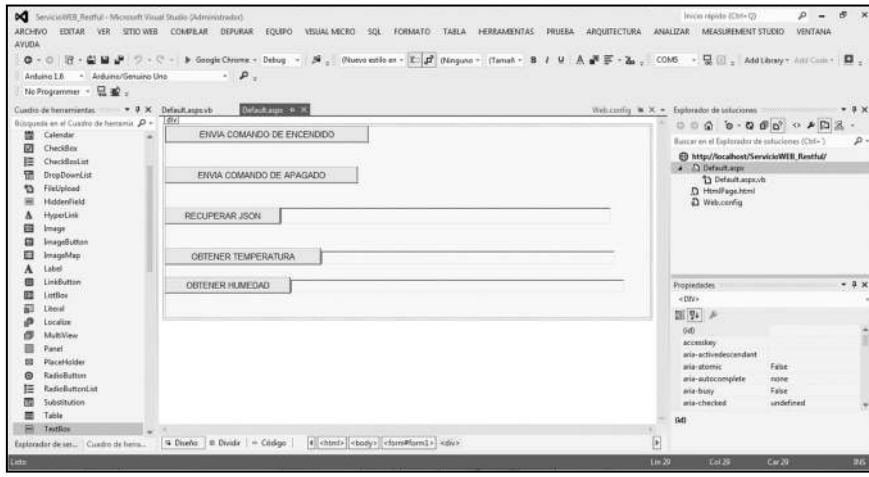


Figura 9.26 Pantalla de la aplicación RESTful en ASP.NET.

Asimismo, la figura 9.27 muestra cómo se ve en pantalla la ejecución de la aplicación.

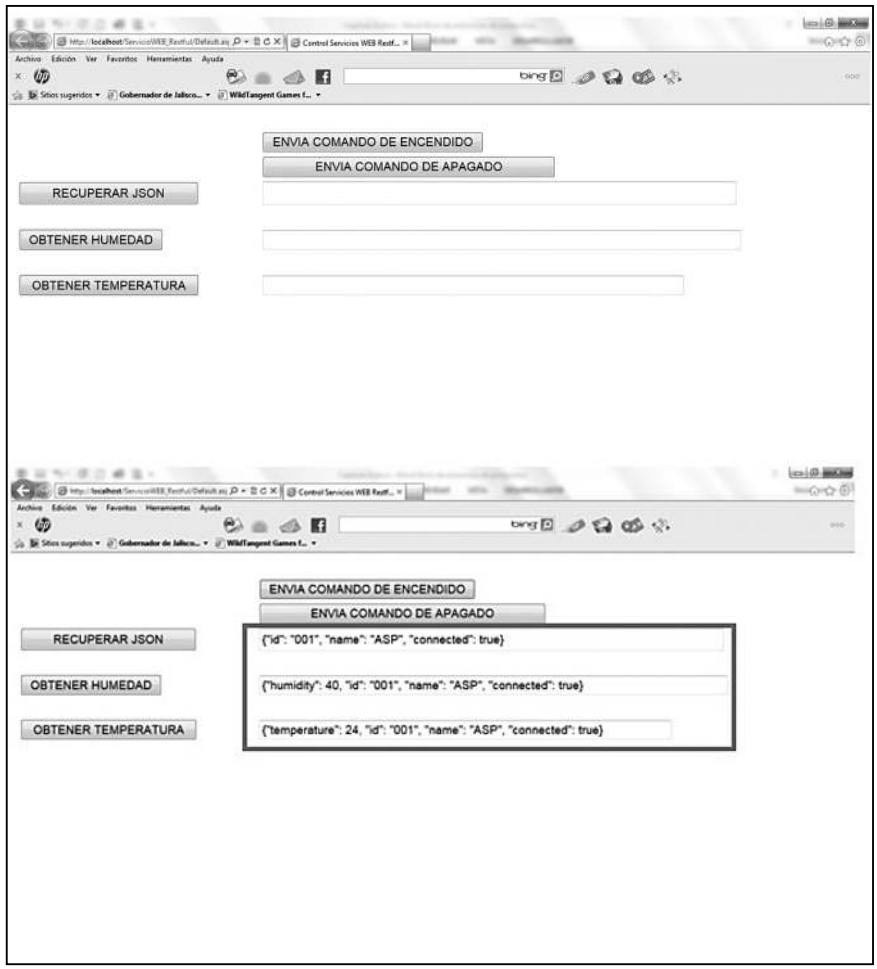


Figura 9.27 Pantallas de la aplicación en ejecución.

9.4.5 Monitoreo de un sensor de flujo de agua desde una página web en ASP.NET

En este ejemplo se describirá cómo monitorear el sensor de flujo de agua desde una página web utilizando el API de Arduino aREST a través del envío de comandos a la tarjeta Arduino y haciendo peticiones a la misma placa. La gran ventaja que esto presenta es que desde una misma página web se pueden monitorear varios dispositivos conectados. De tal manera, se puede tener un sitio de monitoreo dónde observar el comportamiento de cada dispositivo interconectado.

Para llevar a cabo este proyecto se requiere de un sensor de flujo de agua (ver figura 9.28), una tarjeta Arduino UNO, el Ethernet Shield y cables para realizar las conexiones.

Figura 9.28 Sensor de flujo de agua.

Conexiones de hardware

El sensor de flujo tiene tres pines de conexión, el pin de color rojo se conecta a +VCC (+5 volts), el pin de color negro se conecta a tierra (GND) y el pin color amarillo, la señal de datos, se conectará al pin 2 de la Arduino. (Ver figura 9.29.)

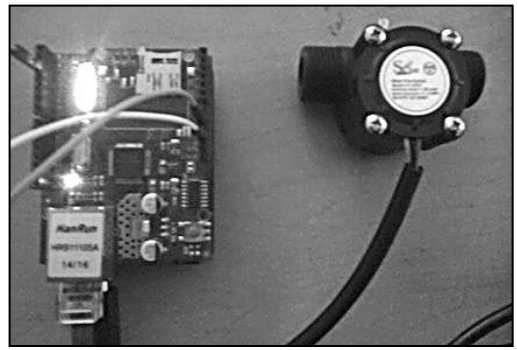


Figura 9.29 Conexión del sensor con la Arduino.

Lectura de la señal del sensor

Para leer el sensor se debe llevar a cabo lo siguiente:

1) Utilizar una interrupción de la siguiente manera para que los pulsos generados por el paso del agua sean contabilizados:

```
attachInterrupt(0, count_pulse, RISING);
```

La interrupción es de tipo RISING y contabiliza los pulsos que pasan de un estado bajo a uno alto.

2) Establecer la función para el conteo de los pulsos:

```
void count_pulse()
{
    pulse++;
}
```

3) Publicar las variables que se desean monitorear:

```
rest.variable("pulso",&pulse);
rest.variable("flujo",&flow_rate);
rest.variable("volumen",&volume);
```

4) Realizar las operaciones para obtener los datos:

```
flow_rate = pulse * 1000/pulses_per_litre;
volume = volume + flow_rate * 0.1;
```

5) Insertar el código en Arduino para activar el conteo de pulsos por segundo. (Ver figura 9.30):

```
int pin = 2;
volatile unsigned int pulse;
const int pulses_per_litre = 450;

void setup()
{
   Serial.begin(9600);

   pinMode(pin, INPUT);
   attachInterrupt(0, count_pulse, RISING);
}

void loop()
{
    pulse=0;
    interrupts();
    delay(1000);
    noInterrupts();

    Serial.print("Pulsos por segundo: ");
    Serial.println(pulse);
}

void count_pulse()
{
   pulse++;
}
```

Figura 9.30 Pantalla del conteo de los pulsos por segundo.

6) Calcular la tasa de flujo de agua según los pulsos contabilizados. Para ello, se inserta en la Arduino el siguiente código:

```
int pin = 2;
volatile unsigned int pulse;
const int pulses_per_litre = 450;


void setup()
{
     Serial.begin(9600);

     pinMode(pin, INPUT);
     attachInterrupt(0, count_pulse, RISING);
}

void loop()
{
    pulse = 0;

    interrupts();
    delay(100);
    noInterrupts();

    Serial.print("Pulsos por segundo: ");
    Serial.println(pulse);

    Serial.print("Flujo de Agua: ");
    Serial.print(pulse * 1000/pulses_per_litre);
    Serial.println("mililitros por segundo");
    delay(10000);
```

```
}

void count_pulse()
{
    pulse++;
}
```

Los resultados deben aparecer en pantalla como se muestra en la figura 9.31.

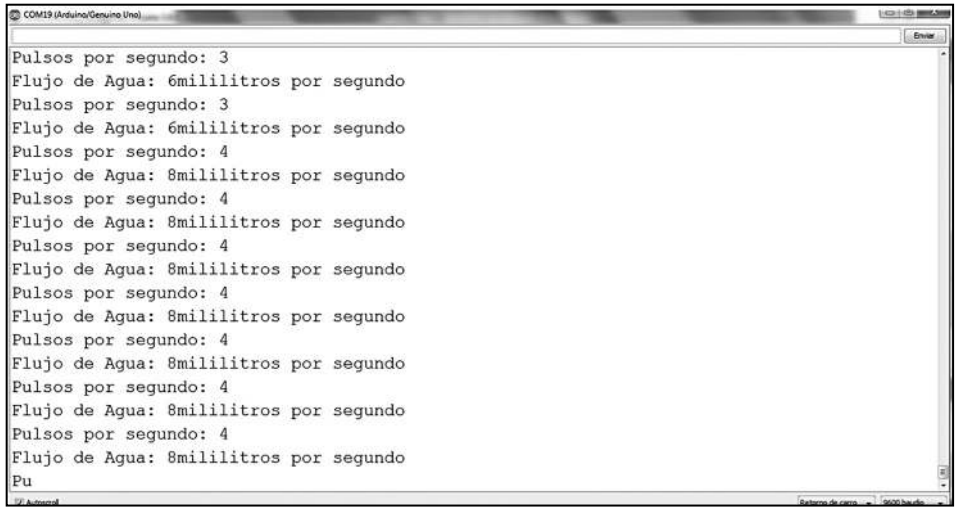


Figura 9.31 Resultados en pantalla del conteo de pulsos por segundo.

7) Calcular el flujo y el volumen de agua utilizando el siguiente código en Arduino:

```
int pin = 2;
volatile unsigned int pulse;
float volume = 0;
float flow_rate = 0;
const int pulses_per_litre = 450;

void setup()
{
     Serial.begin(9600);
     pinMode(pin, INPUT);
     attachInterrupt(0, count_pulse, RISING);
}
void loop()
{
    pulse=0;
     interrupts();
    delay(100);
    noInterrupts();

    Serial.print("Pulsos por segundo: ");
```

```
    Serial.println(pulse);

    flow_rate = pulse * 1000/pulses_per_litre;
     Serial.print("Flujo del Agua: ");
    Serial.print(flow_rate);
    Serial.println(" mililitros por segundo");

    volume = volume + flow_rate * 0.1;
    Serial.print("Volumen: ");
    Serial.print(volume);
    Serial.println(" mililitros");
    delay(10000);
}

void count_pulse()
{
    pulse++;
}
```

Los resultados en pantalla deben verse como lo muestra la figura 9.32.

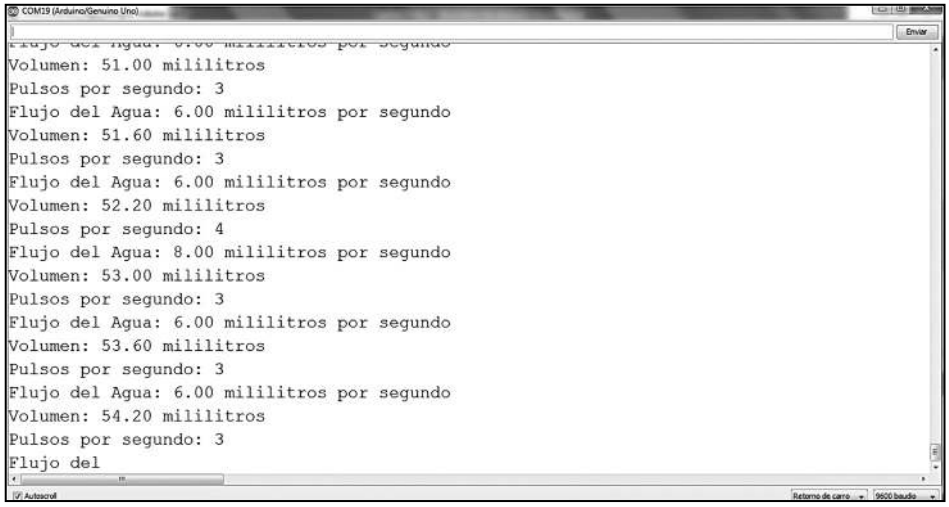


Figura 9.32 Resultado en pantalla de la medición del flujo y del volumen de agua.

8) Establecer el código completo de la aplicación en Arduino:

```
#include <SPI.h>
#include <Ethernet.h>
#include <aREST.h>

byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x0A, 0xC2};

IPAddress ip(192,168,1,150);
```

```
EthernetServer server(80);


aREST rest = aREST();

int pin = 2;
int pulse;
float volume = 0;
float flow_rate = 0;
const int pulses_per_litre = 450;

void setup(void)
{
  Serial.begin(115200);
  pinMode(pin, INPUT);
  attachInterrupt(0, count_pulse, RISING);

  rest.variable("pulso",&pulse);
  rest.variable("flujo",&flow_rate);
  rest.variable("volumen",&volume);

  rest.set_id("001");
  rest.set_name("Sensor");

  if (Ethernet.begin(mac) == 0) {
  Serial.println("Fallo la configuración con el servidor
DHCP");
  Ethernet.begin(mac, ip);
  }
  server.begin();
  Serial.print("IP: ");
  Serial.println(Ethernet.localIP());

}
void loop(){
pulse = 0;
interrupts();
delay(100);
noInterrupts();

flow_rate = pulse * 1000/pulses_per_litre;
volume = volume + flow_rate * 0.1;

EthernetClient client = server.available();
rest.handle(client);
}


void count_pulse()
{
    pulse++;
}
```

Envío de comandos aREST desde el navegador

Es importante resaltar que la comunicación con la tarjeta Arduino se hace a través de los servicios RESTful. Es decir, se envían los comandos desde el navegador a la tarjeta. La respuesta se obtiene en el formato de JSON. (Ver figura 9.33.)

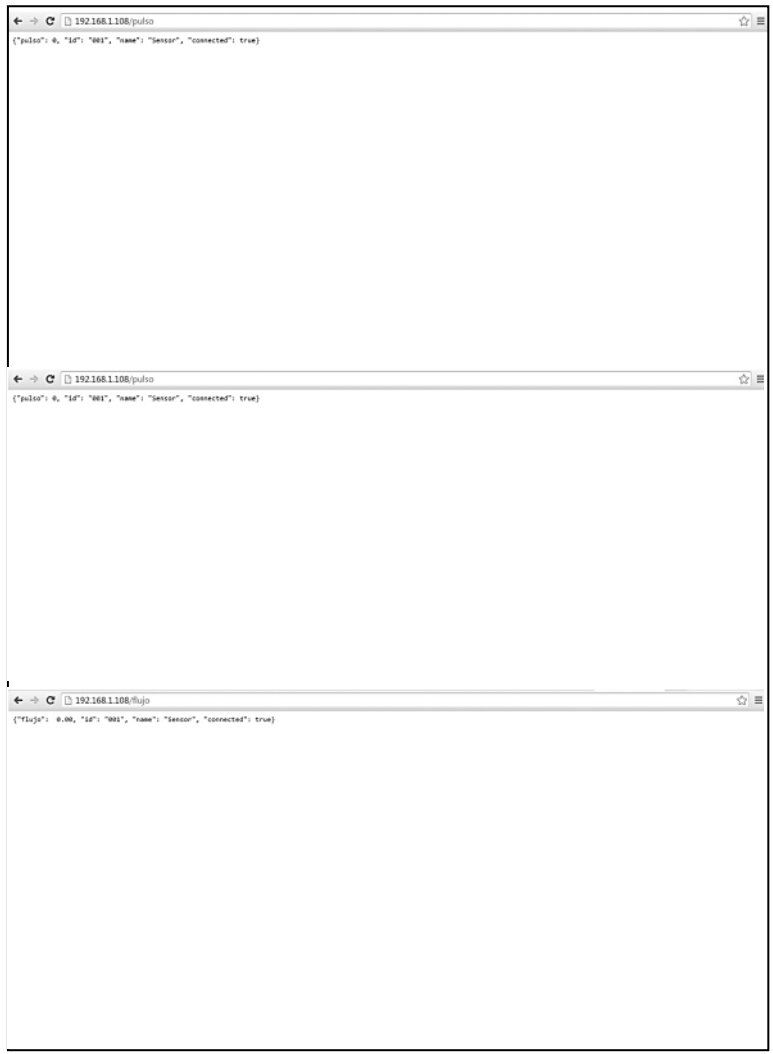


Figura 9.33 Comunicación entre Arduino y RESTful en formato JSON.

De acuerdo con la figura 9.33, la respuesta del comando enviado a la tarjeta se realiza a través de la dirección IP de la tarjeta Arduino en formato JSON, un estándar de servicios web:

```
http://192.168.1.108/flujo
```

Una vez que se ha establecido la comunicación, hay que implementar el llamado de la respuesta desde .NET a través de la clase. El formato de instrucción es:

```
myWebClient.DownloadString("http://192.168.1.108/pulso")
```

Posteriormente, se debe verificar la dirección IP asignada a la Arduino para que se pueda comunicar con la tarjeta.

Aplicación en ASP.NET para monitorear el sensor del flujo de agua

Para inicializar la aplicación, en la pantalla de page load de la página se colocan los siguientes códigos:

```
Protected Sub Page_Load(sender As Object, e As EventArgs)
Handles Me.Load

        Dim cadena As String
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/pulso")
        Dim dato As String = lectura
        cadena = Mid(dato, 11, 1)
        TextBox2.Text = cadena

        Dim cadena1 As String
        Dim myWebClient1 As New WebClient()
        Dim lectura1 As String = myWebClient1.DownloadS-
tring("http://192.168.1.108/flujo")
        Dim dato1 As String = lectura1
        cadena1 = Mid(dato1, 11, 4)
        TextBox3.Text = cadena1


        Dim cadena2 As String
        Dim myWebClient2 As New WebClient()
        Dim lectura2 As String = myWebClient2.DownloadS-
tring("http://192.168.1.108/volumen")
        Dim dato2 As String = lectura2
        cadena2 = Mid(dato2, 12, 6)
        TextBox4.Text = cadena2

    End Sub
```

Para que la aplicación se actualice y los datos se modifiquen en pantalla se requiere el uso de la extensión AJAX. En la pantalla se muestra un Script Manager, un UpdatePanel y el Timer; esto permitirá que únicamente los datos solicitados a la tarjeta Arduino se actualicen de forma automática, en lugar de toda la página. Ésa es una de las características específicas de este proyecto, además de la comunicación con Arduino a través del módulo Ethernet. (Ver figura 9.34.)

Figura 9.34 Gracias a las funciones de Script Manager, UpdatePanel y Timer se pueden actualizar únicamente dichos datos en la página web.

El código de la función AJAX para la actualización de los datos es:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFi-
le="Default.aspx.vb" Inherits="_Default" %>
<%@ Import Namespace="System.Net" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; char-
set=utf-8"/>

    <title>Monitoreo de Sensores</title>
</head>
<body>
    <form id="form1" runat="server">
    <div style="height: 454px">
        <asp:ScriptManager ID="ScriptManager1" runat="-
server"></asp:ScriptManager>

        <asp:UpdatePanel ID="UpdatePanel1" runat="ser-
ver">
```

```
            <ContentTemplate>

                <asp:Timer ID="Timer1" runat="server" In-
terval="1000"></asp:Timer>

                <br />
                <asp:Label ID="Label1" runat="server"
Font-Bold="True" style="font-size: xx-large" Text="Página
de Monitoreo de Sensor de Flujo"></asp:Label>
                <br />
                <br />
                <asp:Label ID="Label2" runat="server"
Font-Bold="True" style="font-size: xx-large" Text="Utili-
zando ASP .NET y REST API"></asp:Label>
                <br />
                <br />
                <br />

               <%Dim cadena As String
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/pulso")
        Dim dato As String = lectura
        cadena = Mid(dato, 11, 1)
        TextBox2.Text = cadena

        Dim cadena1 As String
        Dim myWebClient1 As New WebClient()
        Dim lectura1 As String = myWebClient1.DownloadS-
tring("http://192.168.1.108/flujo")
        Dim dato1 As String = lectura1
        cadena1 = Mid(dato1, 11, 4)
        TextBox3.Text = cadena1

        Dim cadena2 As String
        Dim myWebClient2 As New WebClient()
        Dim lectura2 As String = myWebClient2.DownloadS-
tring("http://192.168.1.108/volumen")
        Dim dato2 As String = lectura2
        cadena2 = Mid(dato2, 12, 6)
        TextBox4.Text = cadena2%>

                <asp:Button ID="Button2" runat="server"
Height="53px" Text="Pulsos por segundo" Width="260px" />
                <asp:TextBox ID="TextBox2" runat="server"
Width="45px"></asp:TextBox>
                <br />
                <br />
                <asp:Button ID="Button3" runat="server"
Height="49px" Text="Flujo de Agua en mililitros por se-
gundo" Width="260px" />
                <asp:TextBox ID="TextBox3" runat="server"
Width="72px"></asp:TextBox>
```

```
                <br />
                <br />
                <asp:Button ID="Button4" runat="server"
Height="70px" Text="Volumen en mililitros" Width="263px"
/>
                <asp:TextBox ID="TextBox4" runat="server"
Width="74px"></asp:TextBox>

                <br />
                <br />
            </ContentTemplate>
         </asp:UpdatePanel>

  
        <br />
        <br />

        <br />

        <br />

   </div>
    </form>
</body>
</html>
```

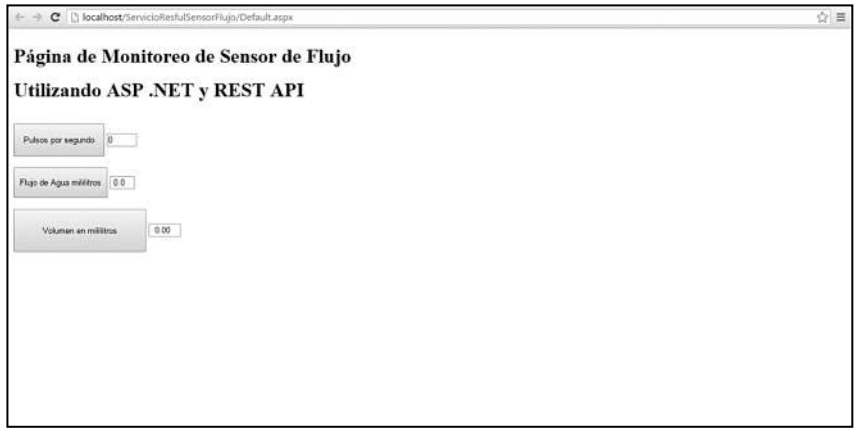
Ahora bien, la aplicación en ASP.NET tiene tres botones. Al hacer clic en cada uno aparece cierta información que realiza una petición a la tarjeta a través de la ejecución del evento del botón, como se puede ver en los siguientes códigos:

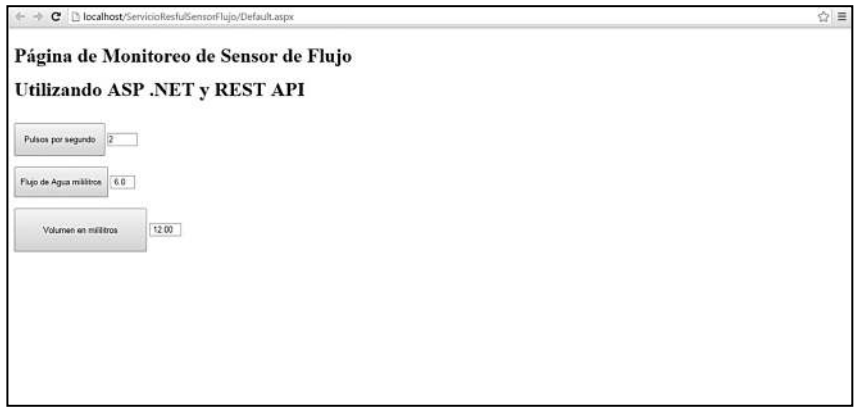
```
Protected Sub Button2_Click(sender As Object, e As Even-
tArgs) Handles Button2.Click
        Dim cadena As String
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/pulso")
        Dim dato As String = lectura
        cadena = Mid(dato, 11, 1)
        TextBox2.Text = cadena
    End Sub


    Protected Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
        Dim cadena As String
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/flujo")
        Dim dato As String = lectura
        cadena = Mid(dato, 11, 4)
        TextBox3.Text = cadena
    End Sub
```

```
Protected Sub Button4_Click(sender As Object, e As Even-
tArgs) Handles Button4.Click
        Dim cadena As String
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.1.108/volumen")
        Dim dato As String = lectura
        cadena = Mid(dato, 12, 6)
        TextBox4.Text = cadena
    End Sub
```

La figura 9.35 muestra cómo se ve la ejecución de la aplicación.





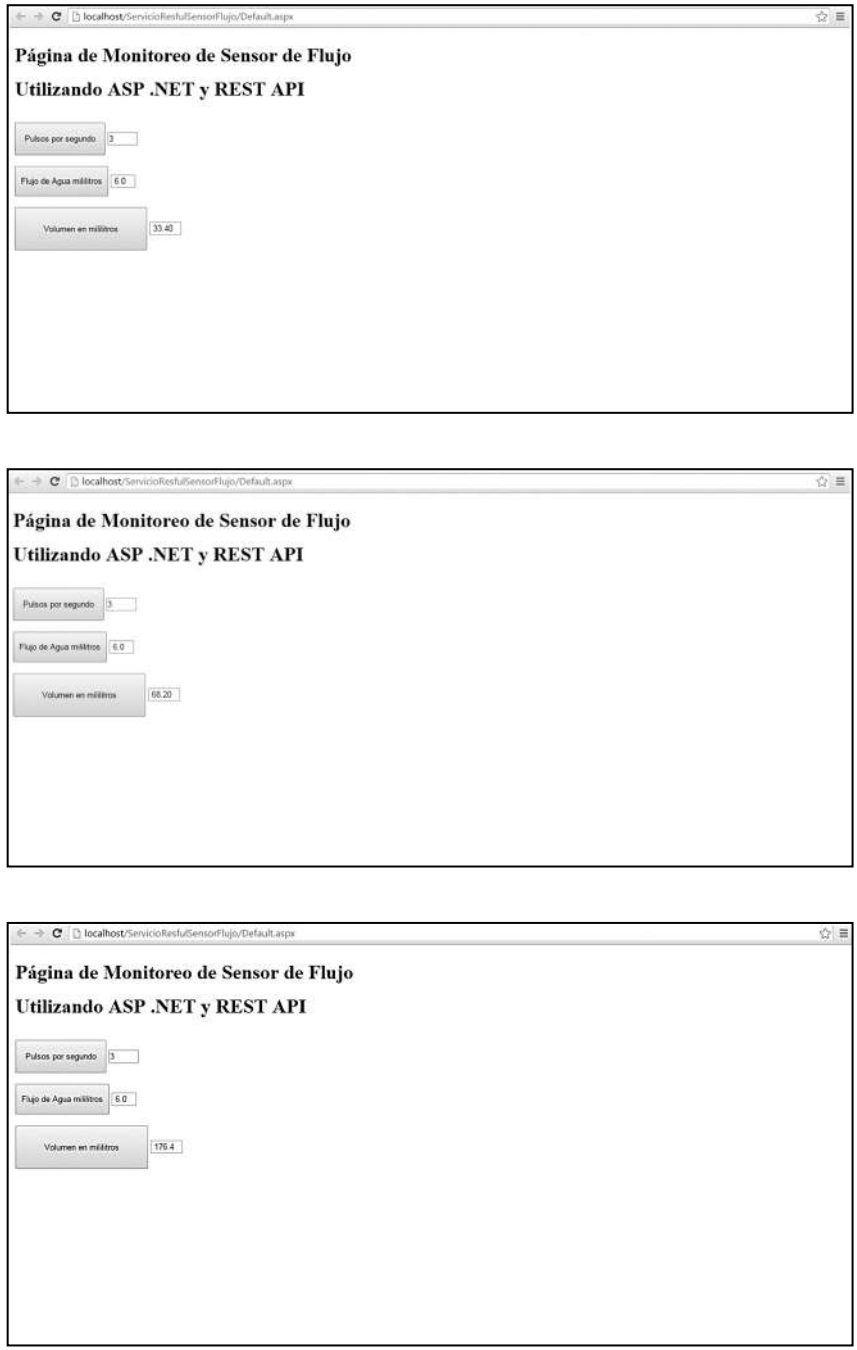


Figura 9.35 Muestra de cómo se ve en pantalla la ejecución de esta aplicación que monitorea el flujo y volumen de agua desde una página web.

9.4.6 Registro de datos en tiempo real de un panel solar a través de servicios web en la nube

En esta sección se explicará cómo leer los registros de un panel solar a través de un servicio web en la nube (cloud). Para realizar este proyecto se requiere lo siguiente:

- Tarjeta Arduino UNO
- Módulo Ethernet Shield
- Panel solar de 12 voltios. (Ver figura 9.36.)

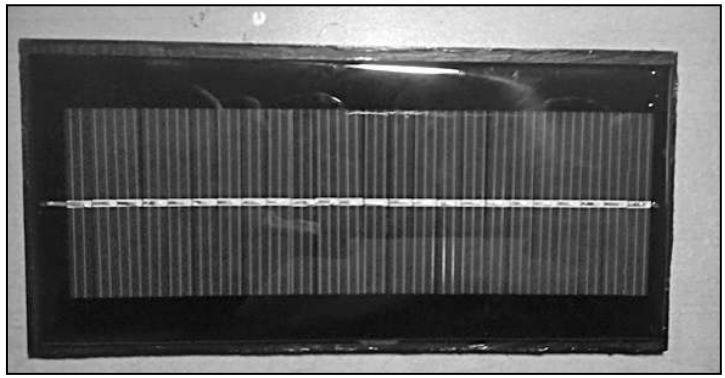


Figura 9.36 Panel solar de 12 voltios.

Para conectarnos con el servicio en la nube se debe a entrar a la siguiente dirección. (Ver figura 9.37):

```
https://dweet.io/
```

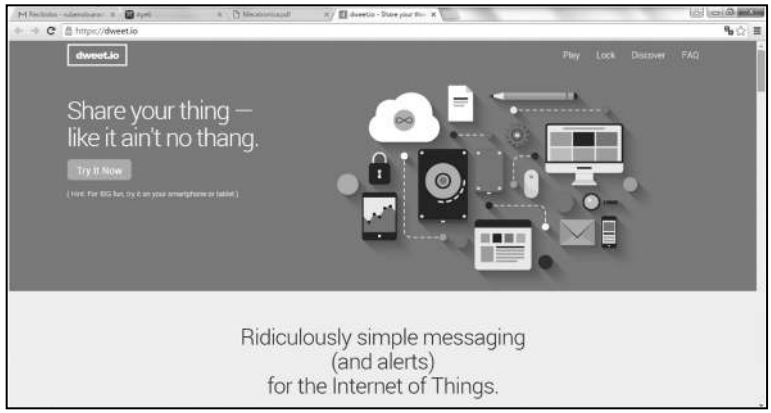


Figura 9.37 Conexión del servidor a la nube.

No es necesario tener una cuenta registrada en este servicio, solamente es necesario considerar las siguientes configuraciones para el registro de datos:

1) Entrar en la siguiente dirección para publicar un dato:

```
https://dweet.io/dweet/for/my-thing-name?hello=world
```

2) Remplazar el nombre de my.thing-name por el nombre de nuestro método. Se obtiene el último registro entrando a:

```
https://dweet.io/get/latest/dweet/for/my-thing-name
```

3) Se obtienen los últimos 500 registros en:

```
https://dweet.io/get/dweets/for/my-thing-name
```

4) El acceso a dweet desde Arduino se hace con la siguiente instrucción:

```
"POST /dweet/for/panelsolar?A0="
```

5) Se hace una prueba con la siguiente dirección para ver los datos. (Ver figura 9.38):

```
https://dweet.io/get/latest/for/panelsolar
```

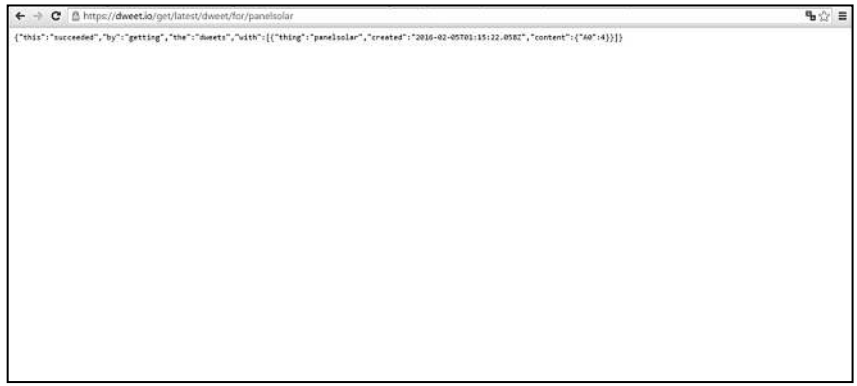


Figura 9.38 Realizando la prueba del panel solar desde la nube.

6) Se puede monitorear la señal del panel desde dweet con la dirección. (Ver figura 9.39):

```
https://dweet.io/follow/panelsolar
```

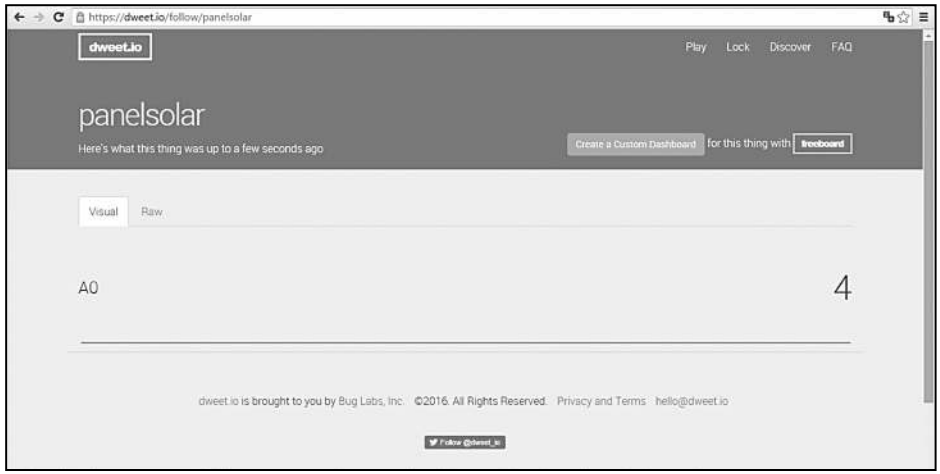


Figura 9.39 Lectura del panel solar desde la nube.

La figura 9.40 muestra cómo se ve en pantalla el registro de todos los datos enviados a dweet.

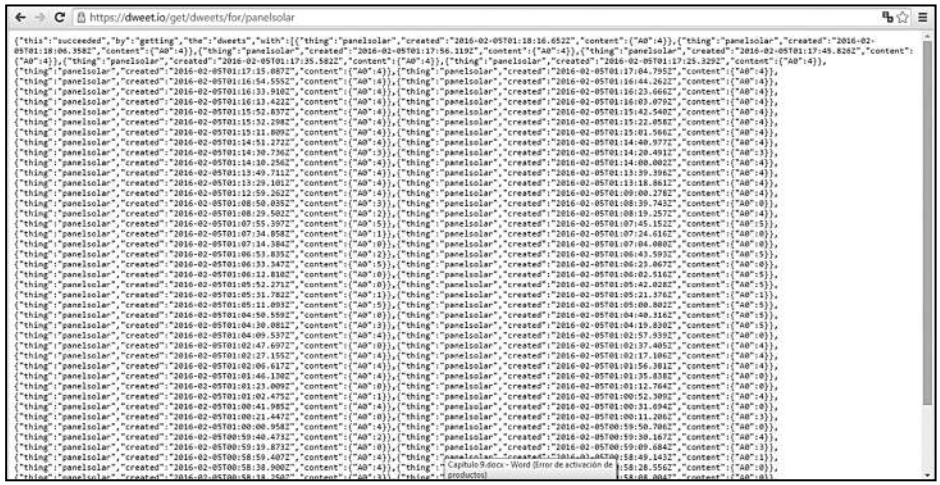


Figura 9.40 Pantalla de registros del panel solar en dweet.

Para aumentar la complejidad de esta aplicación se puede crear un dashboard (panel de control) en Freeboard. Los pasos son los siguientes:

1) Entrar a la página. (Ver figura 9.41):

```
https://freeboard.io/
```

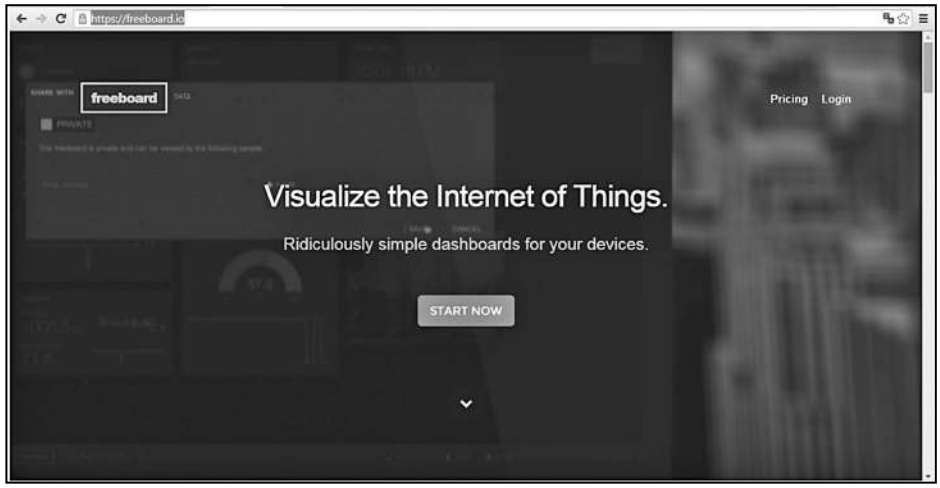


Figura 9.41 Entrando a Freeboard para crear el dashboard del panel solar.

2) Hacer clic en Start now. Enseguida aparecerá una pantalla pidiendo que se cree una cuenta para tener acceso. (Ver figura 9.42.)

Figura 9.42 Página para crear una cuenta del dashboard.

3) Una vez que se tiene acceso, aparece una pantalla que permitirá monitorear el panel. (Ver figura 9.43.)

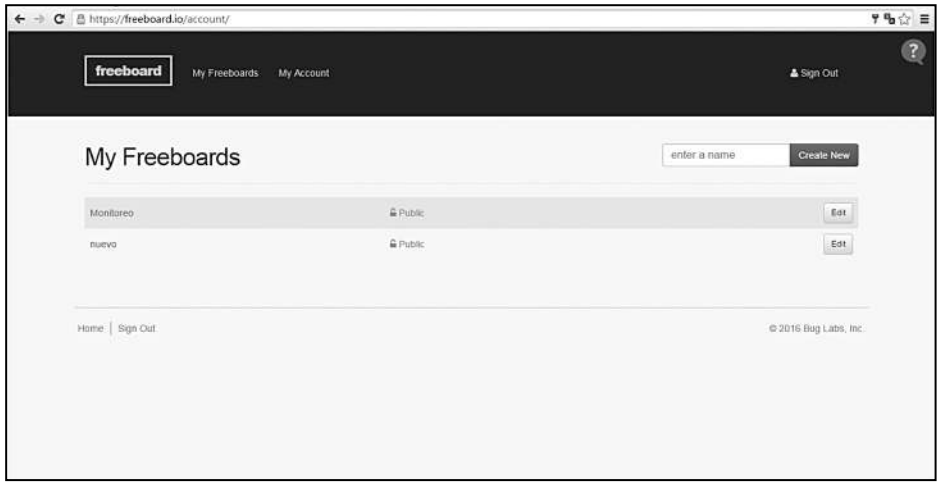


Figura 9.43 Pantalla para configurar el monitoreo.

4) En esa misma pantalla se debe hacer clic en el link Add (agregar). La figura 9.44 muestra lo que aparecerá en pantalla.

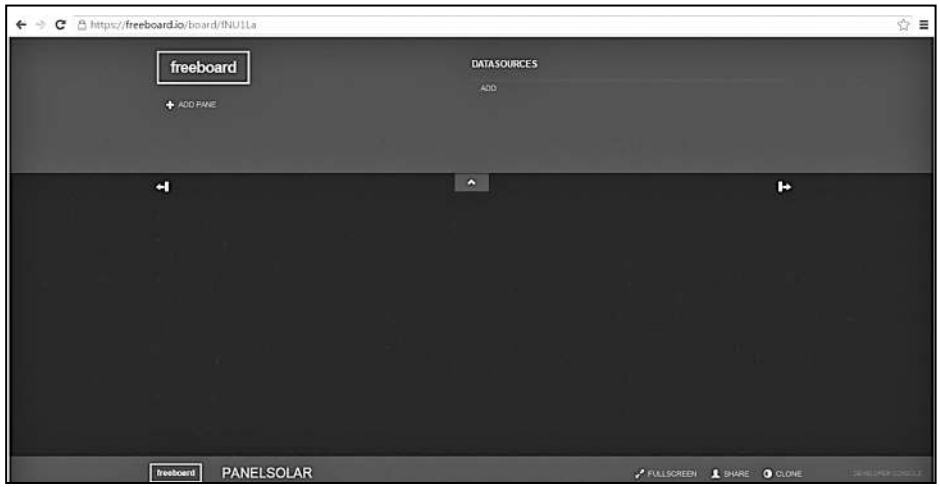


Figura 9.44 Configuración del dashboard del panel solar.

5) Posteriormente, hay que seleccionar Dweet.io, como se ve en la figura 9.45.

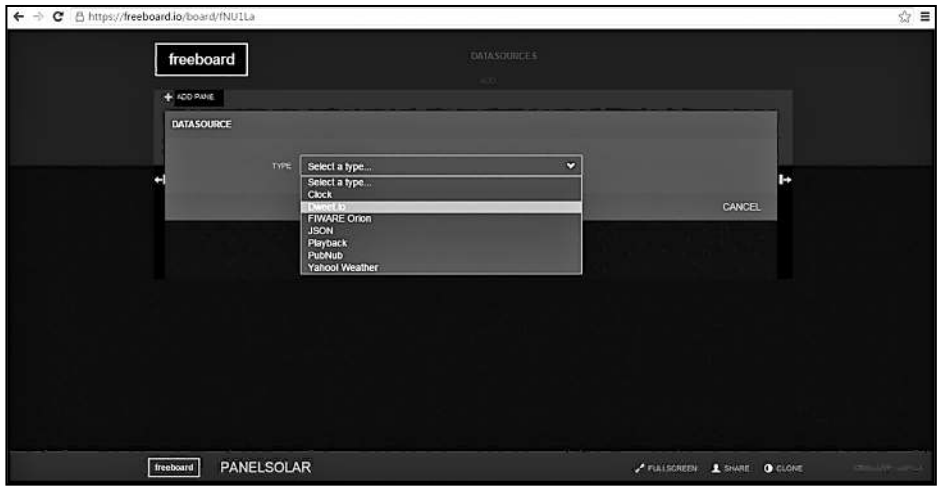


Figura 9.45 Seleccionando la fuente de los datos (dweet.io).

6) Después se deben capturar los datos que se piden; es decir, el nombre asignado y el nombre del objeto. (Ver figura 9.46):

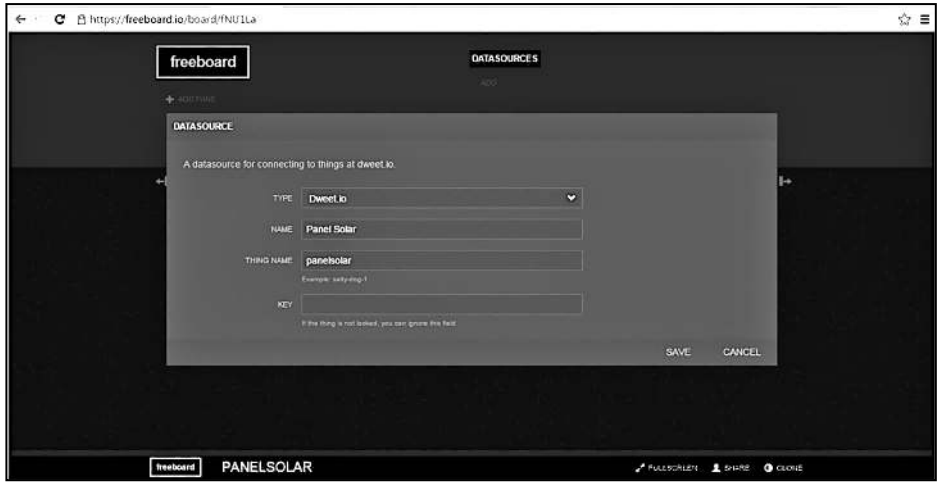


Figura 9.46 Capturando los datos del panel solar.

7) Hay que crear un gráfico para que el dashboard muestre el valor del panel solar. Para ello, hacer clic en Add Pane y enseguida dar clic en el botón del signo de más (+). (Ver figura 9.47.)

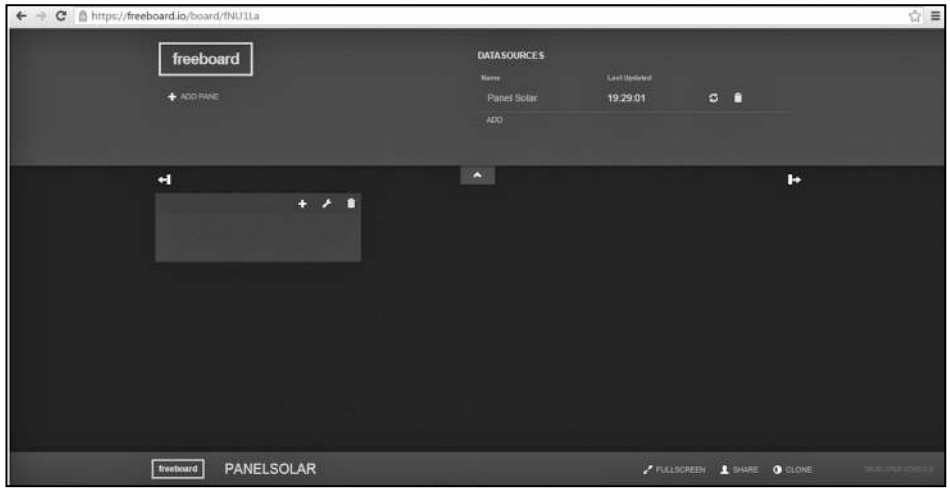


Figura 9.47 Creando gráfico del panel solar.

8) Luego, seleccionar Gauge, como se muestra en la figura 9.48.

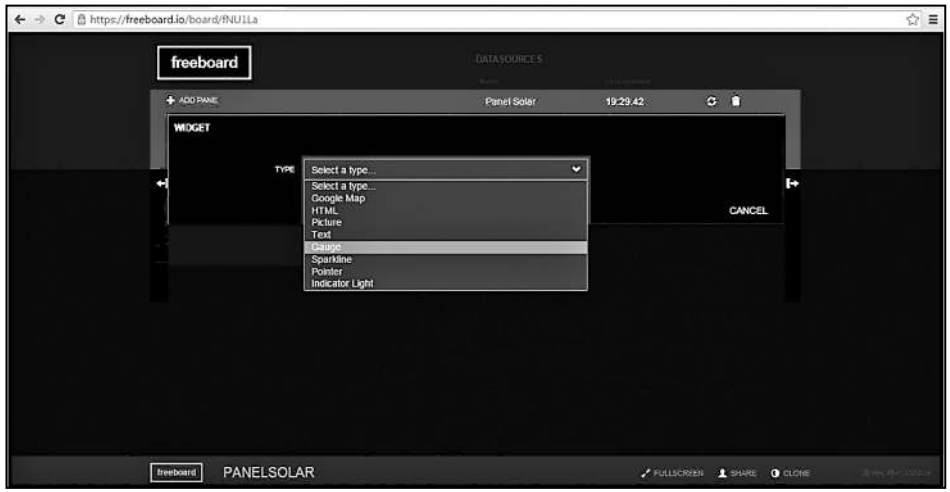


Figura 9.48 Seleccionando un medidor (gauge) para el dashboard.

9) Posteriormente, hay que capturar la información que nos piden: TITLE (título del objeto), VALUE (valor-la fuente de los datos), UNITS (unidad de medida; en este caso, voltios), MINIMUM (mínimo) y MAXIMUM (máximo; en este caso, 12). (Ver figura 9.49.)

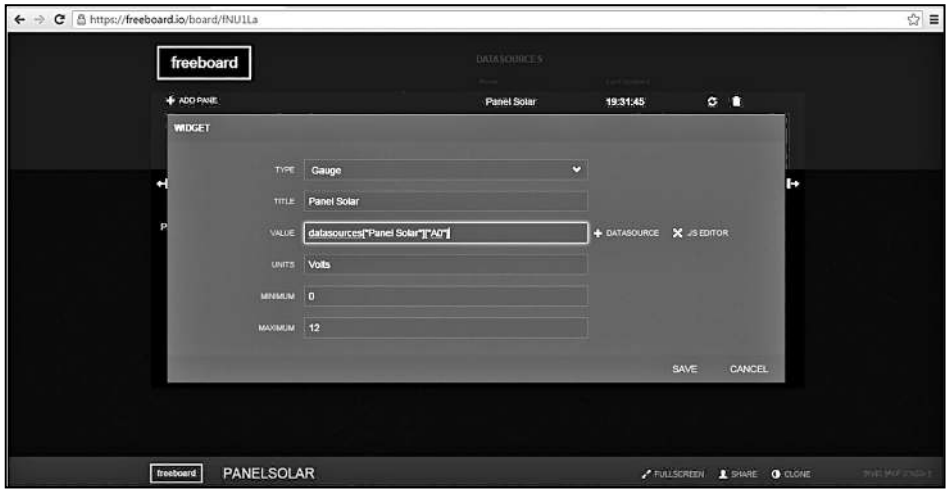


Figura 9.49 Capturando los datos del widget que se quiere en el dashboard.

Ahora se puede ver el valor real que el sistema está leyendo. (Ver figura 9.50):

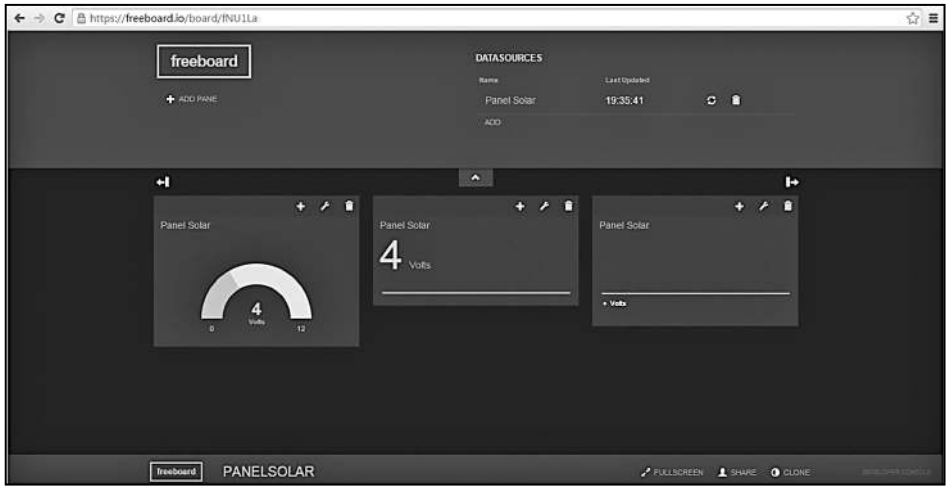


Figura 9.50 Dashboard con los datos del panel solar.

Incluso se puede cotejar si la información del dashboard es correcta utilizando un multímetro, como se aprecia en la figura 9.51.

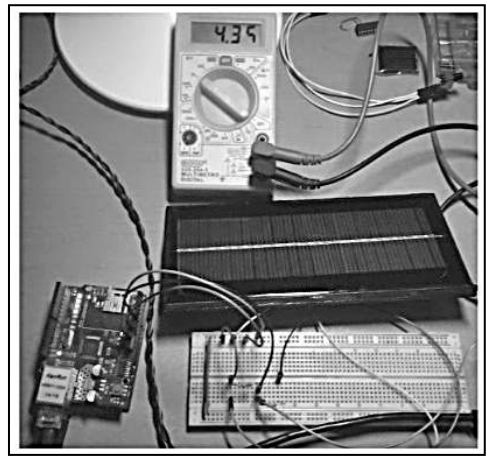


Figura 9.51 Los datos del multímetro coinciden con los datos del dashboard.

El código de la aplicación es el siguiente:

```
#include <SPI.h>
#include <Ethernet.h>

const int analogIn = A0;
int analogVal = 0;
float volts;

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,177);

// initialize the library instance:
EthernetClient client;

char server[] = "www.dweet.io";

unsigned long lastConnectionTime = 0;              //
last time you connected to the server, in milliseconds
const unsigned long postingInterval = 10L * 1000L; // de-
lay between updates, in milliseconds
// the "L" is needed to use long type numbers

void setup() {
  // start serial port:
  Serial.begin(9600);
  Serial.println("--- Start ---");

  // give the ethernet module time to boot up:
  delay(1000);
  // start the Ethernet connection using a fixed IP ad-
dress and DNS server:
```

```
  Ethernet.begin(mac, ip);
  // print the Ethernet board/shield's IP address:
  Serial.print("My IP address: ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // if there's incoming data from the net connection.
  // send it out the serial port.  This is for debugging
  // purposes only:
  if (client.available()) {
    char c = client.read();
    Serial.write(c);
  }

  // if ten seconds have passed since your last connec-
tion,
  // then connect again and send data:
  if (millis() - lastConnectionTime > postingInterval) {
    httpRequest();
  }

}
// this method makes a HTTP connection to the server:
void httpRequest() {
  // close any connection before send a new request.
  // This will free the socket on the WiFi shield
  client.stop();

  // if there's a successful connection:
  if (client.connect(server, 80)) {
    Serial.println("connected");
    analogVal = analogRead(analogIn);
    volts = (analogVal * 5) / 1024;

    // Make a HTTP request:
    String s = "POST /dweet/for/panelsolar?A0=";
    s.concat(volts);
    Serial.println(s);
    client.println(s);

    client.println("Host: www.dweet.io");
    client.println("Connection: close");
    client.println();

    // note the time that the connection was made:
    lastConnectionTime = millis();
  }
  else {
    // if you couldn't make a connection:
    Serial.println("connection failed");
  }
}
```

9.4.7 Control de un módulo GSM/GPRS

En esta sección utilizaremos el módulo GSM/GPRS para control con Arduino. El uso de esta tecnología aplicada al Internet de las cosas es otra herramienta que se puede aplicar en los dispositivos de control a distancia. El material que se requiere para realizar este proyecto es el siguiente:

- Tarjeta Arduino UNO
- Módulo GSM/GPRS con antena (ver figura 9.52)
- SIM del proveedor de servicios de telefonía

Figura 9.52 Módulo GSM/GPRS para Arduino con módem SIM900.

La figura 9.53 muestra las partes del módulo. Más abajo se encuentra la explicación de cada elemento.

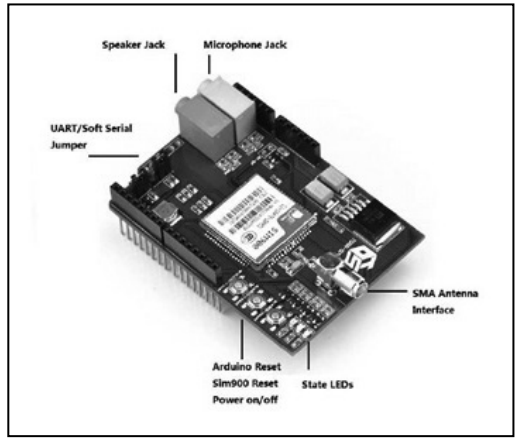


Figura 9.53 Elementos del módulo GSM/GPRS.

1) Indicadores led (State LEDs)

- Encendido de módulo GSM: indica que el módulo está encendido y que está recibiendo energía de la tarjeta Arduino.
- Encendido de módem SIM900: este led permanece prendido cuando el módem está encendido.
- Encendido del status de la red: indica que existe comunicación con la red GSM.

2) Botones: la tarjeta tiene tres botones (Arduino Reset, Sim900 Reset y Power on/off; como se muestra en la figura 9.53), que se describen a continuación:

- Botón de encendido del módem SIM900: sirve para encender el módem. Cabe resaltar que se debe presionar este botón por primera vez para encenderlo e iniciar su configuración. Si se presiona durante un segundo, el módem se apaga. Más aún, se puede encender el módem de forma automática desde el programa de Arduino, activando el pin digital 8 de la tarjeta Arduino.
- Botón de reseteo del módem SIM900: resetea el módem.
- Botón de Reseteo de Arduino: resetea la tarjeta Arduino.

La figura 9.54 muestra la configuración de Jumpers para establecer la comunicación con la tarjeta Arduino:

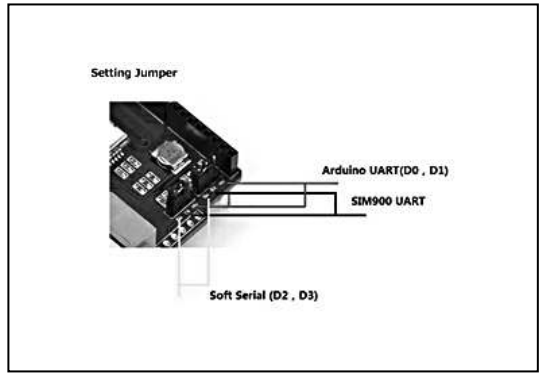


Figura 9.54 Los Jumpers del módem SIM900.

Como se muestra en la figura 9.54, si los jumpers se configuran en modo de comunicación por SW (software) con TX y RX, hay que utilizar los pines digitales 2 y 3 de la Arduino, en los cuales no se conecta nada. En este caso, se utilizará la librería SoftwareSerial para establecer la comunicación serial con Arduino. Por otro lado, si los pines se configuran en el modo de HW (hardware), la comunicación serial entre Arduino y el módulo GSM se realizará mediante los pines digitales 0 y 1. Para fines prácticos, en este proyecto se utilizará el modo de configuración por software. De tal forma, la configuración debe ser tal cual se muestra en la figura 9.54.

Comunicación serial entre el módulo GSM y la tarjeta Arduino

La comunicación entre el módulo y Arduino se debe establecer de forma serial; asimismo, los pasos para configurar la velocidad de comunicación del módulo GSM son:

1) Conectar el módulo a la tarjeta Arduino UNO.
2) Cargar el sketch de prueba para iniciar la comunicación con el módulo.

```
#include <SoftwareSerial.h>
SoftwareSerial GPRS(2,3);
unsigned char buffer[64]; // buffer array for data recie-
ve over serial port
int count=0;     // counter for buffer array

void setup()
{
 GPRS.begin(19200);   // the GPRS baud rate LA PRIMERA
VEZ SE PONE LA VELOCIDAD A 9600 Y CUANDO SE HAGA EL CAM-
BIO SE PONE A 19200
 Serial.begin(19200); // the Serial port of Arduino baud
rate.LA PRIMERA VEZ SE PONE LA VELOCIDAD A 9600 Y CUANDO
SE HAGA EL CAMBIO SE PONE A 19200
}

void loop()
{
 if (GPRS.available())              // if date is comming
from softwareserial port ==> data is comming from gprs
shield
 {
   while(GPRS.available())          // reading data into
char array
   {
     buffer[count++]=GPRS.read();     // writing data
into array
     if(count == 64)break;
 }
   Serial.write(buffer,count);            // if no data
transmission ends, write buffer to hardware serial port
   clearBufferArray();              // call clearBufferA-
rray function to clear the storaged data from the array
   count = 0;                       // set counter of
while loop to zero
 }
 if (Serial.available())            // if data is availa-
ble on hardwareserial port ==> data is comming from PC or
notebook
   GPRS.write(Serial.read());       // write it to the
GPRS shield
}
void clearBufferArray()              // function to clear
buffer array
{
 for (int i=0; i<count;i++)
```

```
   { buffer[i]=NULL;}                    // clear all index
of array with command NULL
}
```

Este sketch permite que desde el monitor Serial de Arduino se puedan enviar los comandos AT al módem SIM900. Se recomienda que este sketch solamente se descargue en Arduino cuando se van a probar los comandos AT, lo cual normalmente se hace al inicio. Asimismo, se recomienda que la velocidad del puerto COM sea 19200 (ver figura 9.55). Es decir, la primera vez que se cargue el sketch, configurar la velocidad a 9600 (ver figura 9.56) para que no aparezcan caracteres raros en el monitor serial. El modulo se debe configurar a una velocidad de 19200 para que el Arduino y el modem SIM900 puedan establecer comunicación.

Para configurar el modulo a una velocidad de 19200, que es como debe de trabajar, es necesario cargarle el sketch y ejecutar los comandos at a una velocidad de 9600.

Es muy importante que la velocidad del sketch sea la misma que se selecciona en el monitor serial.

Figura 9.55 Configuración de la velocidad a 19200 en el puerto COM.

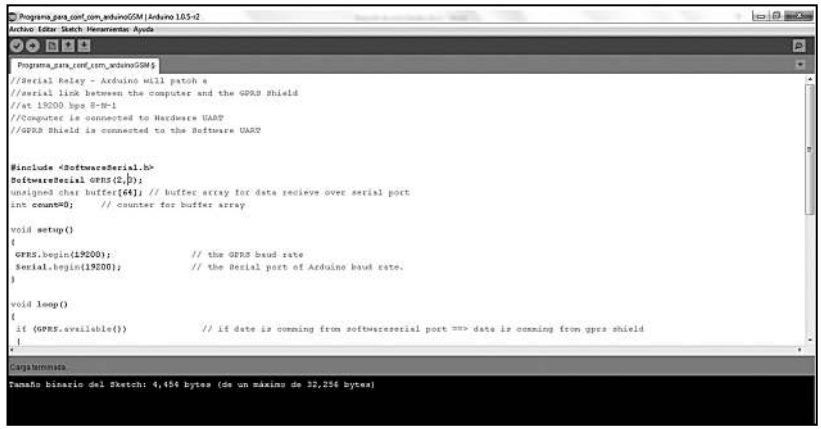


Figura 9.56 Cambiando en el sketch la velocidad a 9600.

3) Verificar el puerto serial por el que se hará la comunicación con el módulo GSM. Luego, presionar el botón de monitor serial, en este caso el COM 11. También, seleccionar retorno de carro y la velocidad a 9600. (Ver figura 9.57.)

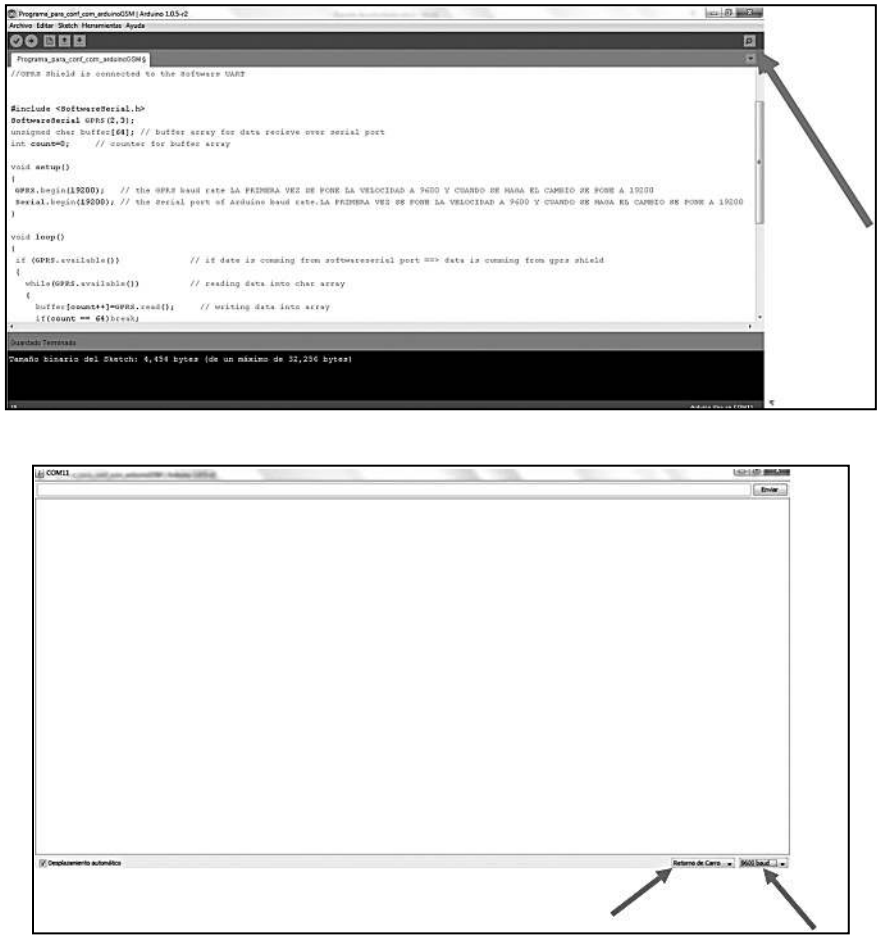


Figura 9.57 Estableciendo el puerto serial y configurando la velocidad y el retorno de carro.

4) Establecer comunicación con el módem mediante comandos AT.
5) Teclear el comando AT en el monitor Serial. (Ver figura 9.58.)

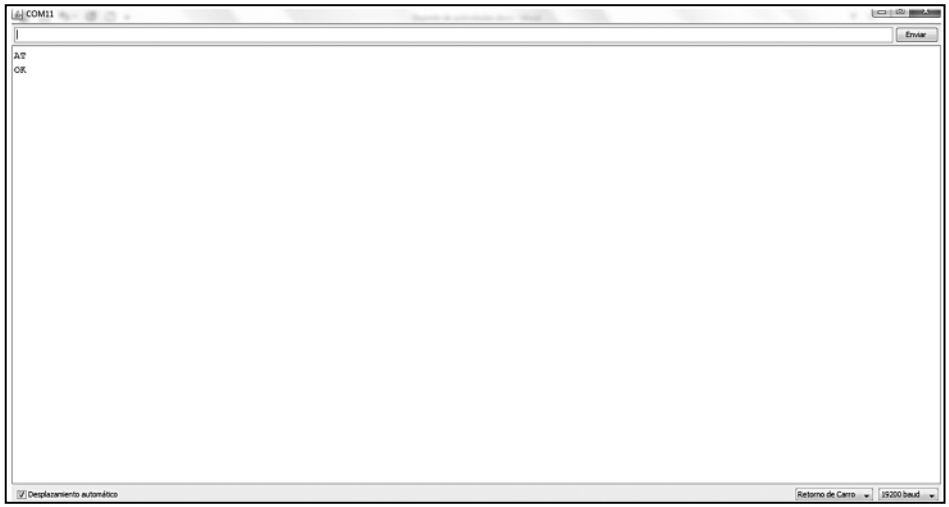


Figura 9.58 Comando AT en el monitor serial.

6) Si aparece la respuesta "OK", significa que ya hay comunicación con el módulo SIM900.

7) Teclear el comando AT+IPR=19200 para cambiar a la velocidad que se requiere para que la tarjeta Arduino se pueda comunicar. (Ver figura 9.59.)

Figura 9.59 Tecleando comando AT+IPR.

8) Teclear el comando AT&W para guardar la configuración realizada. Aquí también debe aparecer la respuesta "OK". (Ver figura 9.60.)

Figura 9.60 Tecleando el comando AT&W.

9) Teclear el comando AT+IPR? para mostrar le velocidad de comunicación configurada previamente. La respuesta del programa será mostrar la velocidad y la palabra "OK". (Ver figura 9.61.)

Figura 9.61 Tecleando el comando AT+IPR?

10) Teclear el comando AT+CPIN? para verificar si el chip SIM está instalado y la red GSM lo reconoce. La respuesta que debería mostrar es CPIN: READY. (Ver figura 9.62.)

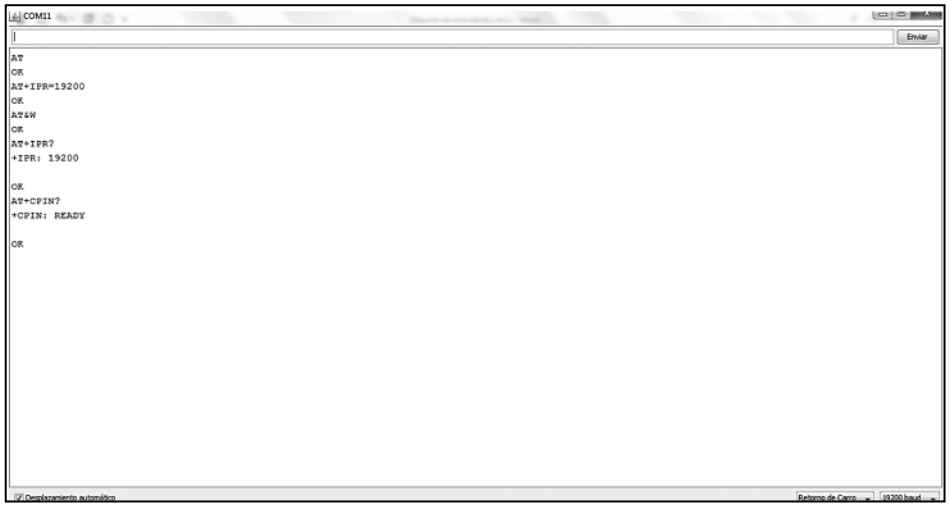


Figura 9.62 Tecleando el comando CPIN: READY.

11) Presionar el botón Reset del shield GSM para verificar la respuesta del módem. Esto es para que indique que ambos equipos, el modem SIM900 y la tarjeta Arduino, se están comunicando entre sí. De tal forma, RDY significa que el módulo está conectado; CFUN: 1, que el módem se reinició; CPIN: READY, que el SIM está insertado correctamente; PACSP: 1, que la red GSM reconoció el dispositivo y CALL Ready, que todo está listo para entablar comunicación. (Ver figura 9.63.)

Figura 9.63 Módem y Arduino están listos para entablar comunicación.

La figura 9.64 muestra que hay comunicación entre ambos dispositivos.

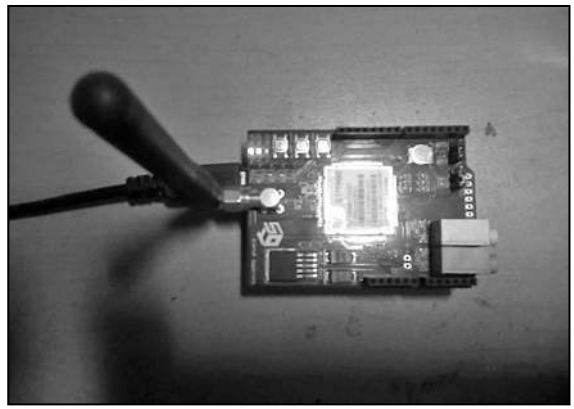


Figura 9.64 Módem y tarjeta Arduino en comunicación.

12) Cargar el sketch adjunto en correo para enviar un mensaje SMS desde el programa de Arduino. Luego se abre el monitor serial y, si se presiona la tecla "t", Arduino enviará un mensaje, en este caso, al teléfono 4772236196. (Ver figura 9.65.)

```
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial mySerial(2,3);

void setup()
{
 mySerial.begin(19200);
 Serial.begin(19200);
 delay(500);
}
void loop()
{
 if (Serial.available())
   switch(Serial.read())
  {
    case 't':
      SendTextMessage();
      break;
    }
 if (mySerial.available())
   Serial.write(mySerial.read());
}
void SendTextMessage()
{
 mySerial.print("AT+CMGF=1\r");
 delay(100);
 mySerial.println("AT+CMGS=\"4772236196\"");
 delay(100);
```

```
 mySerial.println("Hola desde Arduino Mensaje de Prue-
ba!!");
 delay(100);
 mySerial.println((char)26);
 delay(100);
 mySerial.println();
}
```

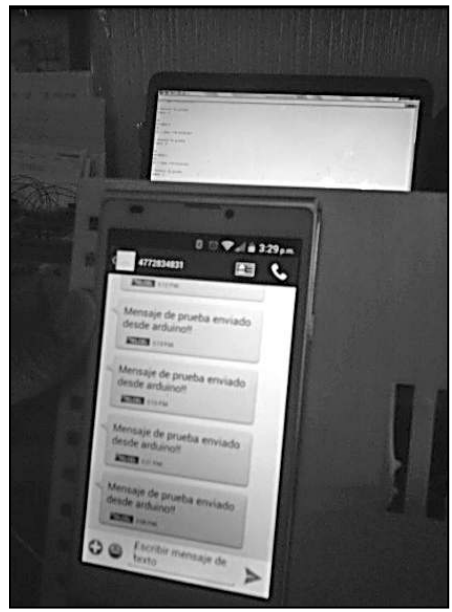


Figura 9.65 Enviando mensaje de prueba al teléfono 4772236196 desde Arduino.

13) Para enviar un mensaje se debe descargar el sketch adjunto cuando se presiona un botón conectado a la entrada 10 digital. Cabe señalar que en este ejemplo el módem se enciende de forma automática desde el programa de Arduino. (Ver figura 9.66.)

```
#include <SoftwareSerial.h>
SoftwareSerial cell(2,3);

const int buttonPin = 10;
int buttonState = 0;

void setup(){

 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2000);
 digitalWrite(8,LOW);
```

```
 delay(3000);

cell.begin(19200);
}
void loop(){
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH){
     delay(2000);
     envioSMS();
    }
}
void envioSMS()
{
    cell.print("AT+CMGF=1\r");
    delay(100);
    cell.println("AT + CMGS = \"4772236196\"");
    delay(100);
    cell.println("Se presiono boton entrada digital 10");
    delay(100);
    cell.println((char)26);
    delay(100);
}
```

Figura 9.66 Enviando mensaje al presionar un botón.

9.4.8 Abrir una chapa al enviar un mensaje de texto SMS

A continuación se describirá cómo establecer la comunicación entre una chapa electrónica, el módulo GSM y Arduino, para abrir la chapa enviando un mensaje SMS. El material para realizar esta práctica es el siguiente:

- Chapa electrónica
- 1 diodo 1N4001
- 1 transistor TIP122
- 1 resistencia de 1kohm
- Eliminador de 12 volts

Se envía un mensaje de texto y se activa el pin número 12 (ver figura 9.67) con el objetivo de que se retraiga la chapa; después de un cierto tiempo, la chapa vuelve nuevamente a su posición de forma automática. La figura 9.68 muestra cómo armar el cableado y cómo se ve el mensaje de texto en el teléfono.

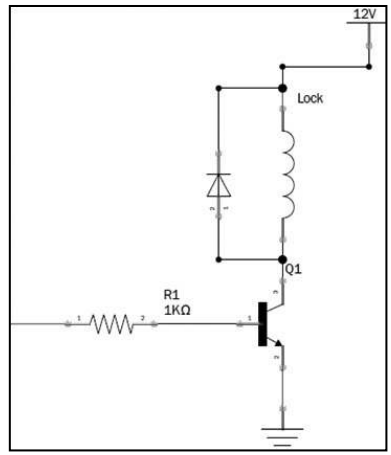
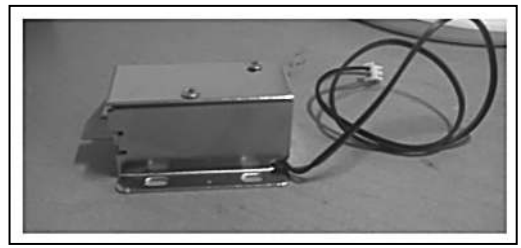


Figura 9.67 Diagrama a seguir para el armado del circuito.



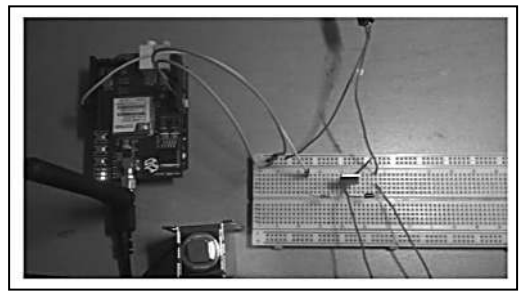


Figura 9.68 Conexiones entre el módulo y la chapa para moverla al enviar un mensaje de texto.

El formato de instrucción para el programa es el siguiente:

```
#include <SoftwareSerial.h>
char inchar;
SoftwareSerial SIM900(2,3);

int salida = 12;

void setup()
{
  Serial.begin(19200);
  pinMode(salida,OUTPUT);
  digitalWrite(salida,LOW);

 pinMode(8,OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2000);
 digitalWrite(8,LOW);
 delay(3000);

 SIM900.begin(19200);
  delay(20000);
 SIM900.println("AT+CMGF=1");
 delay(100);
 SIM900.println("AT+CNMI=2,2,0,0,0");
 delay(100);
 Serial.println("Ready...");
}
 void loop()
{
 if(SIM900.available() > 0)
 {
  inchar=SIM900.read();
  if (inchar == 'R')
```

```
    {
     delay(10);
     inchar=SIM900.read();
       if (inchar == '=')
       {
         delay(10);
         inchar = SIM900.read();
         if (inchar == '1')
         {
           digitalWrite(salida, HIGH);
           delay(5000);
           digitalWrite(salida,LOW);
         }
         }
           SIM900.println("AT+CMGD=1,4");
        }
    }
}
```

9.4.9 Solicitud de temperatura y humedad con el sensor DHT11 a través de un mensaje SMS

A continuación se explicará cómo solicitar datos de temperatura y humedad del sensor DHT11 mediante un mensaje de SMS. Lo único que hay que hacer es enviar el mensaje T=0 y en respuesta se obtiene la temperatura y la humedad que mide el sensor. (Ver figura 9.69.)

Figura 9.69 Recibiendo lecturas de temperatura y humedad en mensaje SMS.

El código de la aplicación es el siguiente:

```
#include <SoftwareSerial.h>
#include "DHT.h"
SoftwareSerial cell(2,3);

#define DHTPIN 12
#define DHTTYPE DHT11

// DHT instance
DHT dht(DHTPIN, DHTTYPE);

char inchar; //variable para recibir el mensaje

void setup(){
 dht.begin();

 //comando para encender el modem SIM900
 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2000);
 digitalWrite(8,LOW);
 delay(3000);

 //inicia la comunicacion serial con el modem SIM900 a
19200
 Serial.begin(19200);
 cell.begin(19200);
}

void loop(){
 //lee las lecturas de temperatura y humedad

 float temp = dht.readTemperature();
 float hum = dht.readHumidity();

 if(cell.available() > 0)
  {
    inchar = cell.read();
    if (inchar == 'T')
    {
      delay(10);
      inchar = cell.read();
      if (inchar == '=')
      {
        delay(10);
        inchar = cell.read();
        if (inchar == '0')
        {
          cell.println("AT+CMGF=1");    // set SMS mode
to text
          delay(100);
```

```
            cell.println("AT+CNMI=2,2,0,0,0");
            delay(100);
            cell.println("AT+CMGS=\"4772236196\"");
            delay(100);
            cell.print("Temperatura = ");
            cell.print(temp,1);
            delay(100);
            cell.print(" Humedad = ");
            cell.print(hum,1);
            delay(100);
            cell.println((char)26);
            delay(100);
          }
          else if (inchar == '1')
          {
            //digitalWrite(led3, HIGH);
            //delay(1000);
            //digitalWrite(led3,LOW);
          }
          }
           cell.println("AT+CMGD=1,4");
        }
    }
}
```

9.4.10 Permitir un acceso mediante la huella digital

A continuación se describirá cómo permitir un acceso mediante la lectura de la huella digital. Para ello se necesita un sensor de huella digital. (Ver figura 9.70.)

Figura 9.70 Sensor de huella digital.

La figura 9.71 muestra cómo se conecta el hardware.

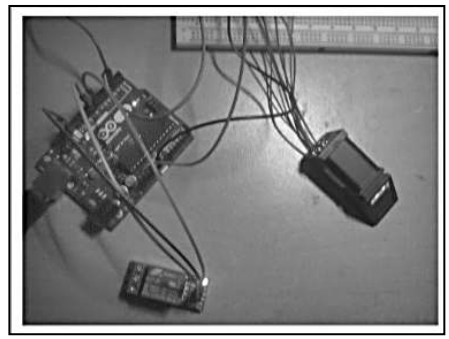


Figura 9.71 Conexión del hardware.

A continuación se presenta un ejemplo para dar de alta los ID de las huellas digitales directamente de la librería de Adafruit_Fingerprint:

```
/*************************************************
  This is an example sketch for our optical Fingerprint
sensor

  Designed specifically to work with the Adafruit BMP085
Breakout
  ----> http://www.adafruit.com/products/751

  These displays use TTL Serial to communicate, 2 pins
are required to
  interface
  Adafruit invests time and resources providing this open
source code,
  please support Adafruit and open-source hardware by
purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any re-
distribution
 *************************************************/

#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

uint8_t id;
uint8_t getFingerprintEnroll();

// Software serial for when you dont have a hardware se-
rial port
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
```

```
// On Leonardo/Micro/Yun, use pins 8 & 9. On Mega, just
grab a hardware serialport

SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySe-
rial);

// On Leonardo/Micro or others with hardware serial, use
those! #0 is green wire, #1 is white
//Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Se-
rial1);

void setup()
{
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(500);

  Serial.begin(9600);
  Serial.println("Adafruit Fingerprint sensor enroll-
ment");
  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
}
uint8_t readnumber(void) {
  uint8_t num = 0;
  boolean validnum = false;
  while (1) {
    while (! Serial.available());
    char c = Serial.read();
    if (isdigit(c)) {
       num *= 10;
       num += c - '0';
       validnum = true;
    } else if (validnum) {
      return num;
    }
  }
}

void loop()                     // run over and over
again
{
  Serial.println("Ready to enroll a fingerprint! Please
Type in the ID # you want to save this finger as...");
  id = readnumber();
  Serial.print("Enrolling ID #");
  Serial.println(id);
```

```
  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #");
Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(1);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint featu-
res");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint featu-
res");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
```

```
  }

  Serial.println("Remove finger");
  delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  Serial.print("ID "); Serial.println(id);
  p = -1;
  Serial.println("Place same finger again");
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.print(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint featu-
res");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint featu-
res");
      return p;
    default:
```

```
        Serial.println("Unknown error");
        return p;
    }

    // OK converted!
    Serial.print("Creating model for #");  Serial.printl-
n(id);

    p = finger.createModel();
    if (p == FINGERPRINT_OK) {
      Serial.println("Prints matched!");
    } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
      Serial.println("Communication error");
      return p;
    } else if (p == FINGERPRINT_ENROLLMISMATCH) {
      Serial.println("Fingerprints did not match");
      return p;
    } else {
      Serial.println("Unknown error");
      return p;
    }

    Serial.print("ID "); Serial.println(id);
    p = finger.storeModel(id);
    if (p == FINGERPRINT_OK) {
      Serial.println("Stored!");
    } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
      Serial.println("Communication error");
      return p;
    } else if (p == FINGERPRINT_BADLOCATION) {
      Serial.println("Could not store in that location");
      return p;
    } else if (p == FINGERPRINT_FLASHERR) {
      Serial.println("Error writing to flash");
      return p;
    } else {
      Serial.println("Unknown error");
      return p;
    }
}
```

Después hay que dar de alta el ID de la huella tecleándolo en el monitor serial, como lo pide el sistema, según el mensaje de la figura 9.72: *Please Type in the ID # you want to save this finger as...* [Por favor ingresa el # de ID con el que quieres guardar el dedo].

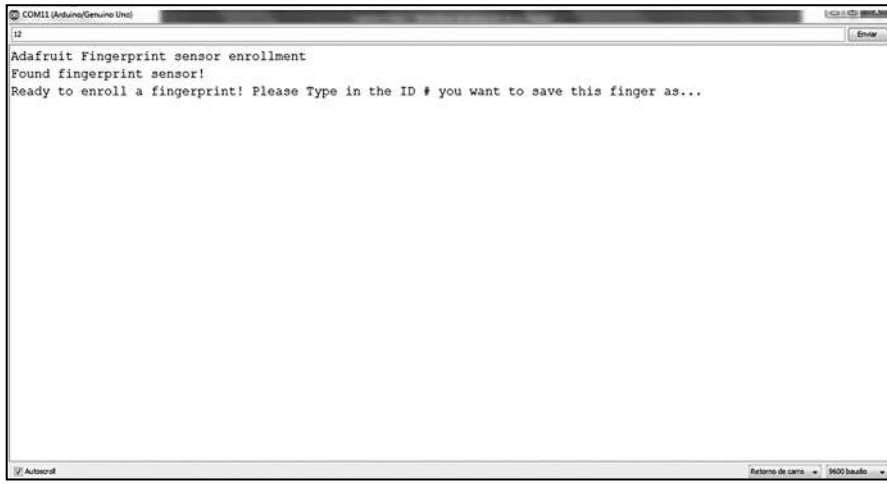


Figura 9.72 Tecleando el nombre del ID de la huella.

La figura 9.73 muestra cómo se ve en pantalla el mensaje que indica que se debe escanear nuevamente el dedo para confirmación de la huella: *Place same finger again* [Vuelve a marcar el mismo dedo].

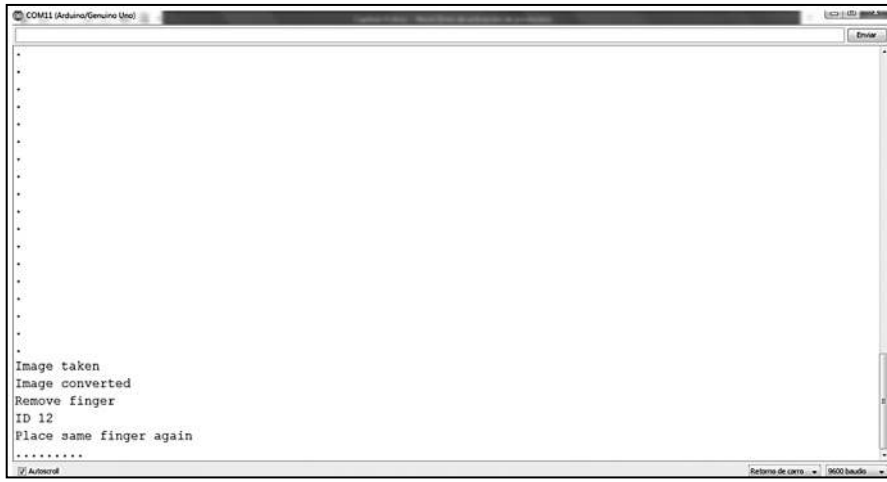


Figura 9.73 Mensaje que pide confirmar la huella.

La figura 9.74 muestra la respuesta del sensor, que indica que ya se guardó la huella digital no. 12 [*ID 12 Stored!*].

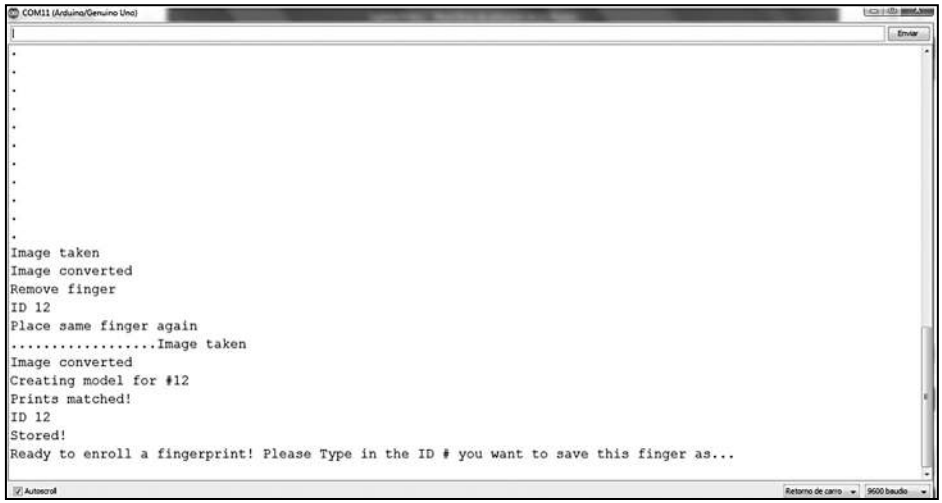


Figura 9.74 Pantalla del mensaje que confirma el registro exitoso de la huella.

Detección de huella digital y envío desde Twitter

Una vez que se ha guardado la huella digital, se utilizarán los servicios de la nube para el envío de un mensaje indicando que ésta ha sido detectada.

Primeramente, se debe autorizar a la tarjeta Arduino para que utilice una cuenta en la página https://arduino-tweet.apsspot.com, tal y como se presenta en la figura 9.75.

Figura 9.75 Abriendo una cuenta para enviar *tweet*.

Se solicitan los siguientes datos de la cuenta de Twitter. (Ver figura 9.76):

a) Nombre de usuario o correo electrónico
b) Contraseña

Figura 9.76 Abriendo una cuenta de Twitter para autorizar a Arduino a usar la aplicación.

La figura 9.77 muestra la pantalla con los datos capturados.

Figura 9.77 Datos capturados.

Una vez que se han capturado los datos de correo electrónico y contraseña se generará un token, o línea de captura, como se muestra en la figura 9.78.

Figura 9.78 Token generado.

Para que se pueda enviar el mensaje se necesita la tarjeta Ethernet Shield, con ello, el Arduino podrá enviar los mensajes al servicio de mensajería. (Ver figura 9.79.)

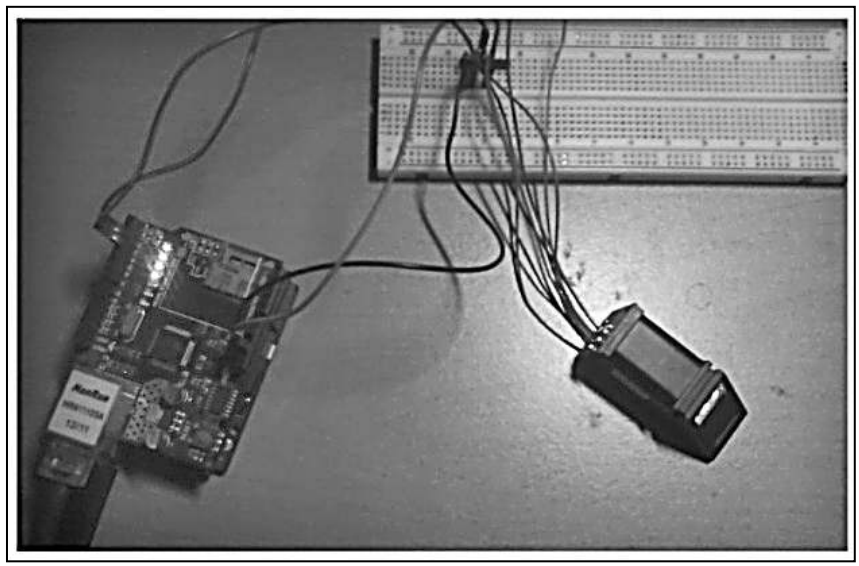


Figura 9.79 El escudo Ethernet conectado a la Arduino y al sensor de huellas digitales.

La figura 9.80 muestra que el programa pide esperar para validar la huella: *Found fingerprint sensor!/Waiting for valid finger...* [Se encontró el sensor de huellas digitales/ Esperando a validar dedo...]

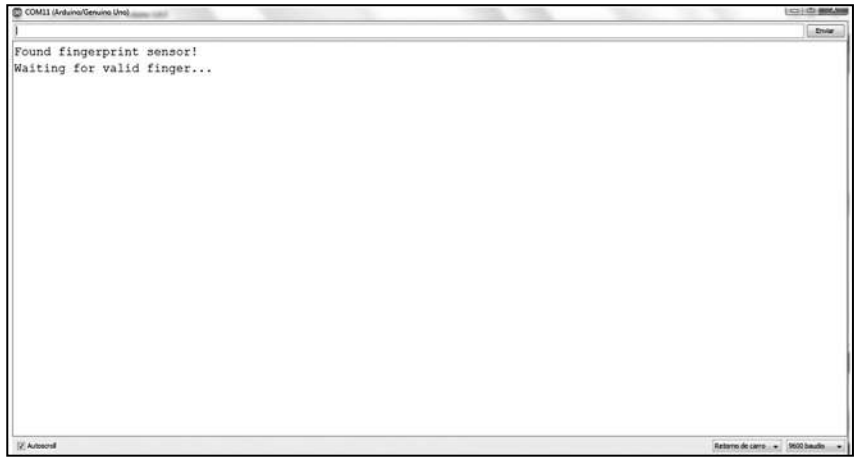


Figura 9.80 Esperando la validación de la huella digital.

La figura 9.81 muestra que se ha detectado la huella correcta: *Found ID#* [Se encontró el ID #].

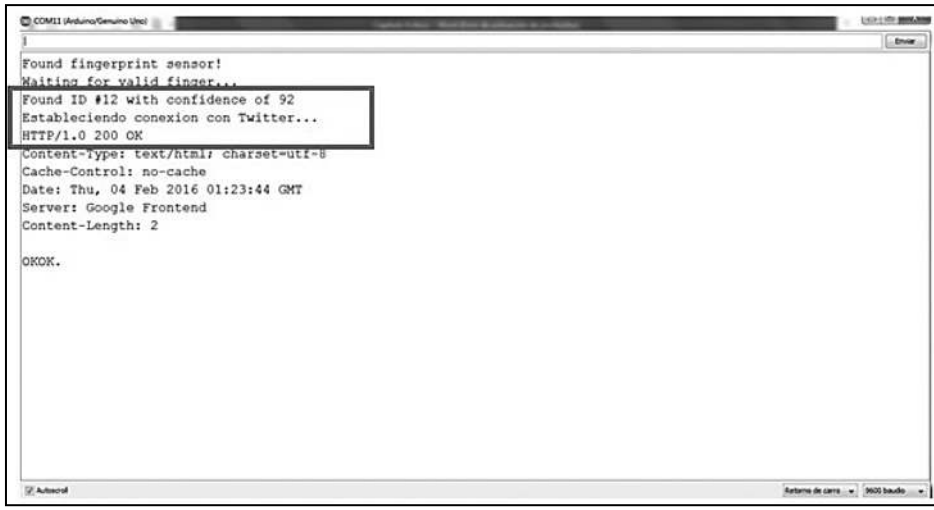


Figura 9.81 Respuesta del mensaje enviado y la detección de la huella correcta.

La figura 9.82 muestra el mensaje enviado a la cuenta de Twitter.

Figura 9.82 Pantalla del mensaje enviado a la cuenta de Twitter.

El código de la aplicación es el siguiente:

```
// Libraries
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Ethernet.h>
#include <SPI.h>
#include <Twitter.h>
// Configuracion de la Ethernet Shield
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

// Si no se especifica la IP, se utiliza DHCP
byte ip[] = {192,168,1,150 };

Twitter twitter("2598488138-ZebRCiSJUhkIW9DydqwK3LeXtY4r-
CRu7MzZEo22");

// Mensaje
//char msg[] = "Acceso Correcto concedido sensor de hue-
lla digital";

// Function to get fingerprint
int getFingerprintIDez();

// Init Software Serial
SoftwareSerial mySerial(2,3);

// Fingerprint sensor instance
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySe-
rial);

// Your stored finger ID
int fingerID = 12;
```

```
// Counters
int activationCounter = 0;
int lastActivation = 0;

void setup()
{
  delay(1000);
  Ethernet.begin(mac, ip);

  // Start Serial
  Serial.begin(9600);

  // Set the data rate for the sensor serial port
  finger.begin(57600);

  // Check if sensor is present
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
  Serial.println("Waiting for valid finger...");
}

//funcion para envio de tweet

void enviarTweet(const char envio[])
{
  Serial.println("Estableciendo conexion con Twit-
ter...");
   if (twitter.post(envio))
   {
    int estado = twitter.wait(&Serial);
    if (estado == 200)
    {
      Serial.println("OK.");
    }
    else
    {
      Serial.print("Error : code ");
      Serial.println(estado);
    }
   }
   else
   {
    Serial.println("Conexion fallida.");
   }
}

void loop()
{
  // Get fingerprint # ID
  int fingerprintID = getFingerprintIDez();
```

```
  // Activation ?
  if ( (activationCounter - lastActivation) > 2000) {

  if (fingerprintID == fingerID)
      {
        enviarTweet("Acceso Correcto concedido sensor de
huella digital");
      }
    lastActivation = millis();
  }
  activationCounter = millis();
  delay(50);
}
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK success!

  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint featu-
res");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint featu-
res");
```

```
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK converted!
  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    Serial.println("Found a print match!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.finge-
rID);
  Serial.print(" with confidence of "); Serial.println(fin-
ger.confidence);
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.finge-
rID);
  Serial.print(" with confidence of "); Serial.println(fin-
ger.confidence);
  return finger.fingerID;
}
```

En este proyecto es importante considerar que el número de ID es el que se guarda previamente al momento de guardar la huella digital. Esa misma ID es la que se debe incluir en el código del programa como se indica a continuación:

```
int fingerID = 12;
  int fingerprintID = getFingerprintIDez();
```

```
  if ( (activationCounter - lastActivation) > 2000) {
  if (fingerprintID == fingerID)
      {
        enviarTweet("Acceso Correcto concedido sensor de
huella digital");
      }
    lastActivation = millis();
  }
  activationCounter = millis();
  delay(50);
}
```

El código para la función que envía el mensaje a Twitter es el siguiente:

```
void enviarTweet(const char envio[])
{
  Serial.println("Estableciendo conexion con Twit-
ter...");
   if (twitter.post(envio))
   {
    int estado = twitter.wait(&Serial);
    if (estado == 200)
    {
      Serial.println("OK.");
    }
    else
    {
      Serial.print("Error : code ");
      Serial.println(estado);
    }
   }
   else
   {
    Serial.println("Conexión fallida.");
   }
}
```

9.4.11 Monitoreo remoto con cámara web conectada a la nube

En los dos proyectos que se realizarán en esta sección se van a utilizar las aplicaciones enfocadas al mundo de Internet ya estudiadas, con un grado mayor de complejidad. Primeramente, se explicará cómo realizar una aplicación con el sensor PIR de movimiento, para que al detectar movimiento se comunique con una cámara web, la cual tomará una foto y la guardará en la memoria Micro SD. Esta foto se enviará a un servicio en la nube en Dropbox. De tal forma, desde cualquier parte del mundo, ya sea desde una computadora, teléfono inteligente o tableta, se podrá consultar y monitorear lo que está ocurriendo en el lugar donde se coloque la cámara.

Por otro lado, en el segundo proyecto se realizará el monitoreo de la cámara web, de forma inalámbrica, desde un navegador web para monitorear la cámara desde cualquier otro lugar, creando una aplicación en .NET.

Para realizar esta aplicación se utilizará el siguiente material. (Ver figura 9.83):

- Tarjeta Arduino YUN
- Cámara web C270 de la marca Logitech (con conexión USB)
- Sensor de movimiento PIR

Figura 9.83 Material para el monitoreo remoto con cámara web conectada a la nube.

Conexión del hardware

Para conectar los tres elementos hay que seguir dos simples pasos:

1) Conectar a la toma de corriente la señal de voltaje VCC del sensor a +5volts, la tierra (GND) del sensor a la GND de la tarjeta Arduino y la señal del sensor al pin 8 de la tarjeta. (Ver figuras 9.84 y 9.85.)

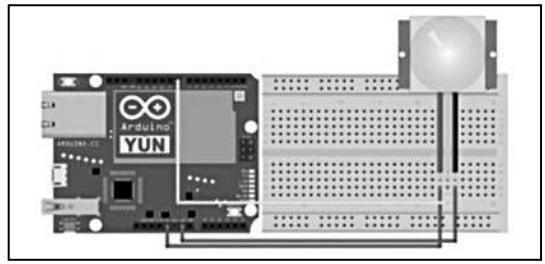


Figura 9.84 Diagrama de conexión de la tarjeta Arduino al sensor.

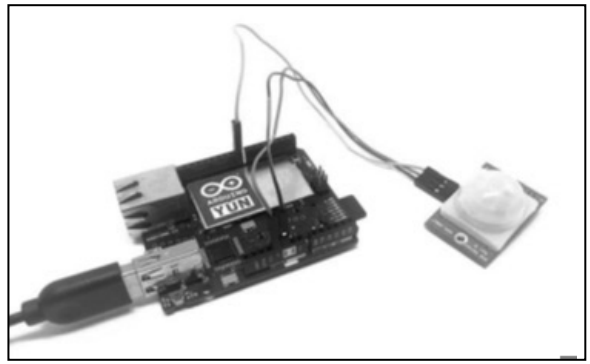


Figura 9.85 Así deben quedar conectados el sensor y la tarjeta.

2) Conectar la cámara al puerto USB. (Ver figura 9.86.)

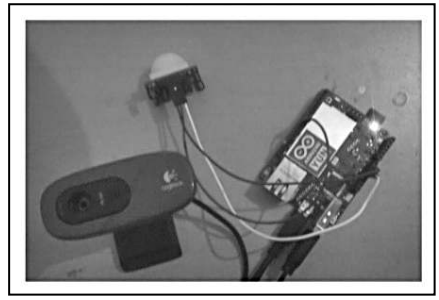


Figura 9.86 Así deben quedar conectados los tres elementos.

Configuración de la tarjeta Arduino YUN

Para configurar la tarjeta Arduino YUN, es necesario acceder de forma inalámbrica desde un navegador y teclear la siguiente dirección: http://192.168.240.1 o la dirección: http://arduino.local

Al principio, el programa solicitará la contraseña y el usuario, como se muestra en la figura 9.87.

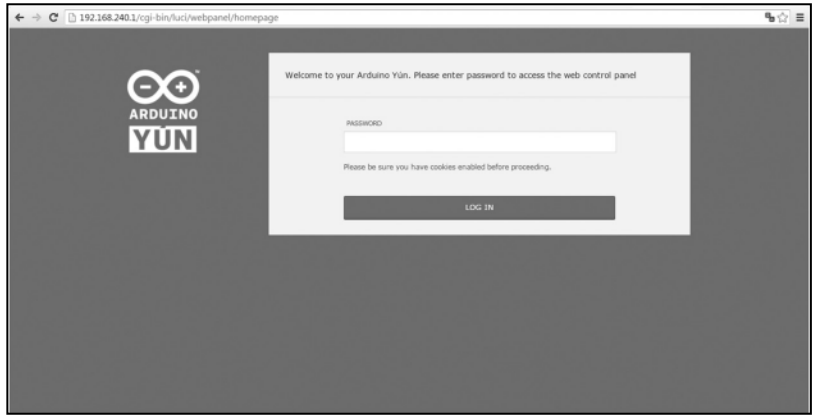


Figura 9.87 Pantalla de acceso al programa de la tarjeta YUN.

Una vez que se tiene acceso, en pantalla mostrará la configuración de la tarjeta Arduino YUN y se debe verificar que sea correcta, como se puede apreciar en la figura 9.88.

Figura 9.88 Verificando la configuración de la tarjeta YUN.

Una vez que se conoce la dirección IP de la tarjeta YUN, hay que utilizar el software de terminal PuTTY para acceder al sistema interno de la tarjeta y ejecutar los comandos básicos de configuración [*Basic Options for your PuTTY sesión*], como se muestra en la figura 9.89.

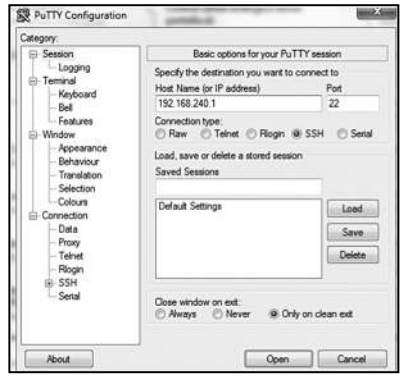


Figura 9.89 Configuración de la sesión en PuTTY.

A continuación, de acuerdo con la pantalla de la figura 9.90, se solicita el usuario, que en este caso es "root".

Figura 9.90 Estableciendo el usuario.

Luego, el sistema solicitará una contraseña o password. (Ver figura 9.91.)

Figura 9.91 Ingresando la contraseña.

Una vez que se accede al programa, aparece el menú principal del entorno de Linux, como se ve en la figura 9.92.

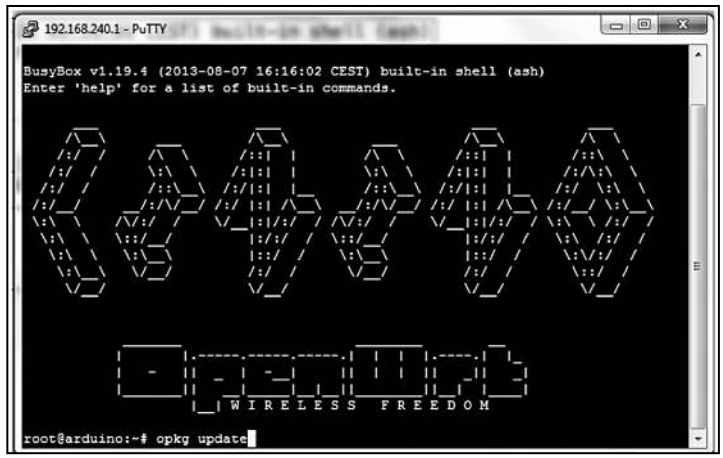


Figura 9.92 Entorno de Linux en PuTTY.

Posteriormente, hay que teclear los comandos de la figura 9.93. Es importante que la tarjeta Arduino se encuentre conectada a Internet a través de un cable Ethernet.

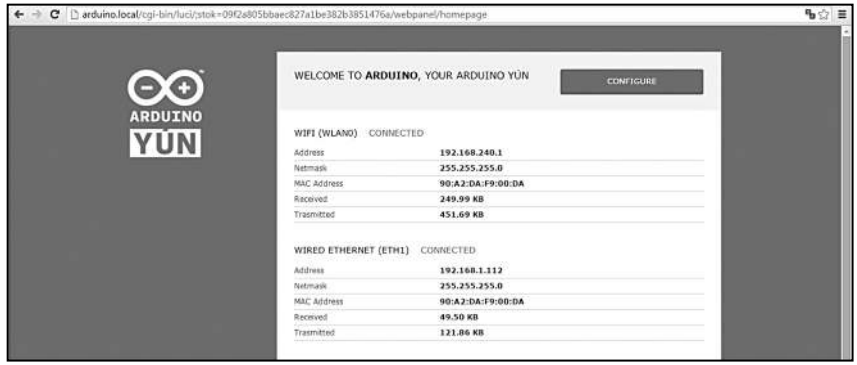


Figura 9.93 Estableciendo los comandos en la tarjeta Arduino YUN.

Después, ejecutar el comando opkg update, como se muestra en la figura 9.94.

Figura 9.94 Ejecutando el comando opkg update.

Asimismo, ejecutar los siguientes comandos:

- opkg install kmod-video-uvc
- opkg install fswebcam
- opkg install mjpg-streamer

Ya que se han ejecutado todos estos comandos, se ha realizado la configuración correcta para que el Arduino YUN pueda funcionar.

Usando una cuenta de Dropbox para conectarla al sistema de Arduino

Si no se tiene una cuenta de Dropbox, se puede abrir una en la dirección: https://www.dropbox.com

Una vez que se tiene cuenta en Dropbox, entrar a la siguiente dirección: https://www.dropbox.com/developers

Para crear una nueva app se debe hacer clic sobre el icono "Create app", que se muestra en la figura 9.95.

Figura 9.95 Creando una nueva app en Dropbox.

Después, dar clic en API Explorer, seleccionar el tipo de API de Dropbox y teclear el nombre de nuestra API a crear; en este caso, se decidió llamarla "CamaraArduinoYUN". (Ver figura 9.96.)

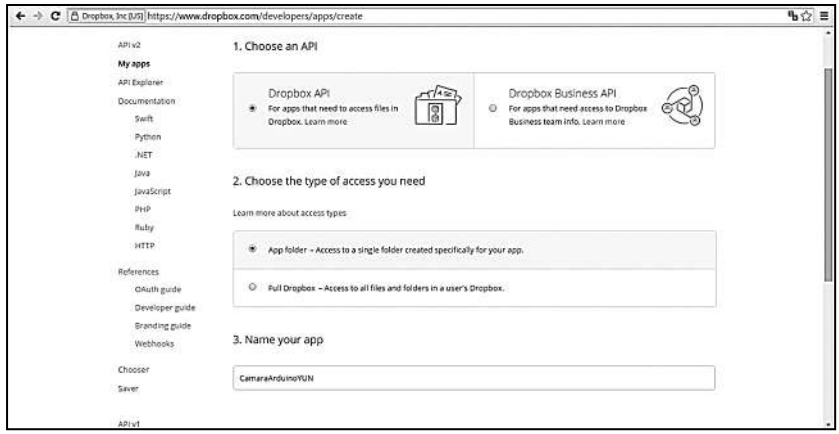


Figura 9.97 Nombrando la app en Dropbox.

A continuación, Dropbox mostrará los datos de la cuenta [*Settings*]. Es importante considerar la información de "App key" y de "App secret" (para ver la información en éste, se puede dar clic en Show), ya que se utilizarán más adelante. (Ver figura 9.98.)

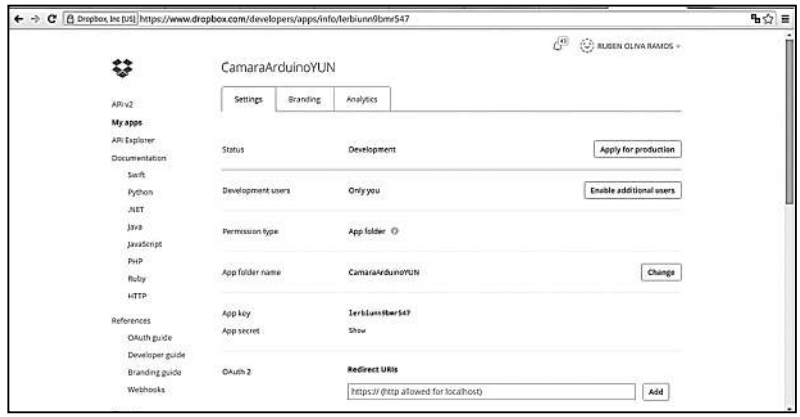


Figura 9.98 Pantalla con la información de la cuenta de la app en Dropbox.

Configuración de la cuenta de Temboo

Para conectar Dropbox con las otras aplicaciones, utilizaremos el servicio de Temboo. Hay que crear una cuenta en la siguiente dirección: https://www.temboo.com/library/

Una vez que se tiene la cuenta, hay que entrar a: https://www.temboo.com/library/Library/Dropbox/OAuth/InitializeOAuth/

De tal forma, se podrá configurar el acceso a Dropbox mediante la autorización desde la cuenta de Temboo. Así se pueden vincular las fotos que la cámara web guardará en Dropbox con la página web, mediante la cual se quiere monitorear la actividad del sensor.

En la figura 9.99 se puede ver en dónde hay que teclear las claves que se generaron en Dropbox: appKey y appsecret; en este caso, se ingresan en el recuadro llamado "INPUT".

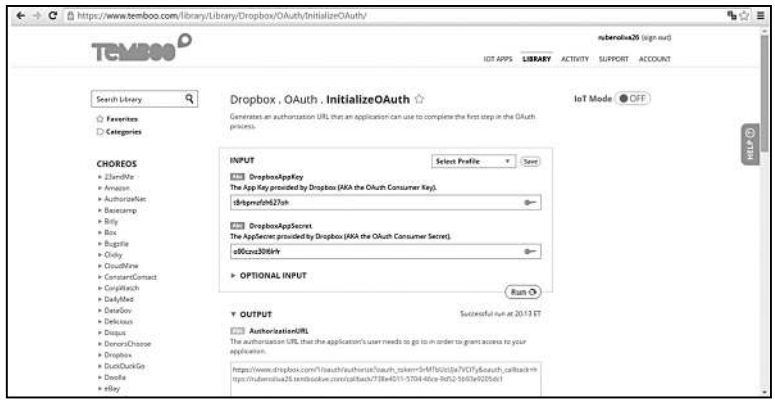


Figura 9.99 Vinculando la aplicación de Dropbox con Temboo.

Después, dar clic en el botón Run, que se encuentra en la esquina inferior derecha del recuadro de INPUT, tras lo cual aparecerán varios datos en el recuadro de OUTPUT, como la URL de autorización y la identificación de devolución de llamada. Hay que notar que el mismo sistema pide guardar estos datos, puesto que se necesitarán para correr las aplicaciones. (Ver figura 9.100.)

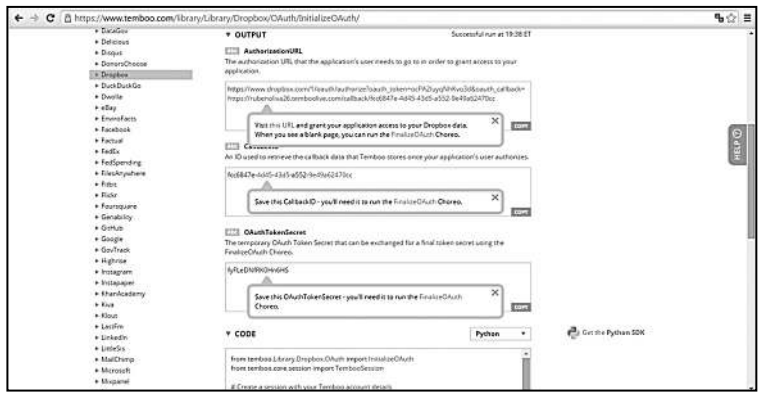


Figura 9.100 Empezando a vincular Dropbox con otras aplicaciones.

El siguiente paso es hacer clic en el vínculo que Temboo da como autorización. (Ver figura 9.101.)

Figura 9.101 Vínculo de acceso a Dropbox desde Temboo.

Al darle clic a Permitir para acceder a la carpeta, enseguida aparecerá una pantalla en blanco. (Ver figura 9.102.)

Figura 9.102 Accediendo a Dropbox desde Temboo.

El siguiente paso para finalizar la autorización es teclear este link:

```
https://www.temboo.com/library/Library/Dropbox/OAuth/Fi-
nalizeOAuth/
```

Hay que capturar los datos que se solicitan y luego hacer clic en Run. (Ver figura 9.103.)

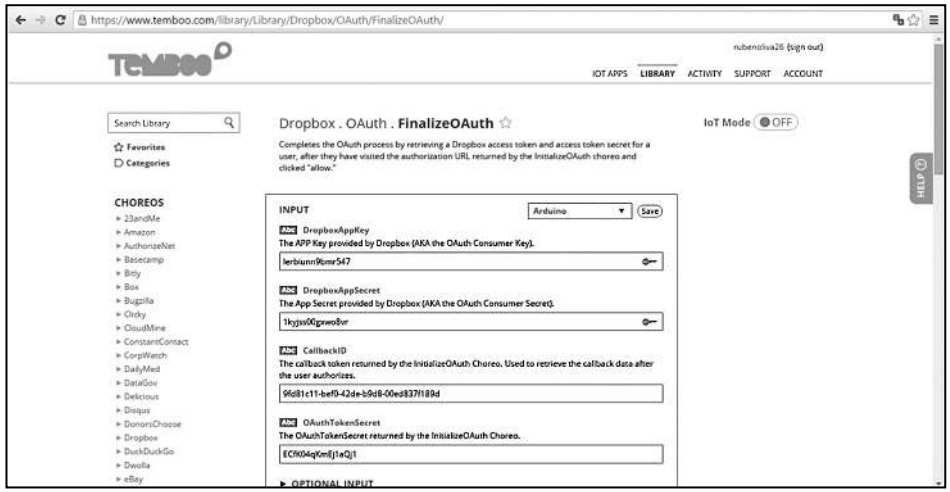


Figura 9.103 Finalizando la autorización para Dropbox.

Al final de todos los pasos, debe aparecer una pantalla similar a la figura 9.104.

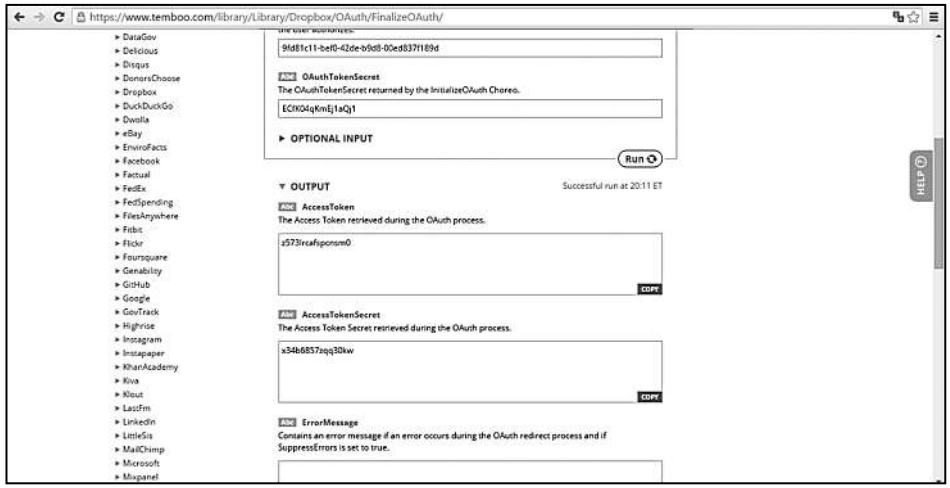


Figura 9.104 Finalizando la autorización de Dropbox.

En la cuenta que se creó de Temboo, entrar a la dirección: https://www.temboo.com/account/applications/

Aparecerá el nombre de la aplicación creada. (Ver figura 9.105.)

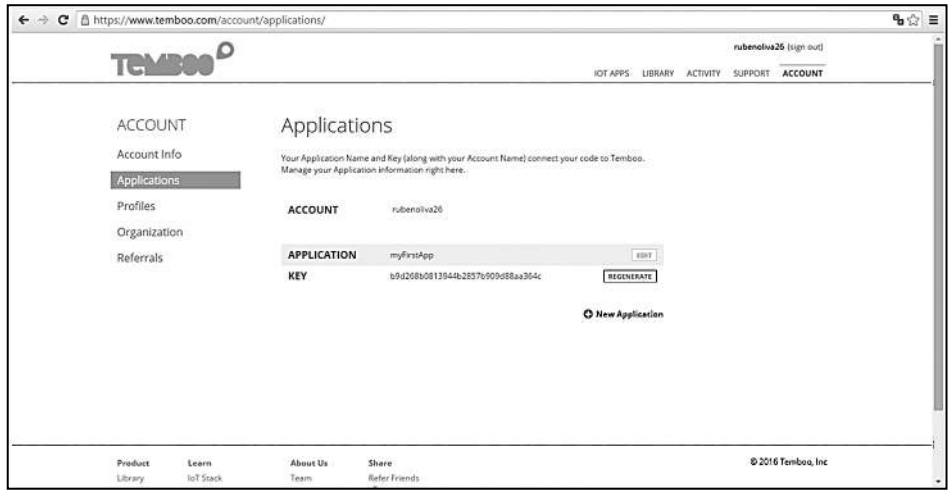


Figura 9.105 Aplicación creada.

Posteriormente, descargar el SDK de Python de la siguiente dirección: http://tembo.com/sdk/python

Luego, colocar la carpeta Temboo y el script de Python en la raíz de la memoria Micro SD.

Para probar la aplicación se debe pasar la mano sobre el sensor; el led verde de la cámara debe prender cuando detecta movimiento.

El código de la aplicación del Arduino es el siguiente:

```
#include <Bridge.h>
#include <Process.h>

// Picture process
Process picture;

// Filename
String filename;

// Pin
int pir_pin = 8;


// Path
String path = "/mnt/sda1/";

void setup() {

  // Bridge
  Bridge.begin();

  // Set pin mode
  pinMode(pir_pin,INPUT);
```

```
}

void loop(void)
{
    if (digitalRead(pir_pin) == true) {

    // Generate filename with timestamp
    filename = "";
    picture.runShellCommand("date +%s");
    while(picture.running());

    while (picture.available()>0) {
      char c = picture.read();
      filename += c;
    }
    filename.trim();
    filename += ".png";

    // Take picture
    picture.runShellCommand("fswebcam " + path + filename
+ " -r 1280x720");
    while(picture.running());

    // Upload to Dropbox
    picture.runShellCommand("python " + path + "upload_
picture.py " + path + filename);
    while(picture.running());
  }
}
```

Por otro lado, el script de Python que se llama cuando se detecta movimiento es el siguiente:

```
# coding=utf-8
# Script to upload files to Dropbox

# Import correct libraries
import base64
import sys
from temboo.core.session import TembooSession
from temboo.Library.Dropbox.FilesAndMetadata import
UploadFile

print str(sys.argv[1])

# Encode image
with open(str(sys.argv[1]), "rb") as image_file:
    encoded_string = base64.b64encode(image_file.read())

# Declare Temboo session and Choreo to upload files
session = TembooSession('yourSession', 'yourApp', 'your-
Key')
uploadFileChoreo = UploadFile(session)
```

```
# Get an InputSet object for the choreo
uploadFileInputs = uploadFileChoreo.new_input_set()

# Set inputs
uploadFileInputs.set_AppSecret("yourAppSecret")
uploadFileInputs.set_AccessToken("yourAccessToken")
uploadFileInputs.set_FileName(str(sys.argv[1]))
uploadFileInputs.set_AccessTokenSecret("yourTokenSecret")
uploadFileInputs.set_AppKey("yourAppKey")
uploadFileInputs.set_FileContents(encoded_string)
uploadFileInputs.set_Root("sandbox")

# Execute choreo
uploadFileResults = uploadFileChoreo.execute_with_re-
sults(uploadFileInputs)
```

Monitoreo inalámbrico de cámara web desde un navegador

Para monitorear el movimiento del sensor desde un navegador hay que seguir los siguientes pasos:

1) Ejecutar el comando:

```
opkg install mjpg-streamer.ipk

desde la consola de PuTTY
```

2) Ejecutar el comando:

```
mjpg_streamer -i input_uvc.so -d /dev/video0 -r 640x480
-f25

o

output_http.so -p 8080 -w /www/webcam" &
```

3) Ejecutar en el navegador la siguiente dirección para probar que la cámara está enviando la imagen a la pantalla:

```
http://arduino.local:8080
```

Hay que hacer clic en el vínculo que dice Stream. (Ver figura 9.106.)

Figura 9.106 Probando la cámara web.

Hasta aquí la complejidad del proyecto ha sido monitorear un lugar de forma inalámbrica. El siguiente paso será crear una interfaz en Visual Basic 2012 .NET con el objetivo de poder monitorear desde un navegador de Internet, a través de un formulario web que accede a la tarjeta Arduino YUN.

La figura 9.107 muestra la aplicación en el formulario:

Figura 9.107 Accediendo a los formularios de Visual Basic .NET.

El código de la aplicación es el siguiente:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
        WebBrowser1.Navigate(TextBox1.Text)
    End Sub
End Class
```

9.5 Resumen

En este capítulo se desarrollaron proyectos que se pueden aplicar en la vida diaria relacionados al Internet de las cosas. Además, estos mismos proyectos se pueden hacer más complejos; por ejemplo, conectando otros sensores o controlando otros dispositivos a través de los servicios en la nube y utilizando aplicaciones como: servicios web, servidores web o servidores de bases de datos.

Utilizando todas las herramientas que se han estudiado se pueden desarrollar aplicaciones personales usando las tecnologías actuales para la implementación de servicios orientados a la comunicación de redes y datos utilizando la tarjeta Ethernet Shield y todas sus aplicaciones.

Sin duda, el uso de la tarjeta Arduino y sus módulos de comunicación tienen como finalidad poder implementar servicios de datos en cualquier momento y al alcance de todos.

En el siguiente capítulo se describirá cómo desarrollar un sistema de monitoreo remoto para adquisición de datos.

9.6 Problemas

1. Elabore un sistema de registro de sensores mediante servicios web creando una base de datos.
2. Cree un sistema de monitoreo con varios sensores tales como LM35DZ y un sensor de luz (fotorresistencia) en una misma página web.
3. Lleve a cabo un prototipo inteligente de control de acceso de puerta principal, chapa electrónica y portón.
4. Realice un sistema de control de puertas y ventanas por medio de mensajes de texto.
5. Crear un sensor de presencia de llamada telefónica y desactivación de alarma por medio de mensaje de texto.
6. Lleve a cabo un sistema de seguridad con sensor PIR y envío de alerta mediante mensaje de texto SMS.
7. Diseñar un sistema de detección mediante sensor PIR y grabación de voz mediante micrófono.

8. Elabore un sistema espía con cámara web, toma de fotografía y envío de alertas a servicio web en la nube con hora y fecha del registro.
9. Cree un sistema de activación por medio de huella digital y, de acuerdo al registro realizado, activar o desactivar lo requerido por ese usuario: luces, puertas, etc.
10. Desarrolle un sistema de registro con base de datos de entradas a las casa (detectadas por sensores), que cuente con horas y días registrados.
11. Realice un control de acceso con sensor de huella digital con permisos cargados en una base de datos, el asignado de permisos deberá ir de acuerdo a los datos cargados.
12. Diseñe un casa domótica con control de las luces, ahorro de energía y activación por tiempo.
13. Llevar a cabo un control del riego de acuerdo a los días y horas desde una interfaz web.
14. Realice un sistema de seguridad con envío de alertas a través de Twitter.
15. Desarrolle un sistema de invernadero con servicios en la nube por medio de sensores interconectados.
16. Crear un sistema de monitoreo para varios dispositivos en una misma página web con servicios en la nube para control.
17. Elaborar un sistema de monitoreo para seguridad con sensor de gas y envío de alertas por medio de mensajes de texto.
18. Realice un sistema de detección de fugas de gas, registro en base de datos y monitoreo desde una página web.
19. Elabore un sistema de detección de humo y monitoreo desde una página web.
20. Llevar a cabo un sistema de gestión energético con bases de datos web y servicios en la nube para equipos con sensores de corriente.

Capítulo 10

Prototipo de un Sistema de Control Supervisorio y Adquisición de datos a distancia (SCADA)

Objetivos

En este capítulo se describirán los pasos para integrar la tarjeta Arduino y el software de .NET con el módulo Ethernet Shield. Esto se hace para lograr conectarse a una red industrial, con el fin de crear un sistema de control supervisorio y de adquisición de datos más complejo de lo anteriormente estudiado, puesto que, además, se presentará y explicará el protocolo Modbus TCP/IP.

10.1 Introducción

En este capítulo se desarrollará el prototipo de un sistema de control supervisorio y adquisición de datos a distancia utilizando una tarjeta Arduino y el módulo de red Ethernet Shield. Asimismo, el uso de las herramientas de software de .NET permitirán conectar una tarjeta Arduino a una red industrial mediante el protocolo Modbus TCP/IP, gracias al cual se podrá aumentar el grado de conectividad y, por lo tanto, de supervisión.

De tal forma que en este capítulo se realizarán los siguientes proyectos:

- Crear un sketch para trabajar en una red con el protocolo de comunicación industrial Modbus.
- Realizar pruebas de datos en la terminal AVReporter Modbus Communication Tester.
- Configurar tags, o etiquetas, en el servidor OPC.
- Crear una interfaz HMI en Visual Studio .NET.
- Crear una interfaz de monitoreo y control desde un servidor web.

10.2 Requerimientos de software y hardware

Para realizar estos proyectos se necesitan los siguientes dispositivos:

- Tarjeta Arduino UNO
- Ethernet Shield con el chip W5100
- Cable de red Ethernet
- Sensor DHT11
- Un potenciómetro de 4.7k
- Un botón pulsador
- Dos leds
- Dos resistencias de 330 ohms

Para llevar a cabo estos proyectos es necesario tener el siguiente software:

- Librería Modbus.h
- Microsoft Visual Studio .NET 2012; instalar Visual Basic .NET
- Módulo de National Instruments para .NET Measurement Studio, versión 2012
- Módulo de National Instruments DSC (*Data Logging and Supervisory Control*); instalar el módulo OPC Server

10.3 Redes Industriales

Las redes industriales están conformadas por varios equipos conectados entre sí para tener control y supervisión sobre procesos automatizados. Por otro lado, la automati-

zación es la ejecución autónoma, repetitiva y coordinada de las tareas necesarias para realizar un proceso en forma óptima, ajustando su desempeño a los lineamientos establecidos por el usuario.

De tal forma, en la actualidad, la ingeniería busca sistemas de control más amplios, que abarquen la instrumentación industrial, incluyan los sensores y actuadores de campo, los sistemas de control y supervisión, los sistemas de transmisión y recolección de datos, así como las aplicaciones de software en tiempo real (SCADA-*Supervisory Control and Data Acquisition*) y terminales HMI, para supervisar y controlar las operaciones de plantas o procesos industriales.

Por lo tanto, la **automatización industrial** emplea diferentes tecnologías, tales como:

- Neumática: para movimientos lineales y rápidos.
- Hidráulica: para movimientos de alta potencia.
- Mecánica: para mecanismos y accionamientos.
- Eléctrica: para motores y elementos de control.
- Electrónica: para control y comunicaciones.

En la figura 10.1 se puede apreciar la pirámide de la automatización y sus cuatro niveles: gestión, supervisión, control y campo.

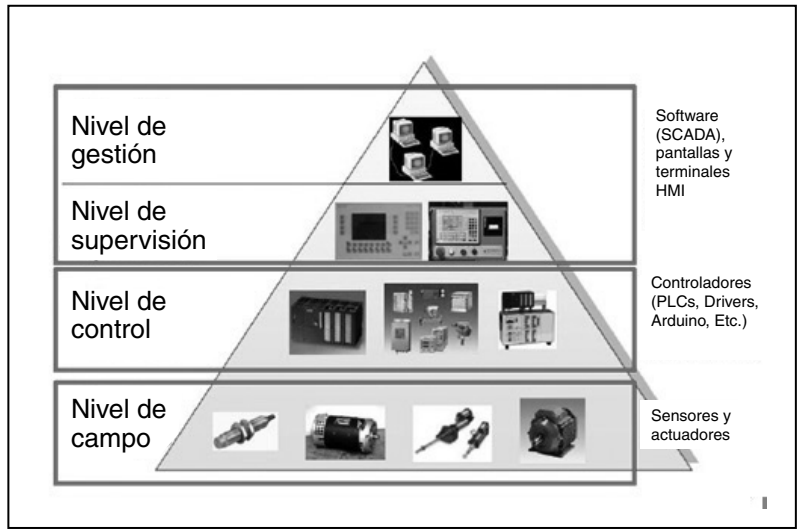


Figura 10.1 Pirámide de la automatización.

Las redes industriales son un tipo especial de **redes de datos**. Asimismo, los principios básicos de las redes de datos se cumplen tanto en las redes de oficina como en las redes industriales. Así, las redes industriales transportan datos con información proveniente de diversos dispositivos, tanto de nivel de campo (sensores y actuadores), como de nivel de control (PLC, controladores, micros, etc.). Dichos datos son, por ejemplo, valores de variables contenidas en un proceso (temperatura, presión, flujo, nivel, peso,

etc.), así como señales digitales para lectura de estados lógicos (sensores de contacto o de proximidad) o activación de dispositivos (actuadores). Actualmente, la arquitectura que forman las redes industriales también se conoce como **sistemas de control distribuido** (DCS por sus siglas en inglés, *Distributed Control System*).

A diferencia de una red de oficina, la red industrial deberá operar bajo condiciones extremas, puesto que se generan dentro de un **ambiente industrial** y, por ejemplo, deben lidiar con ruido electromagnético, distancias, condiciones climáticas severas, condiciones de seguridad y riesgo, etcétera.

En 1978, la Organización Internacional de Estándares, ISO (*International Standards Organization*), divulgó un conjunto de especificaciones que describían la arquitectura de red para la conexión de dispositivos diferentes. El documento original se aplicó a sistemas que eran abiertos entre sí, debido a que todos ellos podían utilizar los mismos protocolos y estándares para intercambiar información.

En 1984, la ISO presentó una revisión de este modelo y lo llamó "Modelo de referencia de Interconexión de Sistemas Abiertos" (OSI-*Open System Interconnection*), que se ha convertido en un estándar internacional y se utiliza como guía para las redes. En la figura 10.2 se puede apreciar la descripción de este modelo OSI.

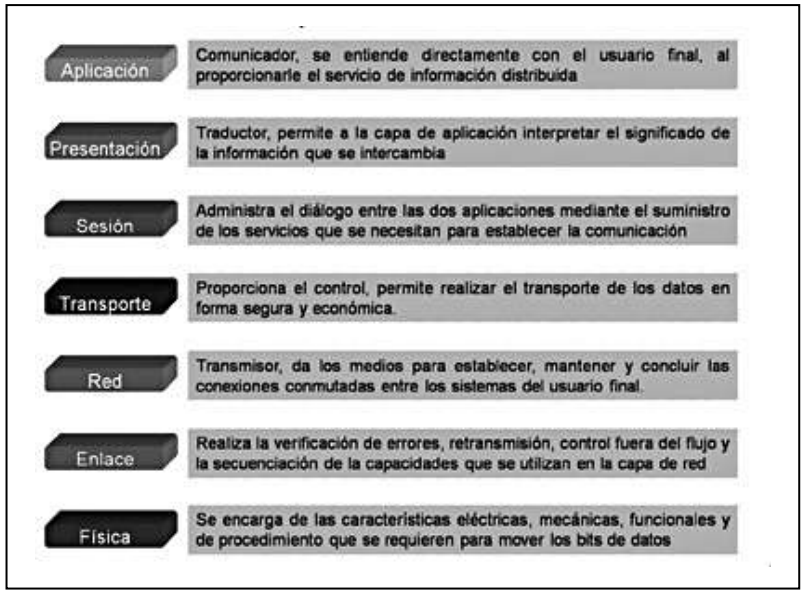


Figura 10.2 Modelo OSI.

Puesto que las redes industriales se componen de la conexión entre diversos dispositivos de campo y de control, comparten un determinado lenguaje o **protocolo de comunicación**, así como una forma de conexión entre ellos llamada **topología**. Éstas pueden tener, principalmente, las formas de: bus, anillo, estrella, estrella extendida, jerárquica o malla. (Ver figura 10.3.)

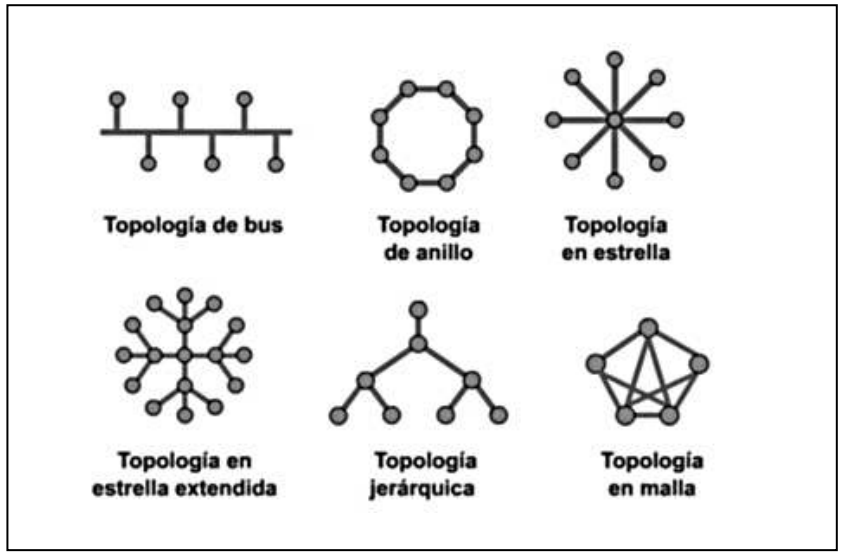


Figura 10.3 Topologías de las redes industriales.

En cuanto al protocolo de comunicación, existe una gran diversidad de ellos. Algunos, los **protocolos cerrados,** están diseñados específicamente por un fabricante o para un determinado equipo. Por otro lado, actualmente también existen los **protocolos abiertos,** diseñados para ser utilizados libremente por cualquier tecnología.

La principal función de la red industrial es enviar y recibir datos, por ejemplo, variables, configuraciones o bits de control, a un determinado dispositivo conectado a esta misma red. Otra función es compartir la información entre diferentes dispositivos, donde sea permitido. Es posible que en esta red se utilice un medio de Interfaz Hombre-Máquina (HMI-*Human Machine Interface*). Gracias a esta interfaz se puede proporcionar, en tiempo real, información importante de los datos que se utilizan durante el proceso mostrando los estados de operación (entradas y salidas) y los valores generados (variables). Asimismo, a través de la interfaz se pueden realizar diversos cambios, si es necesario, en las operaciones.

10.4 Protocolos de comunicación industrial

En el ambiente industrial se utilizan diversos protocolos (ver figura 10.4). Cabe destacar que, de todos ellos, los más utilizados son el protocolo **Modbus** (uno de los primeros protocolos estandarizados desde 1979) y el **Ethernet** (promovido industrialmente en 1996 por Schneider Electric y posiblemente el nuevo estándar mundial en la actualidad). Hoy en día, existe una combinación de ambos, el protocolo **Modbus TCP/IP**.

Este protocolo se utiliza ampliamente gracias a su relativa sencillez y a que muchos equipos comerciales cuentan con su incorporación de fábrica.

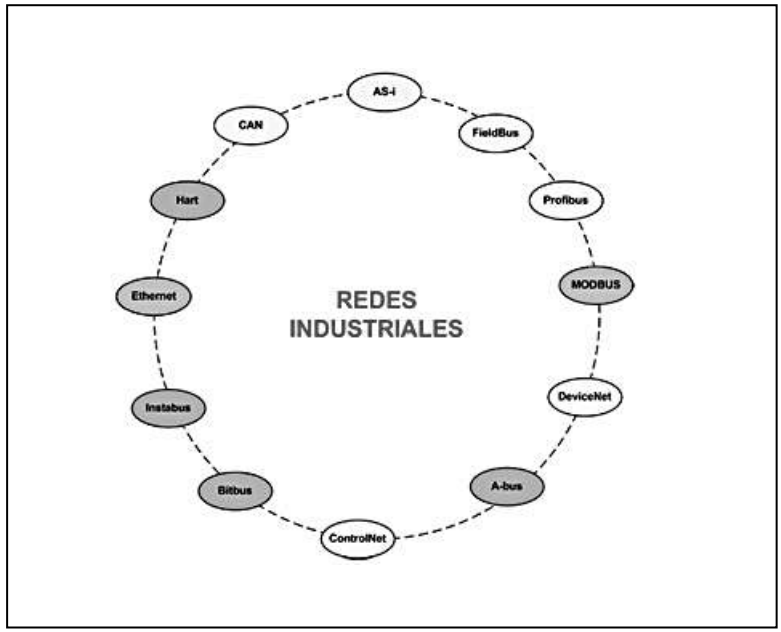


Figura 10.4 Protocolos de redes industriales.

10.4.1 Protocolo Modbus

Modicon desarrolló este protocolo para establecer comunicación entre dos o varias PLC. Debido a su simplicidad y especificación abierta, actualmente diferentes fabricantes lo utilizan ampliamente. Entre los dispositivos que lo utilizan podemos mencionar PLC *(Programmable Logic Controller)*, HMI, RTU *(Remote Terminal Unit)*, controladores, sensores y actuadores remotos. Este protocolo establece cómo se intercambian los mensajes en forma ordenada y detecta errores.

Modos de transmisión

Para conocer más sobre este protocolo, hay que conocer sus tres tipos de transmisión: RTU, ASCII y TCP/IP.

- **Modbus RTU (Remote Terminal Unit).** La comunicación entre dispositivos se realiza por medio de datos binarios. Ésta es la opción más usada del protocolo.
- **Modbus ASCII (American Standard Code for Information Interchange).** La comunicación entre dispositivos se hace por medio de caracteres ASCII.
- **Modbus TCP/IP.** La comunicación entre dispositivos se hace por medio de una red Ethernet. El protocolo Modbus es transportado a través de un protocolo TCP/IP. Es el método de transmisión más utilizado de Modbus.

Comunicación entre maestro y esclavo

El Modbus siempre funciona con un maestro y uno o más esclavos. En todo momento, el maestro controla el inicio de la comunicación con los esclavos, que, según la especifi-

cación, pueden ser hasta 247 en una misma red. El esclavo se limita a retornar los datos solicitados por el maestro. Es decir, usando el Modbus, el maestro envía los mensajes y el respectivo esclavo los responde. Por lo tanto, cada esclavo debe tener una única dirección, que puede ir desde 1 hasta 247; así, el maestro sabe con quién se debe comunicar. El maestro siempre inicia la comunicación enviando un paquete de información bien estructurado a todos los esclavos. Entre otras cosas, en la información se incluye el número del esclavo. Entonces, el esclavo elegido responde, enviando también lo que se le pide por medio de un paquete de información bien estructurado.

Campos de función de Modbus

A continuación se enlistan los campos de función del protocolo Modbus.

Código	Acción	Significado
01	Leer bobinas (0:xxxx)	Obtiene el estado actual ON/OFF de un grupo de bobinas lógicas.
02	Leer entradas (1:xxxx)	Obtiene el estado actual ON/OFF de un grupo de entradas lógicas.
03	Leer registros (4:xxxx)	Obtiene el valor binario de uno o más registros de almacenamiento.
04	Leer registros (3:xxxx)	Obtiene el valor binario de uno o más registros de almacenamiento.
05	Escribir bobina (0:xxxx)	Fuerza el estado de una bobina.
06	Escribir registro (4:xxxx)	Escribe el valor binario de un registro de almacenamiento.
15	Escribir bobinas (0:xxxx)	Fuerza el estado de un grupo de bobinas.
16	Escribir registros (4:xxxx)	Escribe el valor binario de un grupo de registros de almacenamiento.

10.5 Comunicación Modbus TCP/IP con Arduino y el Ethernet Shield

En esta sección se explicará cómo conectar la tarjeta Arduino a una red industrial a través del protocolo Modbus, mediante el protocolo de red TCP/IP. Esto será posible gracias a la librería Mudbus.h, creada por Dee Wykoff, en 2011.

La librería está limitada solo a utilizar las siguientes funciones, que se consideran las más utilizadas:

- Función 1 – Leer Bobinas Registros 0xxxx (Coils 0 - 128)
- Función 3 – Leer Registros 4xxxx (Holding Registers 16 bits 0 - 125)
- Función 5 – Escribir Bobinas Registros 0xxxx(Coils)
- Función 6 – Escribir Registros 4xxxx (Holding Registers 16 bits)

- El puerto de comunicación por default para Modbus TCP/IP es el 502
- El formato numérico para los Registros es de Entero con Signo (-32768 a +32767)

En la programación de Arduino con la librería Modbus, se debe considerar lo siguiente:

1) Incluir las librerías SPI.h, Ethernet.h y Mudbus.h.
2) Crear el objeto Modbus nombre; (sugerido Mb).
3) En la rutina void loop(), colocar la función con el nombre de "objeto creado". Para este ejemplo es Mb.Run(), con el objetivo de mantener la comunicación Modbus activa.
4) En la rutina void setup(), colocar los datos Ethernet mínimos necesarios para la comunicación (MAC e IP) e inicializar la comunicación Ethernet.
5) Los registros numéricos (Holding Register) se guardan en las variables Mb.R[x], en donde x = número de registro (0 - 125).
6) Los estados lógicos (coils) se guardan en las variables Mb.C[x], en donde x = número de registro (0 – 128).

10.6 Configuración del hardware

La figura 10.5 muestra un diagrama básico de cómo hacer la conexión del hardware.

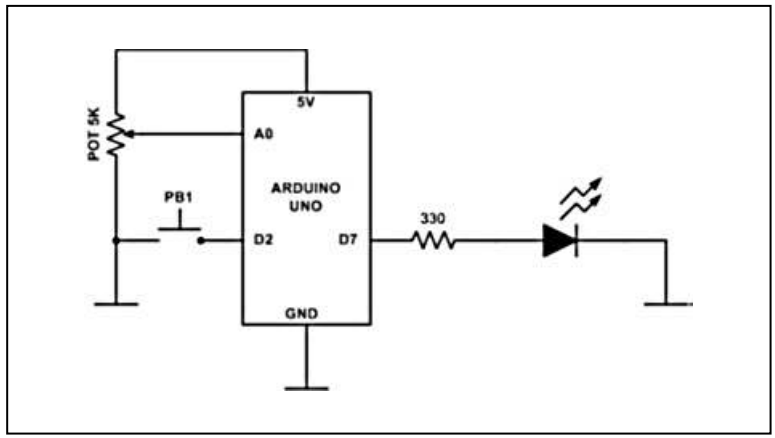


Figura 10.5 Diagrama de conexión.

Para realizar la conexión hay que hacer lo siguiente:

1) Conectar al canal analógico A0 el potenciómetro.
2) Conectar un led al pin 7 (control de salida).
3) Conectar un led al pin 9 (salida PWM).
4) Conectar botón a la entrada 2 (resistencia de pull up).
5) Conectar sensor DTH11 la señal de datos al pin 8. (Ver figura 10.6.)

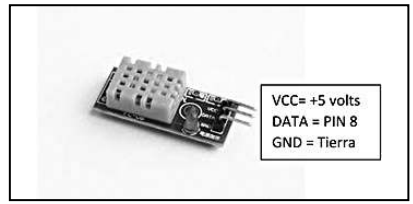


Figura 10.6 Configuración del sensor DHT11.

Una vez que se ha configurado el hardware, hay que hacer una prueba de comunicación de red a través de TCP/IP. Para realizar la lectura de las señales de adquisición es importante definir los siguientes conceptos para leer los datos desde la tarjeta Arduino:

- **Registros**. Definen los valores enviados por la tarjeta Arduino o valores de escritura a la tarjeta. Los direccionamientos para estos registros se configuran con las referencias a partir del número 40001.
- **Coils.** Definen las señales digitales de escritura y lectura desde la tarjeta Arduino. El direccionamiento para estas lecturas se nombra a partir de los números 00001.

En cuanto a la conexión del Ethernet Shield mediante Modbus, se deben realizar los siguientes pasos:

- Declarar la librería #include "Mudbus.h" para el manejo de la comunicación.
- Crear objeto Mudbus Mb para identificar el dispositivo que se va a encargar de enviar y recibir los datos.

1) En el void loop, poner el objeto para que se ejecute Mb.Run().
2) Para la lectura de las señales Mb.R[0], escribir una R y un corchete para identificar el número de registro a leer; por ejemplo, si se escribe: Mb.R[0] = analogRead(0), la lectura del canal analógico 0 se envía al registro Modbus número 0.
3) Si se quiere escribir o leer una señal digital (coils), ésta debe identificarse con: Mb.C[0]; por ejemplo, Mb.C[0] = 1 significa que se va a escribir un estado lógico 1 en el coil referenciado al número 0.

De acuerdo con el diagrama, el sketch de la aplicación para establecer comunicación con el protocolo Modbus es el siguiente:

```
#include <SPI.h>
#include <Ethernet.h>
#include "DHT.h"

#define DHTPIN 8
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

#include "Mudbus.h"
```

```
Mudbus Mb;

void setup()
{
  dht.begin();
  byte mac[] = {0x90, 0xA2, 0xDA, 0x00, 0x0A, 0xC2 };
  byte ip[]  = {192,168,1,150};

 Ethernet.begin(mac, ip); // Inicia Ethernet con la Mac e
IP del Arduino

  pinMode(7,OUTPUT);        // configura pin 7 como sali-
da.
  pinMode(2, INPUT_PULLUP); // configura pin 2 como entra-
da con pull-up
  pinMode(9,OUTPUT);
  delay(5000);  // tiempo de espera para arranque del
Ethernet Shield
}

void loop()
{
   float temp = dht.readTemperature();
   float hum = dht.readHumidity();

   Mb.Run(); // inicia la comunicación Modbus

   // Registros Modbus 40001,40002,40003 y 40004 Función
03.
   Mb.R[0] = analogRead(0);
   analogWrite(9,Mb.R[1]);
   Mb.R[2] = temp;
   Mb.R[3] = hum;

   // Registros Modbus 00001,00002,00003 Función 01.
   if(digitalRead(2) == 0)
   {
    Mb.C[0] = 1;
   }
   else
   {
    Mb.C[0] = 0;
   }

  //condiciones para leer estados lógicos
   if (Mb.C[1] == 1)
   {
    digitalWrite(7,HIGH);
    Mb.C[2] = 1;
   }
   else
   {
    digitalWrite(7,LOW);
```

```
        Mb.C[2] = 0;
      }
  }
```

Posteriormente, debe configurarse el software para realizar una prueba de comunicación Modbus y la tarjeta Arduino UNO: el AVReporter Modbus Communication Tester. Hay que realizar lo siguiente. (Ver figura 10.7):

1. Teclear la dirección IP de la tarjeta Arduino.
2. Identificar el tipo de dispositivo; es decir, Modbus/TCP device.
3. Asignar el puerto 502.
4. Iniciar la lectura de registros. Para ello, se debe poner la dirección inicial Mb.R[0] y la dirección final Mb.R[3]. Es importante considerar que debemos seleccionar la opción de Holding Register para la lectura de valores. Cabe señalar que si queremos leer valores de tipo Coil debemos seleccionar la opción Coil Status y también seleccionar el número de registro a leer.
5. Escribir los registros. Esto permite escribir un valor en los registros y también señales lógicas coils. Es importante considerar que se debe capturar el número de registro a escribir sobre él; en este caso, el número 1, Mb.R[1]; asimismo, en el campo de value se escribe el valor a escribir y se selecciona Preset Register. Si queremos escribir un coil, por ejemplo, registro Mb.C[1], seleccionar el número de registro (en este caso, 1) y teclear el valor a escribir; como son señales lógicas, escribir en el campo value: si es 1, activa el estado alto; si se escribe 0, desactiva el estado bajo. También es muy importante seleccionar Force Single Coil. En cada caso es importante hacer clic en el botón Read (lectura) o Write (escritura).
6. Al final, la pantalla debe mostrar los resultados que está leyendo en ese momento y las direcciones de los registros actuales.

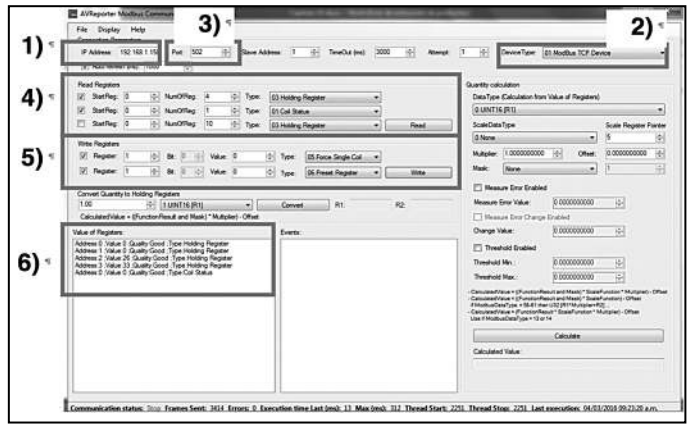


Figura 10.7 La prueba con el AVReporter Modbus Communication Tester.

10.7 Sistemas SCADA

Durante décadas, los sistemas de control solo integraban luces y sonidos para supervisar la operación de éste. De igual forma, la generación de datos durante el proceso se limitaba a la creación de datos temporales, los cuales se observaban en voluminosos *displays* y memorias limitadas para mantener la información. La creciente necesidad de poder observar en tiempo real y de forma continua un proceso, así como la generación y análisis de datos de éste, propició el desarrollo de los modernos sistemas SCADA.

Los sistemas SCADA [*Supervisory Control And Data Acquisition* o Control Supervisorio y Adquisición de Datos] son una HMI desarrollada como aplicación de software, que permite el enlace directo entre un proceso y su control. Se trata de algo similar a una ventana que muestra en tiempo real las condiciones operativas de un sistema de control. Estos sistemas son muy utilizados actualmente en la gran mayoría de las empresas de clase mundial, por ejemplo, las industrias automotriz, aeroespacial, alimenticia, de química y procesos, domótica, etcétera.

10.7.1 Elementos que conforman un sistema SCADA

Los elementos que conforman este tipo de sistemas son los que a continuación se describen. (Ver figura 10.8.)

Control. Modifica el estado de entradas y salidas, como valores de variables, set points, datos, etcétera.

Adquisición de datos. Obtiene datos en tiempo real generados por el proceso (variables), así como el estado operativo de entradas y salidas.

Supervisión. Visualiza estados y valores en un medio gráfico y de objetos relativos al proceso que se ejecuta. De tal forma es posible supervisar la operación de la máquina o proceso en tiempo real. También se pueden crear alarmas para alertar sobre situaciones críticas durante la operación.

Bases de datos y reportes. Recibe y procesa información que puede enviar a una base de datos, en donde los valores se almacenan en tiempo real para poder realizar reportes o informes de las condiciones de operación o del proceso.

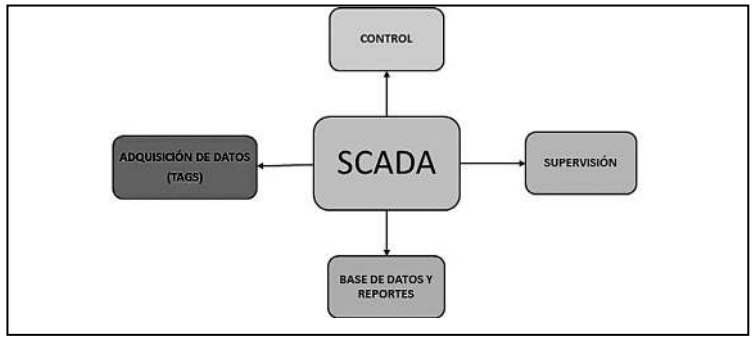


Figura 10.8 Elementos de los sistemas SCADA.

10.8 Servidores OPC

Un servidor OPC es una aplicación de software (un controlador) que cumple con una o más especificaciones definidas por la OPC Foundation. Sus siglas significan, en inglés, *OLE for Process Control*; es decir, OLE para control de procesos. OLE, a su vez, son las siglas de *Object Linking and Embedding;* es decir, vinculación y arraigamiento de objetos.

El servidor OPC hace de interfaz comunicándose, por un lado, con una o más fuentes de datos, utilizando sus protocolos nativos (generalmente PLC, DCS, básculas, módulos I/O, controladores, etc.) y, por el otro lado, con clientes OPC (generalmente SCADA, HMI, generadores de informes, generadores de gráficos, aplicaciones de cálculos, etcétera).

En una arquitectura cliente OPC/servidor OPC, el servidor OPC es el esclavo, mientras que el cliente OPC es el maestro. Las comunicaciones entre el cliente OPC y el servidor OPC son bidireccionales; esto significa que los clientes pueden leer y escribir en los dispositivos a través del servidor OPC.

Existen cuatro tipos de servidores OPC definidos por la OPC Foundation:

- **Servidor OPC DA.** El OPC Data Access es un servidor que se basa en Spezifikationsbasis. Fue diseñado especialmente para la transmisión de datos en tiempo real.
- **Servidor OPC HDA**. Servidor que se basa en la especificación de Acceso a Datos Historizados; es decir, provee al cliente OPC HDA de datos históricos.
- **Servidor OPC A&E Server**. Servidor que se basa en la especificación de Alarmas y Eventos; es decir, transfiere alarmas y eventos desde el dispositivo hacia el cliente OPC A&E.
- **Servidor OPC UA**. Servidor que se basa en la especificación de Arquitectura Unificada; permite a los servidores OPC trabajar con cualquier tipo de datos.

10.8.1 Clientes OPC

Un cliente OPC es el que se va a encargar de desplegar la información tomada del servidor OPC de forma gráfica. Comúnmente, el cliente OPC es una HMI que permite la interacción entre los dispositivos de campo y el software.

10.8.2 Servidor OPC de National Instruments

Para el proyecto de este capítulo se utilizará el módulo DSC *(Data Logging and Supervisory Control)*. Dentro de los programas a instalar viene incluido el servidor OPC. El servidor es un administrador de los dispositivos conectados y de las variables a las cuales se tiene acceso.

10.8.3 Configuración de las tags en el Servidor OPC

Es necesario configurar las tags, o etiquetas, en el servidor OPC de National Instruments. La figura 10.9 muestra cómo se hace esto en la pantalla.

Figura 10.9 Pantalla de Configuración del Servidor OPC.

Primeramente, debe darse de alta un canal. A continuación se describen los pasos para ello.

1) Hacer clic con el botón derecho para agregar un canal. (Ver figura 10.10.)

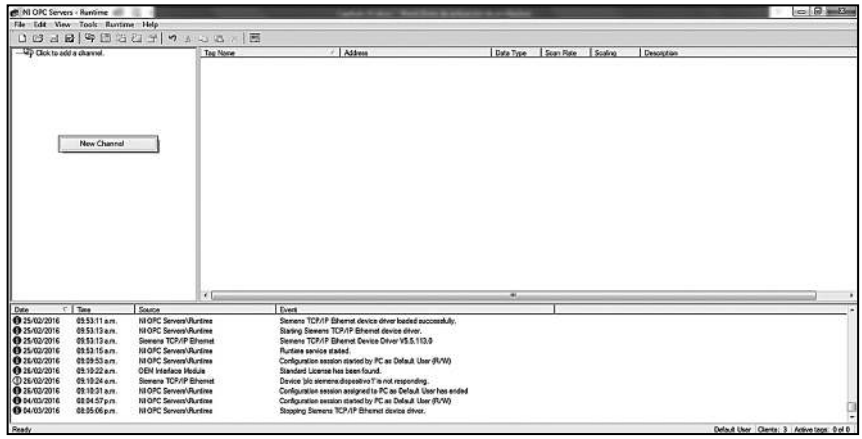


Figura 10.10 Al dar clic con el botón derecho, aparece la opción de dar de alta un canal.

2) Hacer clic en New Channel. Entonces aparecerá una ventana para nombrar al canal. (Ver figura 10.11.)

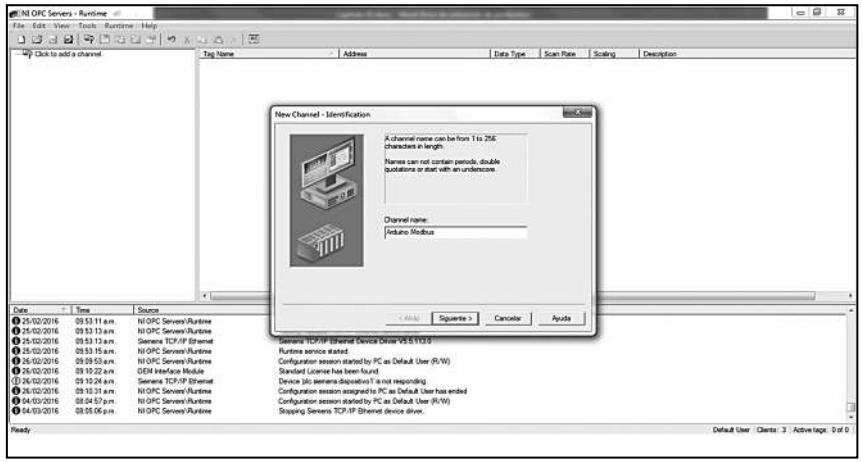


Figura 10.11 Nombrando al canal.

3) Al dar clic en Siguiente aparece una lista de controladores en el menú Device driver; seleccionar Modbus TCP/IP Ethernet. (Ver figura 10.12.)

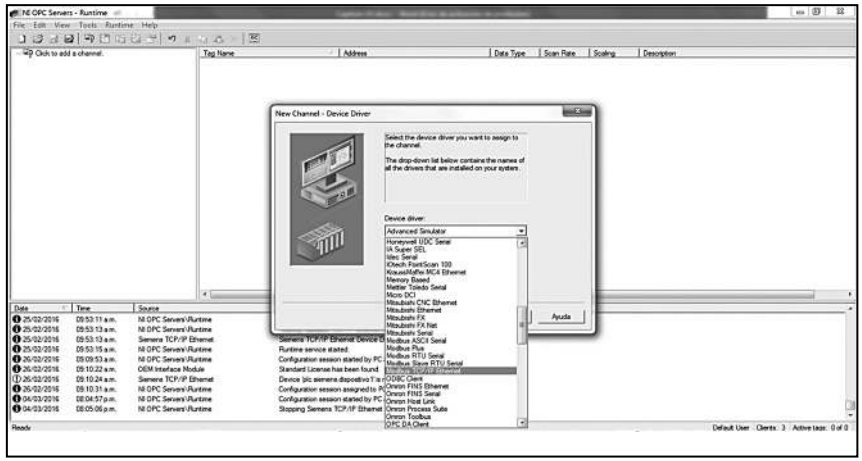


Figura 10.12 Seleccionando driver Modbus TCP/IP Ethernet.

4) Nuevamente, al dar clic a Siguiente, aparece una ventana del adaptador de red (Network Adapter); seleccionar el Default. (Ver figura 10.13.)

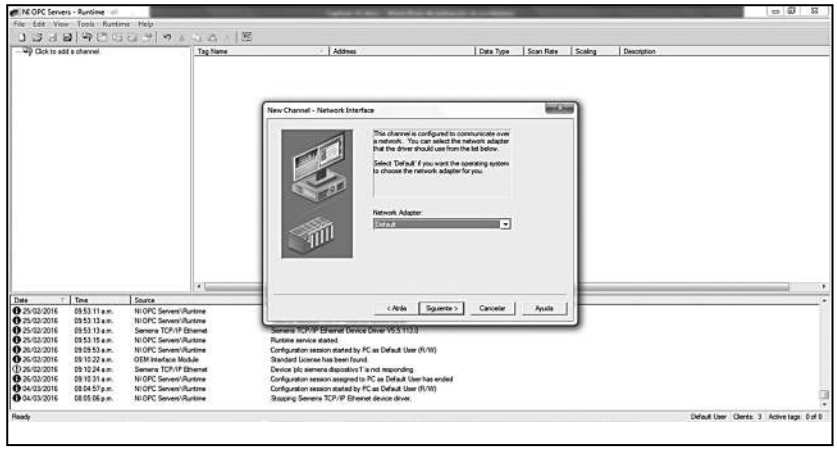


Figura 10.13 Seleccionando el adaptador de red Default.

5) Al dar clic en el botón Siguiente se establecen las configuraciones predeterminadas como aparecen en la figura 10.14.

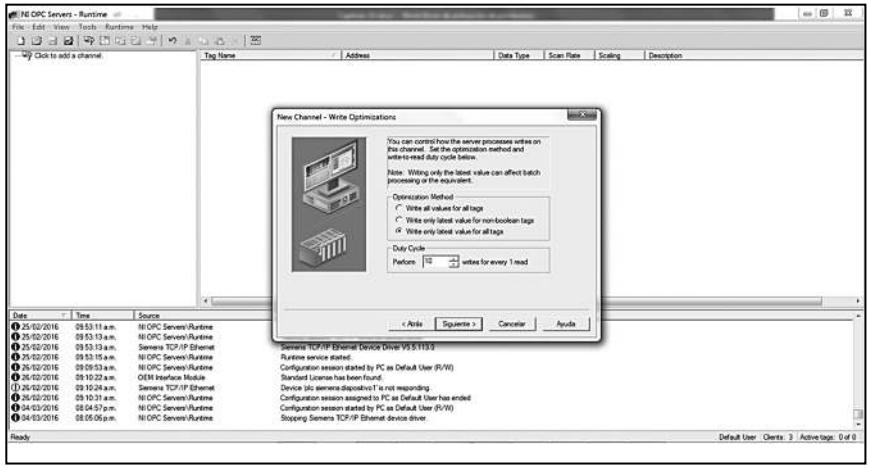


Figura 10.14 Estableciendo las configuraciones predeterminadas.

6) Posteriormente, hay que seleccionar el tipo de puerto y el protocolo TCP/IP. La opción predeterminada es el 502. (Ver figura 10.15.)

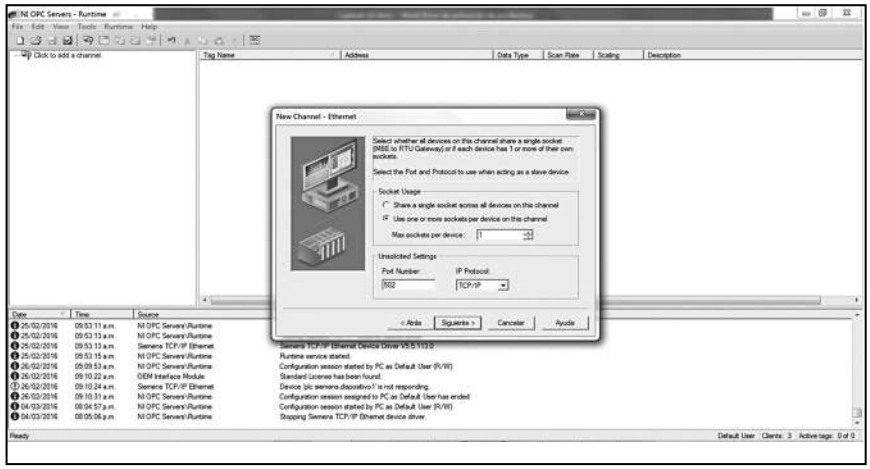


Figura 10.15 Seleccionando el tipo de puerto y el protocolo TCP/IP.

7) Al final, aparece una pantalla con el botón Finalizar. (Ver figura 10.16.)

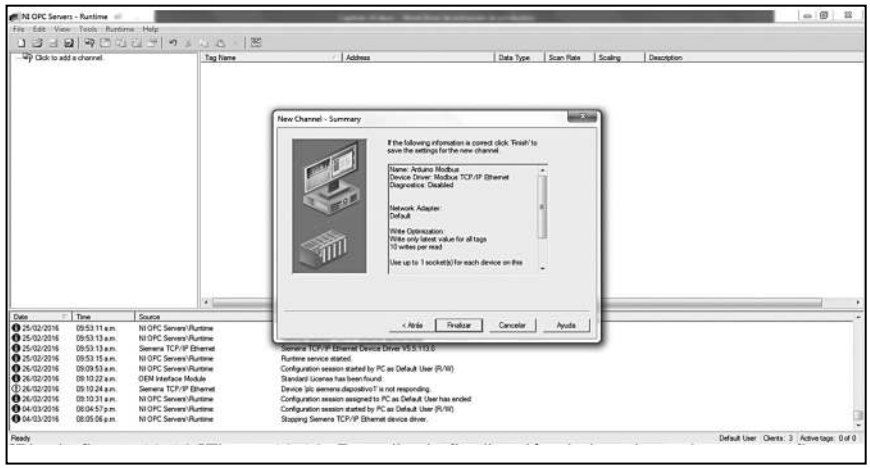


Figura 10.16 Pantalla de finalización de la asistencia para configurar el canal.

Entonces, aparecerá en la ventana principal, del lado izquierdo, el canal creado, como se ve en la figura 10.17.

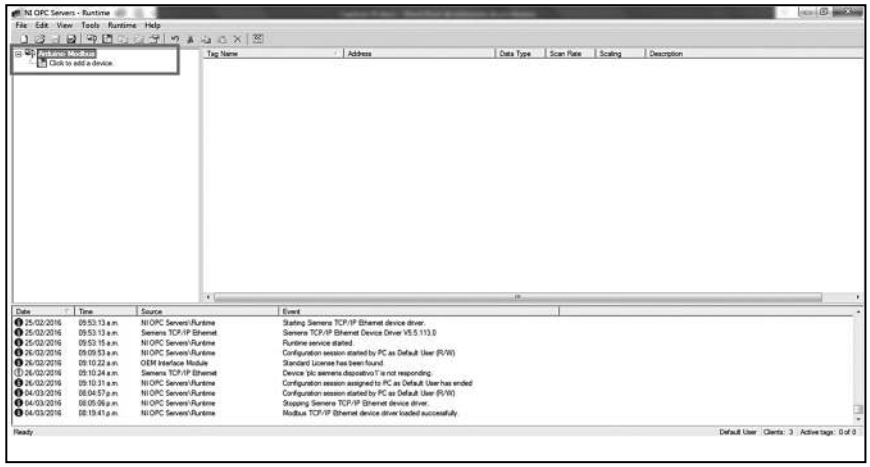


Figura 10.17 Ahora aparece el canal creado en la pantalla principal.

Una vez que se tiene el canal, se debe configurar el dispositivo. Los pasos son los siguientes:

1) Dar clic en el vínculo debajo del canal, llamado Click to add a device. Hay que teclear el nombre del dispositivo; en este caso, Arduino Ethernet. (Ver figura 10.18.)

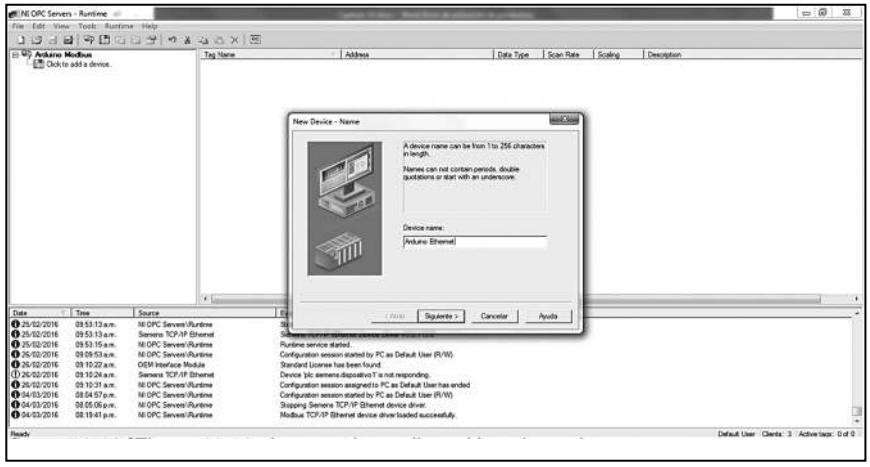


Figura 10.18 Agregar un dispositivo al canal.

2) Dar clic a Siguiente y seleccionar, en el campo Device Model, el tipo de controlador al que se va a conectar: Modbus. (Ver figura 10.19.)

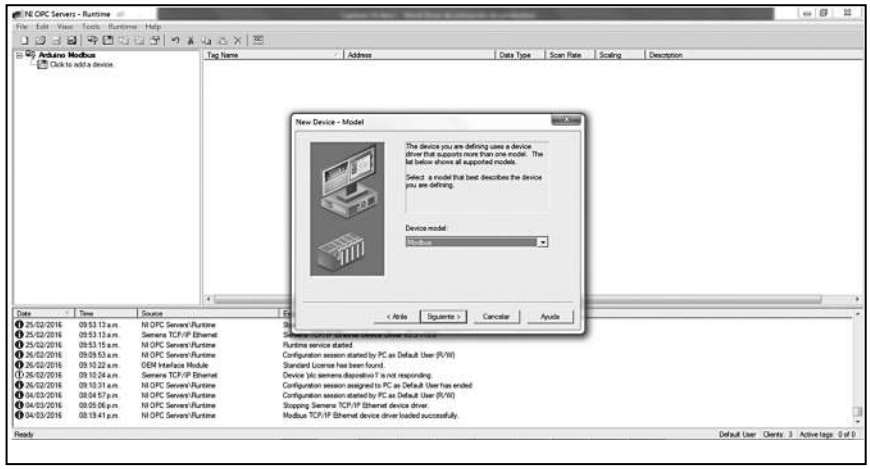


Figura 10.19 Seleccionar el dispositivo Modbus.

3) El programa solicitará la dirección IP de la tarjeta Arduino que se configuró en el sketch y que es la dirección que se le asignó a la placa Arduino: 192.168.1.150. (Ver figura 1.20.)

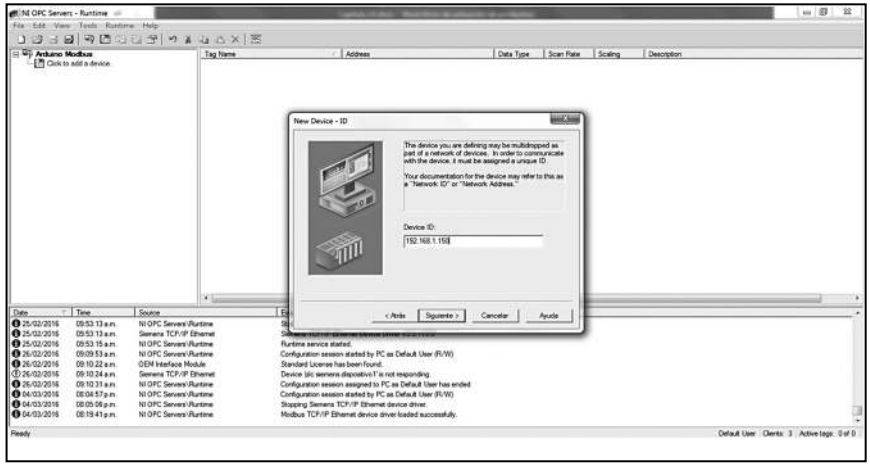


Figura 10.20 Teclear la dirección IP de la tarjeta Arduino.

4) Después hay que establecer los parámetros de configuración de tiempos de respuesta y de conexión. (Ver figura 10.21):

- Connect timeout: 3 seconds
- Request timeout: 1000 miliseconds
- Fail after 3 succesive timeouts
- Inter request delay: 0 miliseconds

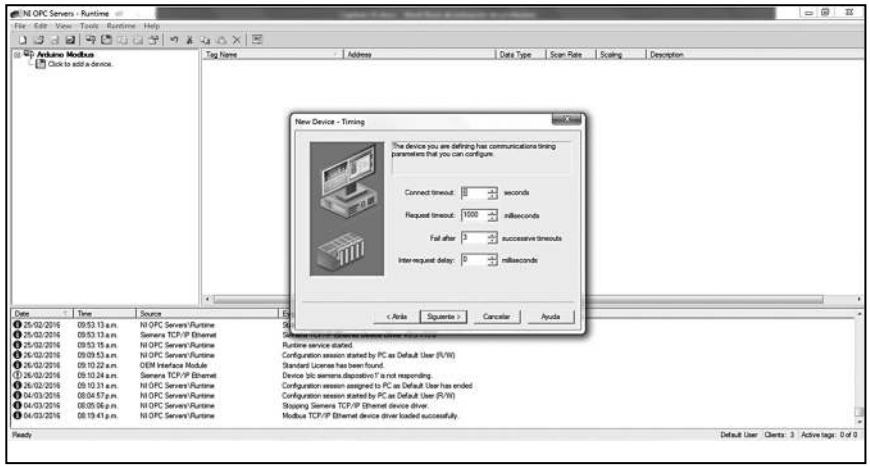


Figura 10.21 Establecer los tiempos de respuesta.

5) Luego de dar clic en el botón Siguiente, aparece una pantalla (ver figura 10.22) en la que no hay que configurar nada.

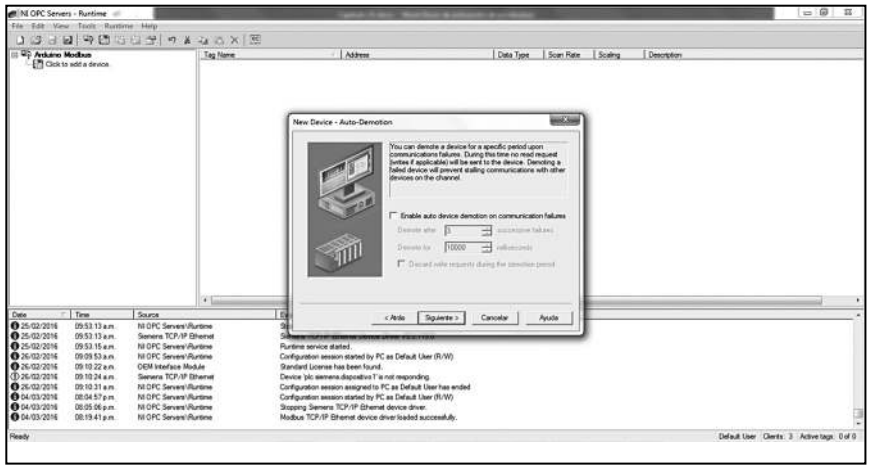


Figura 10.22 Pantalla en la que no hay que configurar nada.

6) Después de dar clic en Siguiente, todos los campos de configuración se dejan sin modificar. (Ver figura 10.23.)

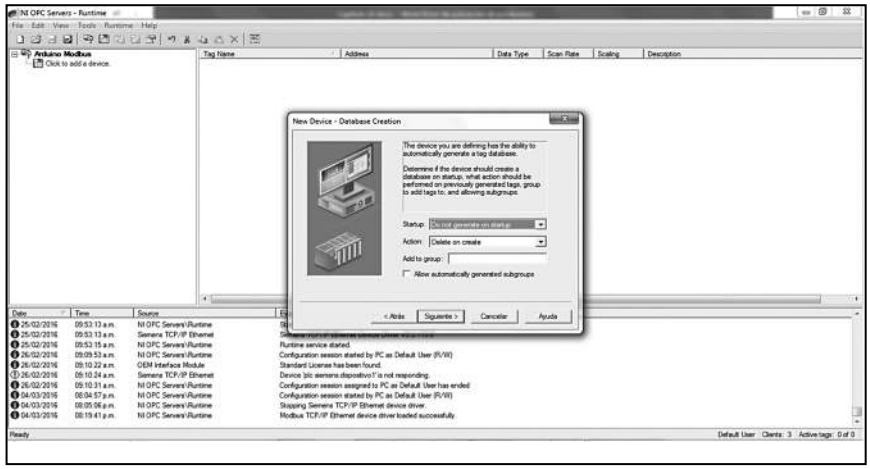


Figura 10.23 En esta pantalla no hay que modificar nada.

7) Después hay que configurar el puerto para el protocolo Modbus, que de forma predeterminada es el 502. (Ver figura 10.24.)

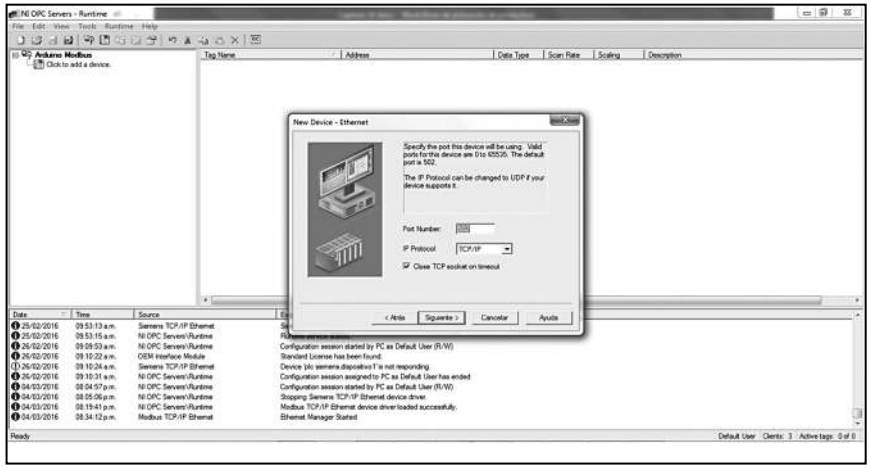


Figura 10.24 Configurando el puerto 502 para Modbus.

8) Luego hay que configurar la escritura de registros y tags en la pantalla Data Access Settings. Todos los campos deben estar activados (☑). (Ver figura 10.25.)

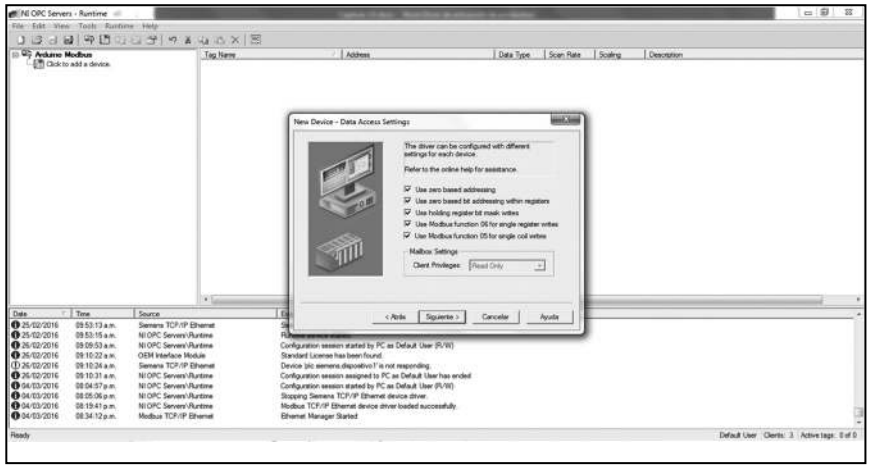


Figura 10.25 Configuración de registros y tags.

9) Dar clic en el botón Siguiente para que aparezcan las configuraciones para el dispositivo. Los primeros tres campos deben estar activados (☑), como se muestra en la figura 10.26.

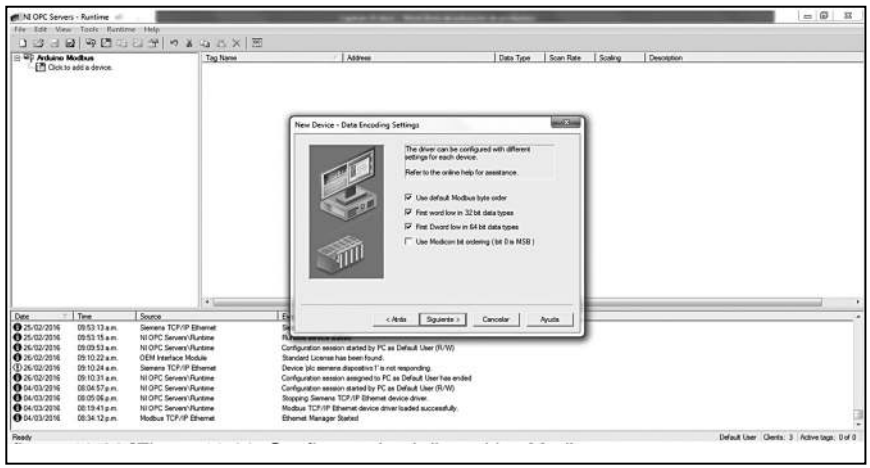


Figura 10.26 Configuración del dispositivo Modbus.

10) Después de dar clic a Siguiente, se podrán configurar los registros y coils. En este caso, todos en 32. (Ver figura 10.27.)

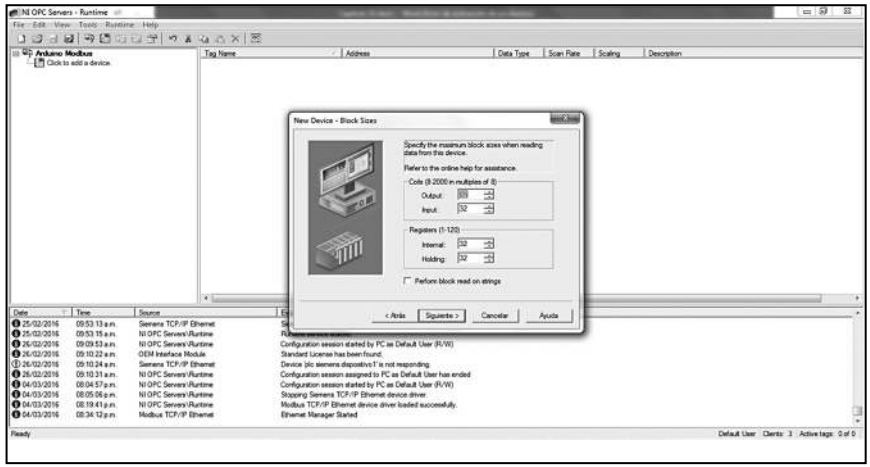


Figura 10.27 Configurar registros y coils en 32.

11) Dar clic en el botón de Siguiente, donde el sistema pedirá que demos la ubicación del archivo de variables tipo .txt (ver figura 10.28.)

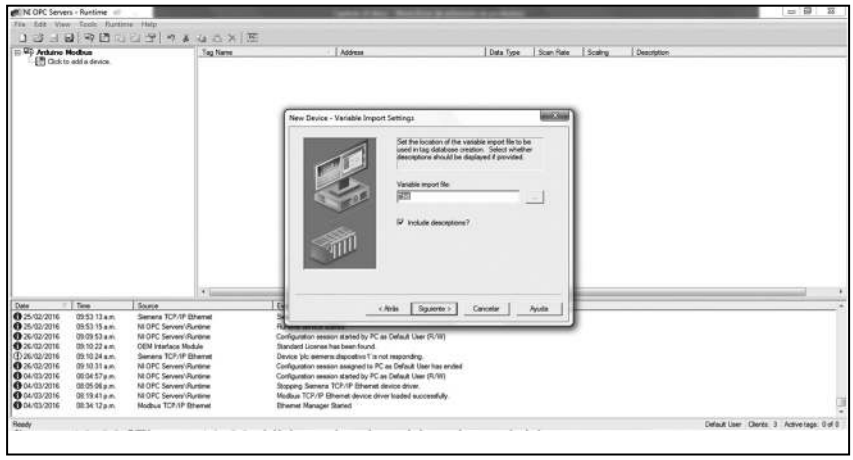


Figura 10.28 Ubicación del archivo de variables.

12) Dar clic en Siguiente y habilitar el comando que desactiva las tags de direcciones ilegales. (Ver figura 10.29.)

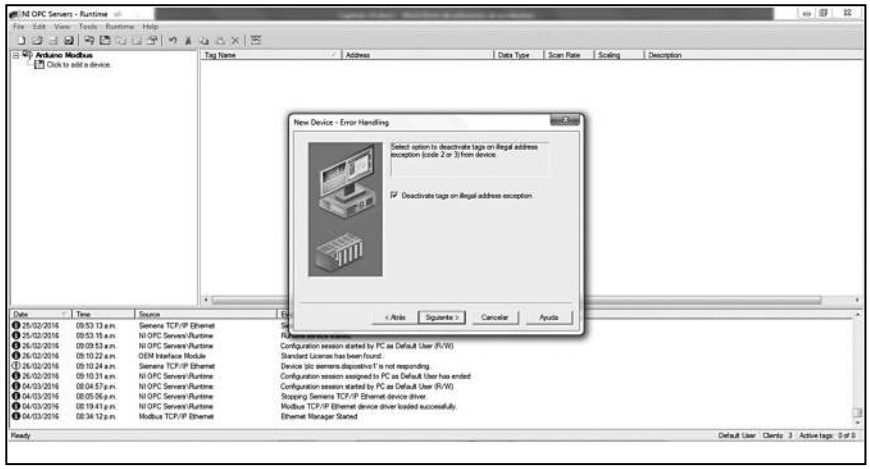


Figura 10.29 Desactivar las tags en direcciones ilegales.

13) Por último, aparecerá la pantalla final del asistente. Si los ajustes que muestra son correctos, dar clic en Finalizar. (Ver figura 10.30.)

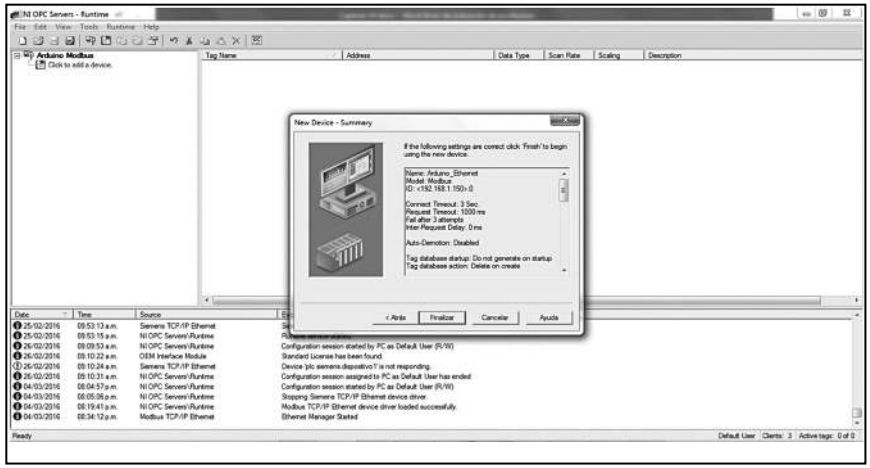


Figura 10.30 Finalizar la configuración del dispositivo.

Una vez cumplidos todos los pasos, en la pantalla principal aparece el dispositivo creado, como lo muestra la figura 10.31.

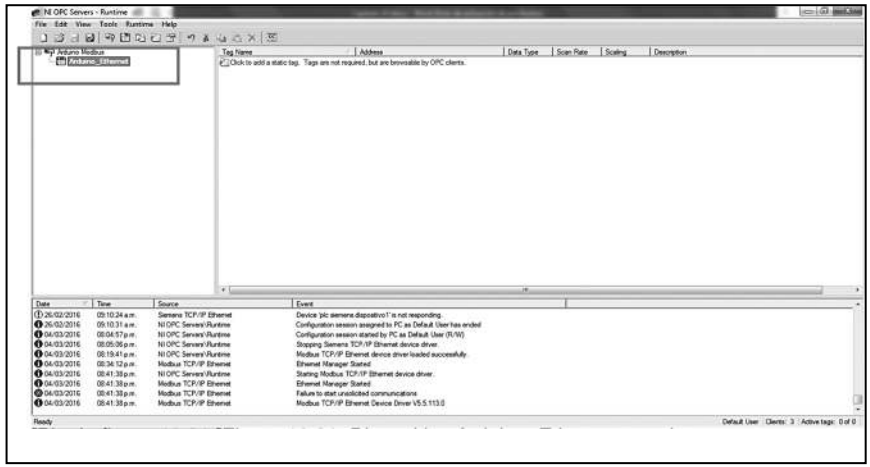


Figura 10.31 Dispositivo Arduino_Ethernet creado.

Una vez que se ha creado el canal y se ha configurado el dispositivo, es posible crear las variables de lectura y escritura; es decir, las tags. A continuación se describe el proceso.

1) En la ventana principal, dar clic con el botón derecho para agregar las tags. (Ver figura 10.32.)

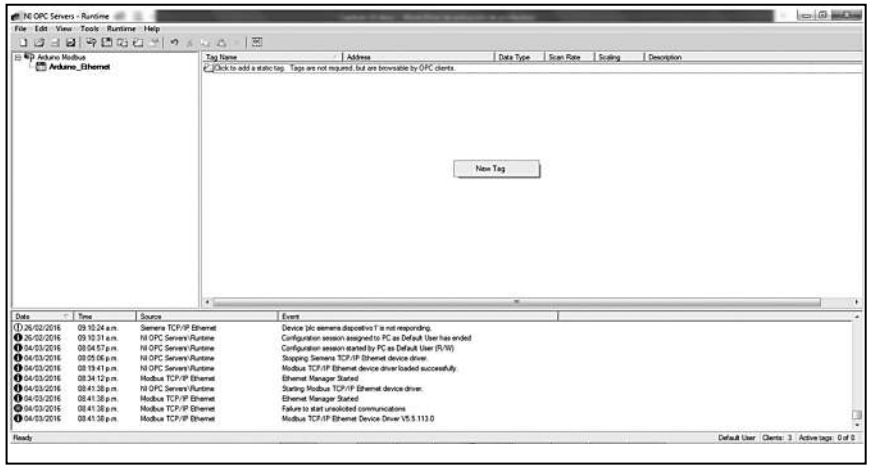


Figura 10.32 Habilitar la opción de agregar tags.

2) Hay que especificar la dirección del registro Modbus de la señal analógica canal 0 (ver figura 10.33). La forma de llegar a este menú es haciendo clic con el botón derecho donde dice New TAG. El canal 0 A0 (señal analógica del Arduino) se va guardar en un registro modbus 40001.

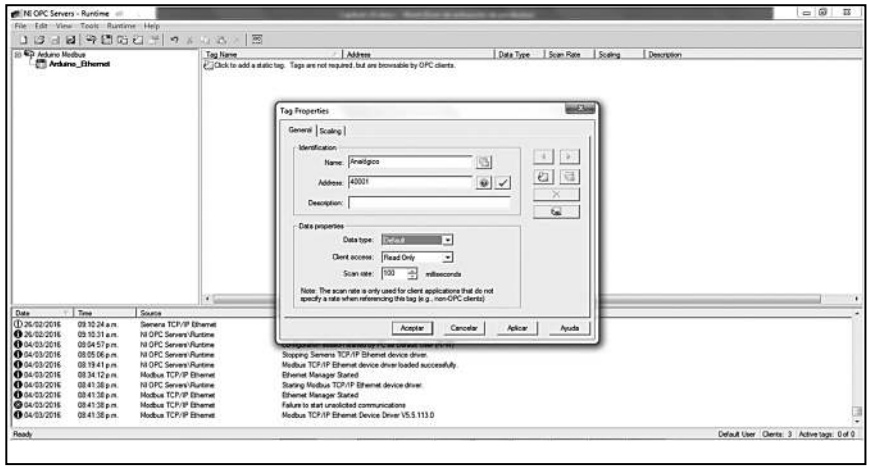


Figura 10.33 Especificar la dirección del registro de Modbus.

3) Una vez configurado, dar clic en el botón Aplicar. (Ver figura 10.34.)

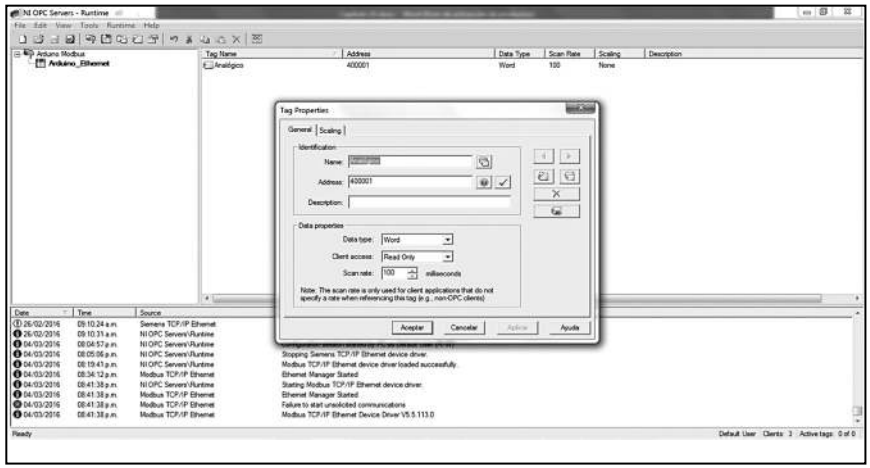


Figura 10.34 Dar clic a Aplicar.

4) Entonces, aparece la ventana con la primera tag creada. (Ver figura 10.35.)

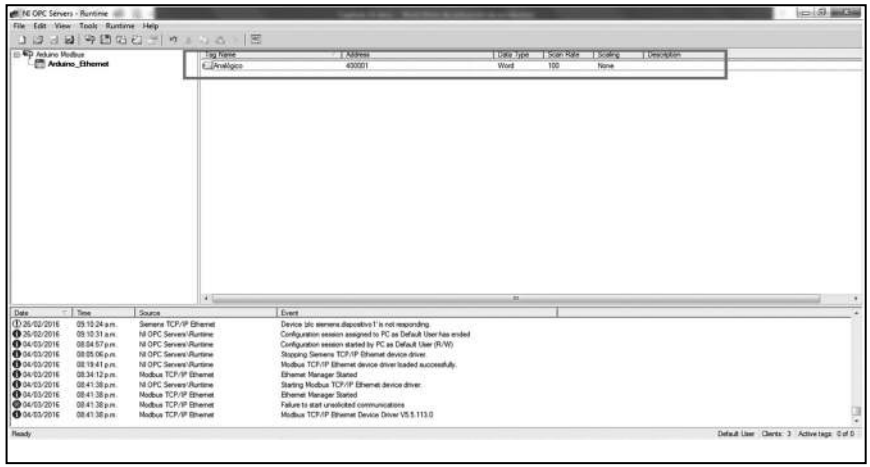


Figura 10.35 Nueva tag creada.

10.8.4 Clientes OPC

Una vez que se tiene la tag, es posible para un cliente OPC monitorear desde esa misma aplicación. (Ver figura 10.36.)

Figura 10.36 Cliente OPC monitoreando el dispositivo Arduino_Ethernet.

Si lo que se quiere es crear una tag de tipo coil, es importante considerar que se debe configurar la dirección del registro 000002 y que es de tipo Boolean. Esta señal se configura a una señal digital 7 de la Arduino. (Ver figura 10.37.)

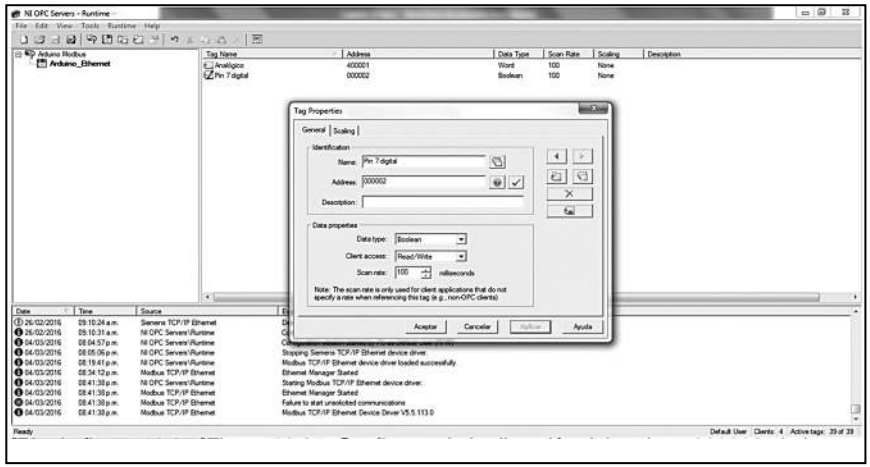


Figura 10.37 Configurando la dirección del registro 000002 al pin digital 7 de Arduino.

Se recomienda hacer una prueba desde el cliente OPC. Para escribir el coil, hay que seleccionar la opción Synchonous Write. (ver figura 10.38). El menú aparecerá dando clic derecho al nombre del dispositivo.

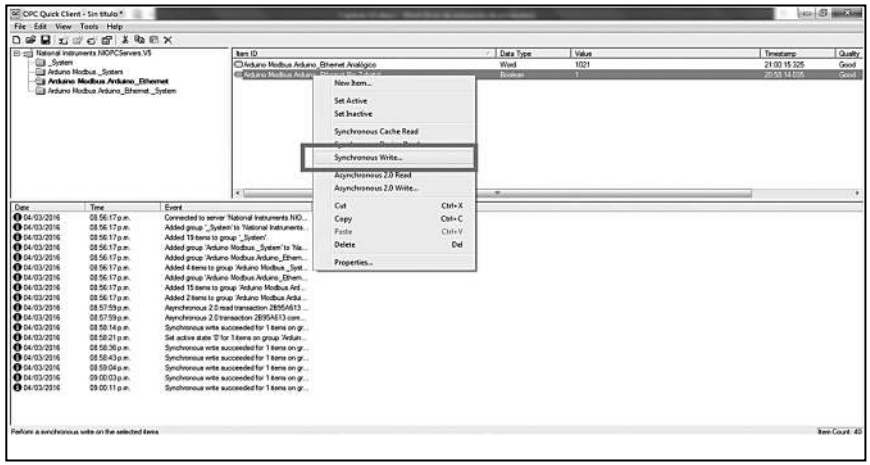


Figura 10.38 Seleccionar Synchonous Write.

Posteriormente, escribir uno (1) o cero (0) para encender o apagar la salida de la Arduino. (Ver figura 10.39.)

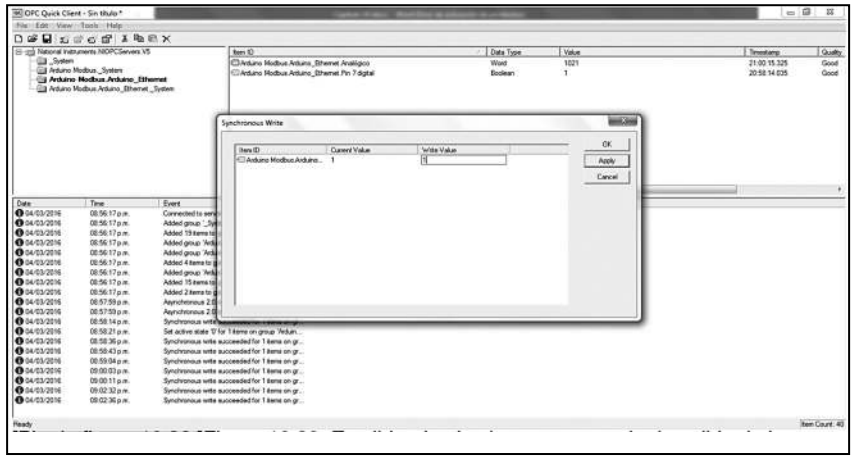


Figura 10.39 Escribir el valor para encender la salida de la Arduino.

10.9 Módulo DSC de National Instruments

El módulo DSC (*Data Logging and Supervisory Control*-Registro de datos y control supervisorio) permite crear un cliente OPC para la implementación de un sistema SCADA utilizando el servidor OPC de National Instruments. De tal forma, se pueden tener varios dispositivos interconectados y acceder a todos ellos desde un cliente OPC.

El módulo DSC puede realizar varias funciones, como se puede ver en la figura 10.40:

- **a)** Almacenar información en una base de datos.
- **b)** Realizar historiales.
- **c)** Ejecutar alarmas y eventos.
- **d)** Hacer networking; es decir, compartir datos en una red.
- **e)** Mantener la seguridad; es decir, restringir alguna modificación o uso de aplicaciones.
- **f)** Actuar como un cliente OPC.

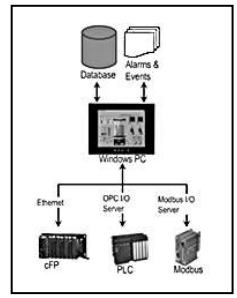


Figura 10.40 Las diferentes funciones del módulo DSC.

Las variables compartidas son el elemento básico del módulo DSC. Están directamente conectadas a las mediciones físicas a través del servidor OPC (ver figura 10.41). Asignan un nombre a una entrada o salida, permitiendo el acceso a estos puntos, simplemente utilizando su nombre en los VI. Un VI es un instrumento virtual que se ejecuta para cumplir una función específica. En este caso son utilizados para accesar a los puertos del hardware y para leer y escribir datos en la memoria. Esto es útil en dos situaciones: cuando se requiere compartir datos en una red y cuando se quiere probar un VI sin tener el hardware presente.

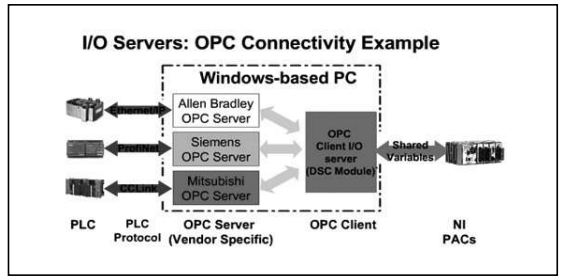


Figura 10.41 Esquema de conectividad del servidor OPC.

10.10 Cliente OPC en .NET

Para crear un Cliente OPC de Visual Studio .NET es necesario crear un proyecto de tipo Measurement Studio, como se muestra a continuación:

1) Crear un proyecto como aplicación de escritorio seleccionando en el menú Archivo el comando Nuevo y, luego, Proyecto, como se muestra en la figura 10.42.

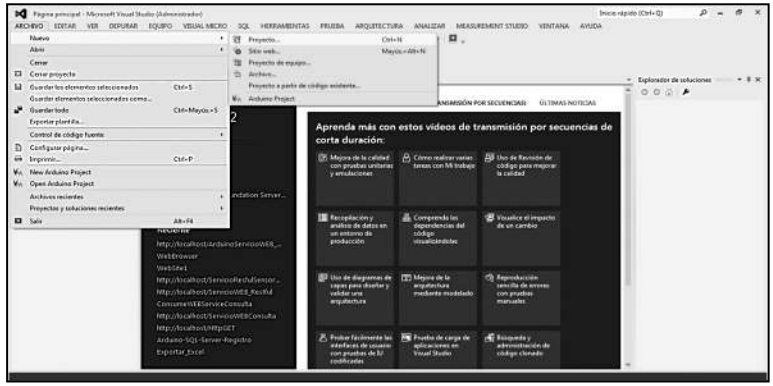


Figura 10.42 Crear un proyecto nuevo en Visual Studio.

2) Después de haber instalado el módulo Measurement Studio de National Instruments para Visual Studio .NET en los tipos de proyectos a crear, debe aparecer un tipo de proyecto de Visual Basic Measurement Studio, que es el seleccionado. (Ver figura 10.43.)

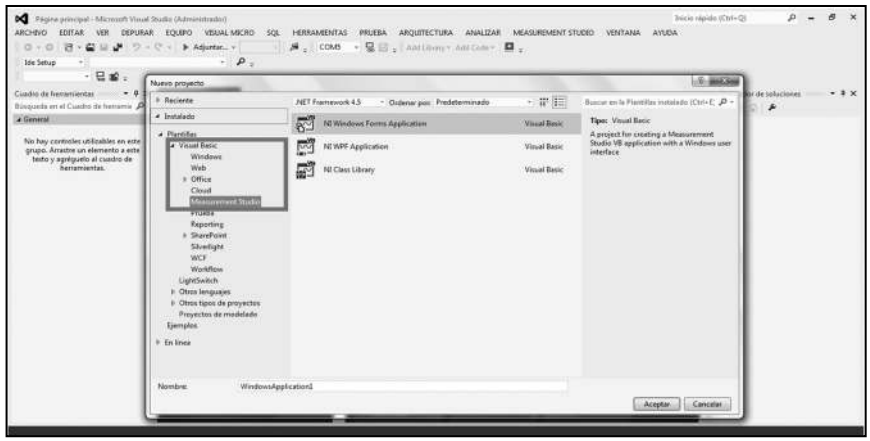


Figura 10.43 Seleccionar proyecto de tipo Visual Basic Measurement Studio.

3) Dentro de los proyectos tipo Measurement hay tres opciones. En este caso se debe elegir el NI Windows Forms Application. Hay que escribir el nombre del proyecto y dar clic en el botón Aceptar. (Ver figura 10.44.)

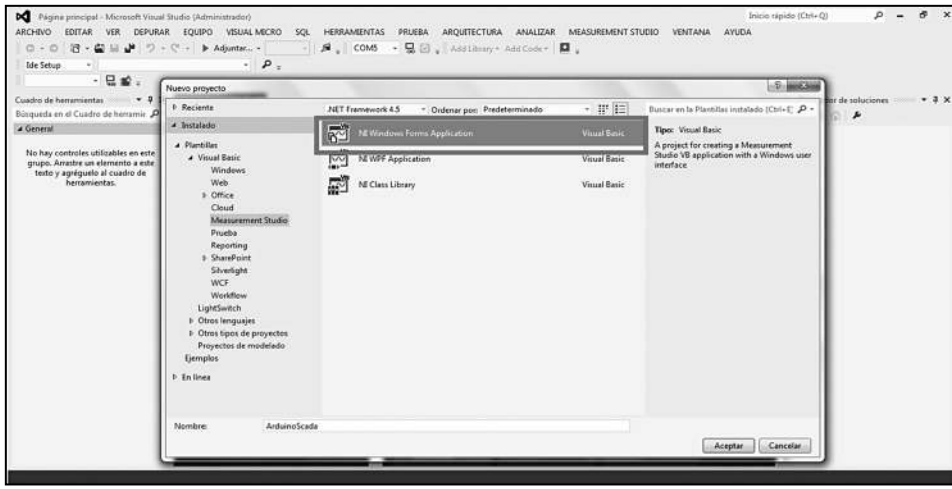


Figura 10.44 Seleccionar el proyecto de Measurement de tipo NI Windows Forms Application.

4) Entonces aparece una ventana de la configuración de las librerías. Es importante activar la comunicación de DataSocket para activar la comunicación de los dispositivos a configurar. (Ver figura 10.45.)

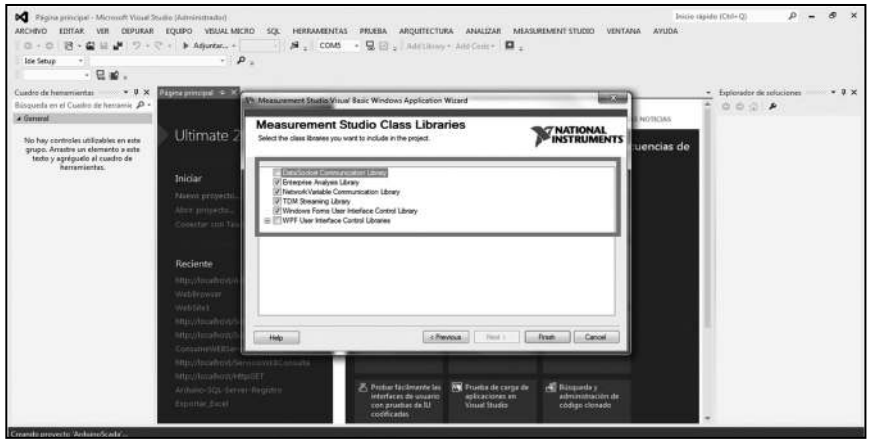


Figura 10.45 Configurar las librerías del proyecto Measurement.

5) Activar la casilla de DataSocket Communication Library y hacer clic en el botón Finish (terminar), como se muestra en la figura 10.46.

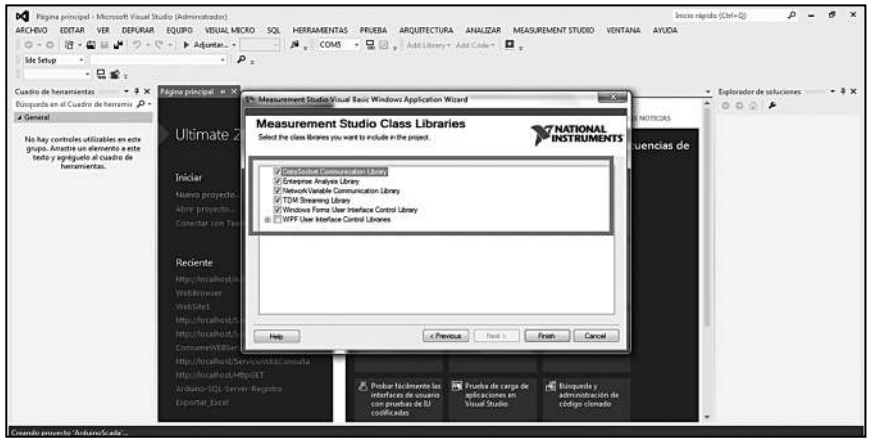


Figura 10.46 Activar la comunicación de DataSocket.

6) Debe aparecer un mensaje indicando que el proyecto se está creando. (Ver figura 10.47.)

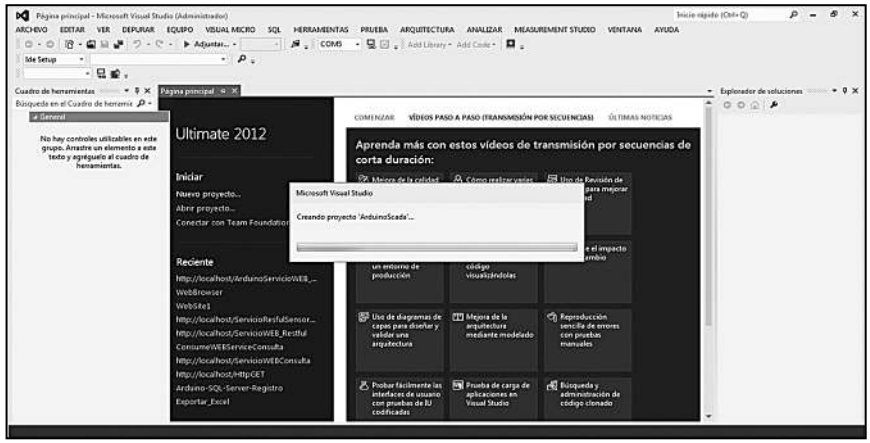


Figura 10.47 Creando el proyecto ArduinoScada.

7) Al terminar, debe aparecer la pantalla de inicio del proyecto, como se muestra en la figura 10.48.

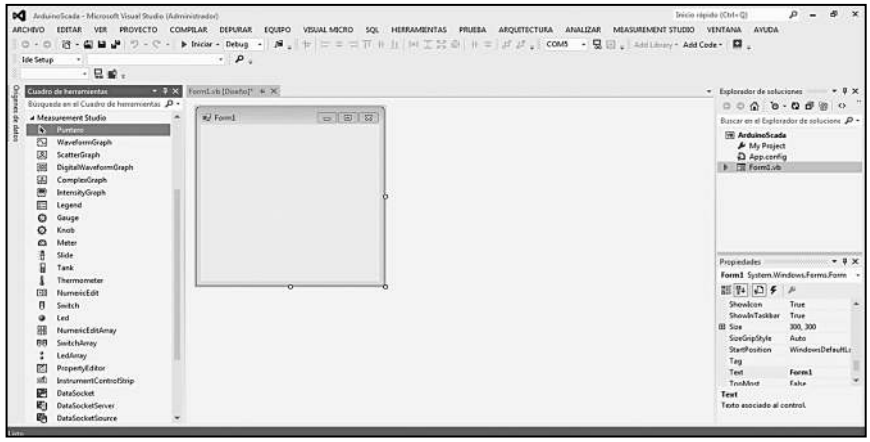


Figura 10.48 Pantalla de inicio del proyecto.

En esa pantalla, del lado izquierdo, aparecerán los controles que permitirán crear el cliente OPC, que están agrupados según su uso. (Ver figura 10.49.)

- Controles utilizados como indicadores para mostrar información:
  - Gauge
  - Meter
  - Tank
  - Thermometer
  - NumericEdit

- Controles utilizados para control de valores y envío de estados lógicos:
  - Knob
  - Slide
  - Switch
- Control utilizado para mostrar un estado lógico:
  - Led
- Controles de tipo array, o matriz, para mostrar varios controles del mismo tipo:
  - NumericEditArray
  - SwitchArray
  - LedArray
- Controles para establecer comunicación y enlace de datos:
  - DataSocket
  - DataSocketServer
  - DataSocketSource
  - NetworkVariableDataSource
  - NetworkVariableBrowserDialog

Figura 10.49 Controles para crear el cliente OPC.

10.11 Implementación del prototipo de la aplicación del sistema de monitoreo y control

En la figura 10.50 se muestran los controles del sistema de monitoreo y control, que son:

1) Lectura de temperatura
2) Lectura de humedad
3) Lectura de señal analógica canal A0
4) Monitoreo de entrada digital pin 2
5) Control de salida digital pin 7
6) Monitoreo cuando la salida digital pin 7 está activa
7) Control de salida PWM pin 9
8) Dirección IP y status del dispositivo

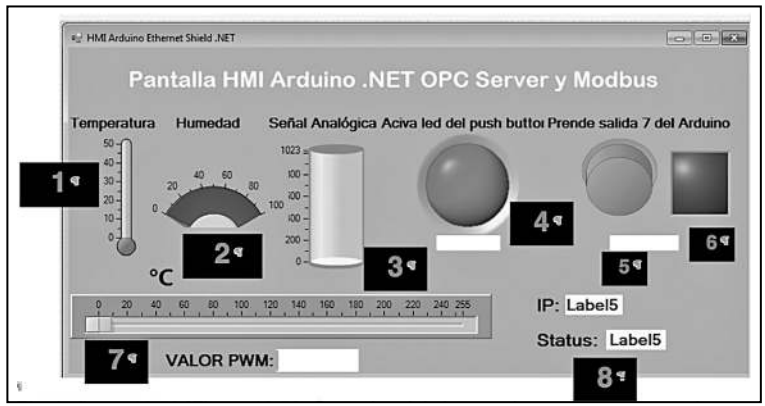


Figura 10.50 Controles del sistema de monitoreo y control.

A continuación se muestra una tabla del direccionamiento de los registros Modbus en el servidor OPC.

Descripción de la señal	Nombre del control	Direccionamiento en el OPC
Temperatura	Thermometer	400003
Humedad	Meter	400004
Señal analógica	Tank	400001
Entrada digital pin 2	Led (Round)	000001
Control Salida digital pin 7	Switch	000002
Monitoreo salida digital 7	Led (Square)	000003
Control PWM pin 9	Slide	400002

10.11.1 Enlazar las direcciones de los registros Modbus

Posteriormente, hay que enlazar las direcciones de los registros Modbus con la aplicación. Para realizar la conexión de los controles mostrados en pantalla se utiliza el control **DataSocket** (ver figura 10.51.) de la siguiente manera:

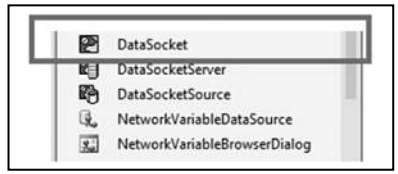


Figura 10.51 Control DataSocket.

1) Arrastrar el control datasocket; en la flecha aparece la opción Select URL, ahí se debe dar clic para realizar la configuración.
2) Hacer clic en la flechita que aparece en el control, en la esquina superior derecha.

(Ver figura 10.52.)

Figura 10.52 Llevar el control DataSocket al formulario.

3) Hacer clic en el vínculo Select URL (ver figura 10.52), después de lo cual aparece una ventana de exploración de archivos. Habrá que ubicar el Servidor OPC de National Instruments.NIOPCServers.V5. (Ver figura 10.53.)

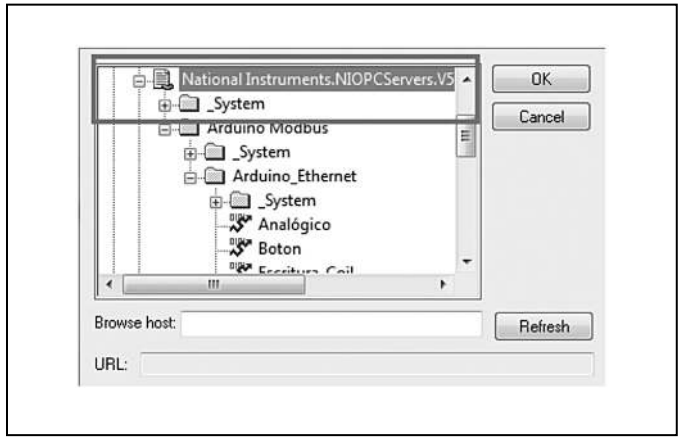


Figura 10.53 Identificar el Servidor NIOPCServers.V5.

4) En esa ruta, buscar el nombre del canal creado previamente en el Servidor OPC; en este caso, Arduino Modbus. (Ver figura 10.54.)

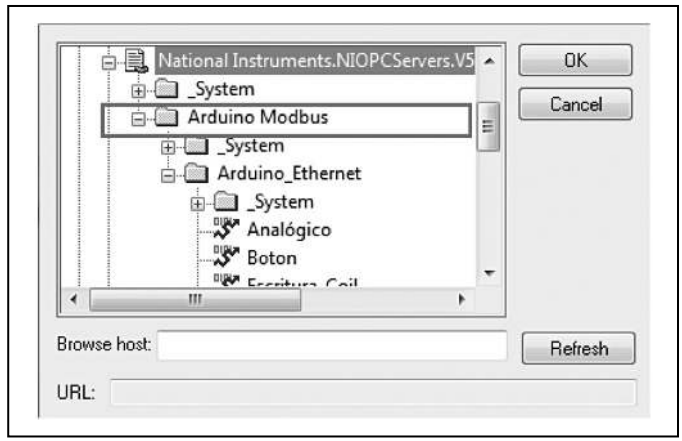


Figura 10.54 Identificar el canal que se creó previamente.

5) Dentro de ese canal, Arduino Modbus, buscar el nombre del dispositivo Arduino_Ethernet. (Ver figura 10.55.)

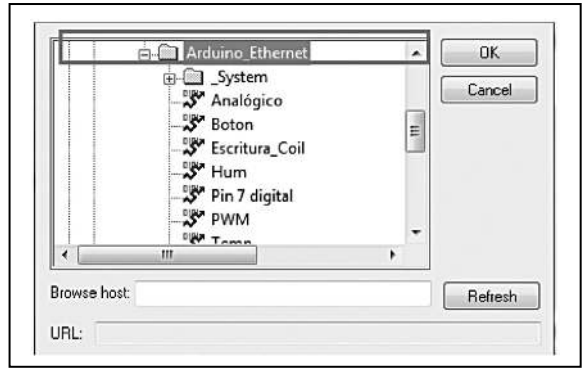


Figura 10.55 Identificar el dispositivo.

6) Seleccionar la tag configurada, de acuerdo a la señal que se va a leer o a escribir. (Ver figura 10.56.)

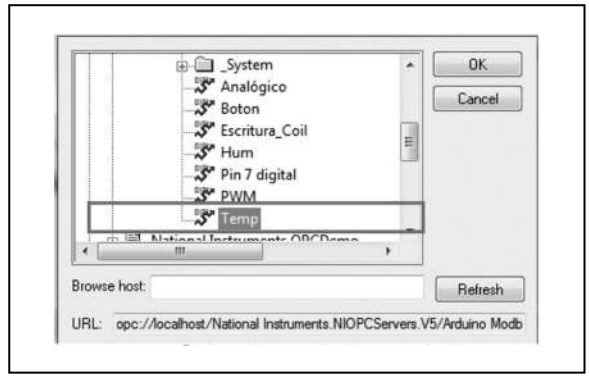


Figura 10.56 Seleccionar la tag de la señal que se utilizará.

7) A partir de ese momento, se tiene vinculada la tag de la señal que se va a leer o a escribir. La dirección de la URL que genera dicha vinculación es la siguiente:

```
opc://localhost/National Instruments.NIOPCServers.V5/Ar-
duino Modbus.Arduino_Ethernet.Temp
```

8) Para vincular cualquier otra tag a la aplicación tanto de lectura como de escritura, repetir los pasos del 1 al 7.

10.11.2 Enlazar los controles de la aplicación con los tags

Una vez que se tiene la ruta de lectura de las tags leídas desde el servidor OPC, hay que enlazar los controles de la aplicación con las tags configuradas, realizando lo que a continuación se explica.

1) Insertar un control timer.
2) Inicializar el timer de la siguiente forma:

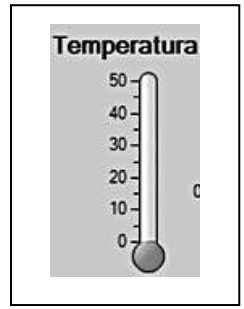
```
Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        Timer1.Enabled = True
    End Sub
```

3) En el evento del timer Timer_tick, capturar lo siguiente:

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs)
Handles Timer1.Tick

        lbltmp.Text = DataSocket6.Data.Value
        Thermometer1.Value = DataSocket6.Data.Value
    End Sub
```

4) Leer el valor del DataSocket6 e igualarlo al control correspondiente donde se va a mostrar el valor. (Recordar control 1 de la figura 10.50.)



A continuación se darán los códigos que confirman el código completo de la aplicación.

Código para definir las variables a utilizar en el programa:

```
Public Class Form1
    Dim salida As Boolean
    Dim boton As Boolean
    Dim led_salida_7 As Boolean
```

Código del timer cuando se carga la forma:

```
    Private Sub Form1_Load(sender As Object, e As Even-
tArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub
```

Evento Timer1_tick para lectura de los registros:

```
    Private Sub Timer1_Tick(sender As Object, e As Even-
tArgs) Handles Timer1.Tick
```

Nótese que la lectura de los registros a través del control DataSocket se iguala al control donde se despliega el dato.

Lectura de la señal analógica canal A0:

```
        Tank1.Value = DataSocket7.Data.Value
```

Lectura de la temperatura:

```
        lbltmp.Text = DataSocket6.Data.Value
        Thermometer1.Value = DataSocket6.Data.Value
```

Lectura de la humedad:

```
        Meter1.Value = DataSocket10.Data.Value
```

Lectura de la IP y el status:

```
lblIP.Text = DataSocket4.Data.Value
lblsts.Text = DataSocket5.Data.Value
```

Lectura de señal de botón de entrada:

```
botón = DataSocket2.Data.Value
```

Lectura de estados lógicos de entrada de botón y salida digital:

```
TextBox1.Text = DataSocket2.Data.Value
TextBox2.Text = DataSocket3.Data.Value
```

Condiciones de comparación para desplegar el estado del led:

```
If boton = True Then
    Led1.Value = True
Else
    Led1.Value = False
End If
```

Lectura de salida 7:

```
led_salida_7 = DataSocket9.Data.Value
```

Condiciones de comparación para desplegar el estado del led:

```
If led_salida_7 = True Then
    Led2.Value = True
Else
    Led2.Value = False
End If

End Sub
```

Evento del Control Switch:

```
Private Sub Switch1_StateChanged(sender As Object, e As
ActionEventArgs) Handles Switch1.StateChanged

If Switch1.Value = True Then
    salida = 1
```

Escritura e instrucción para mandar un estado lógico 1 si se cumple la condición:

```
    DataSocket3.Data.Value = salida
Else
```

Escritura e instrucción para mandar un estado lógico 0 si se cumple la condición:

```
    salida = 0
    DataSocket3.Data.Value = salida
```

```
            End If

        End Sub
```

Evento del control slide:

```
    Private Sub Slide1_ValueChanged(sender As Object, e As
    EventArgs) Handles Slide1.ValueChanged
```

El valor del slide se iguala a la caja de texto para que muestre su valor conforme se vaya moviendo:

```
            TextBox3.Text = Slide1.Value.ToString
```

Escritura del valor del slide para se escriba en el registro PWM:

```
            DataSocket8.Data.Value = Slide1.Value.ToString
        End Sub

    End Class
```

La figura 10.57 muestra la aplicación en funcionamiento.

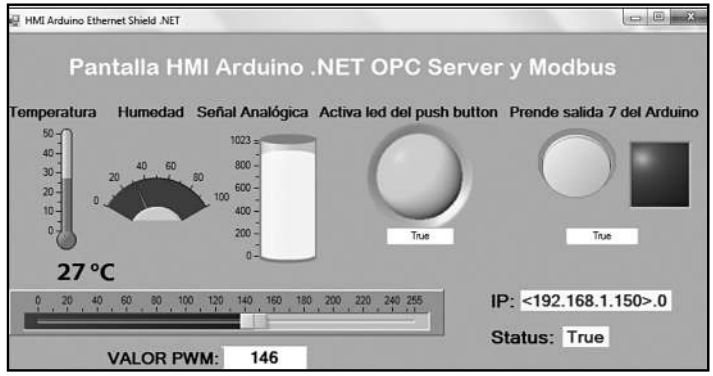


Figura 10.57 Aplicación del servidor OPC.

10.12 Control y monitoreo desde una página web

Ahora se puede crear un proyecto web en Visual Studio seleccionando en el menú Archivo los comandos Nuevo Sitio web, como se aprecia en la figura 10.58.

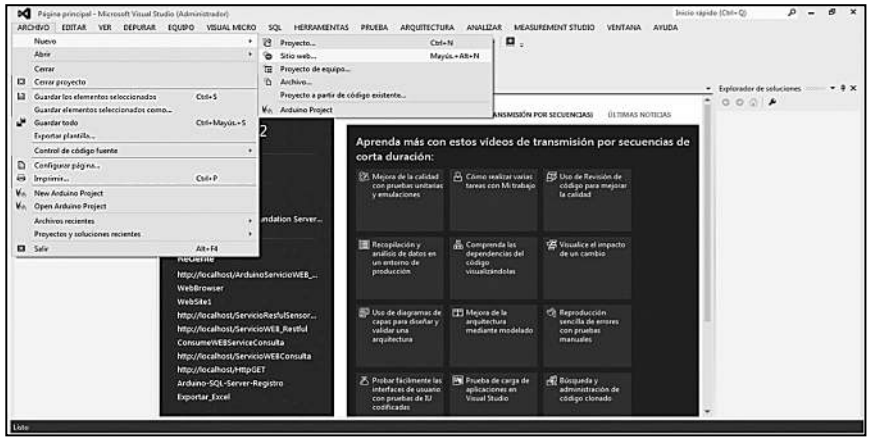


Figura 10.58 Crear un sitio web.

Después, se selecciona un proyecto de tipo NI ASP.NET Web Site, como se puede apreciar en la figura 10.59.

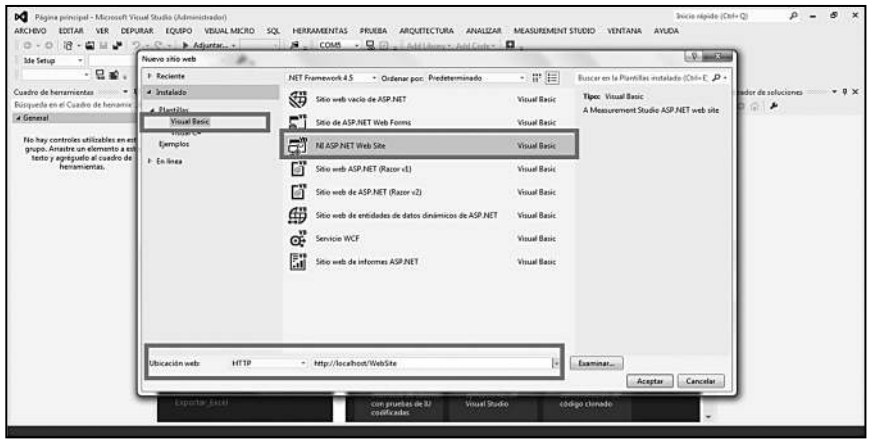


Figura 10.59 Seleccionar el tipo de proyecto para el sitio web.

Recordar que el tipo de proyecto es de tipo web, por lo que se crea en localhost a través del protocolo HTTP. En la pantalla ASP.NET se verá como se muestra en la figura 10.60.

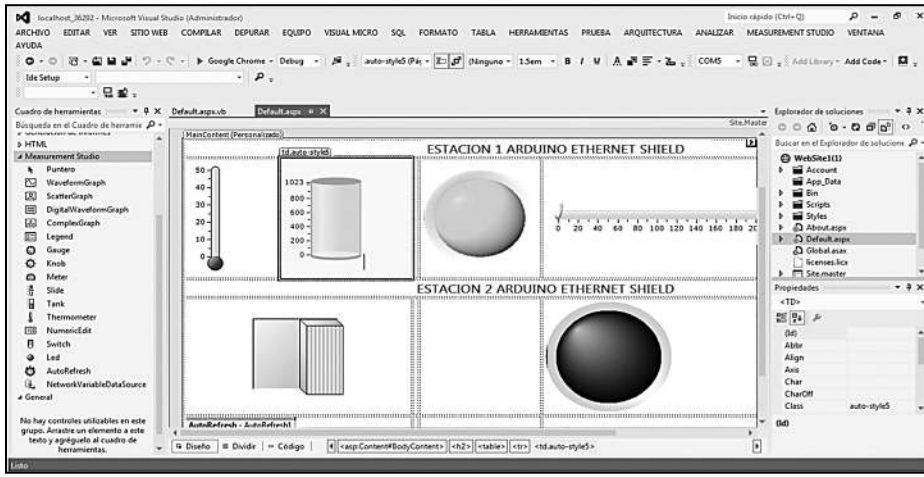


Figura 10.60 Pantalla en ASP.NET.

Cuando se crea una aplicación de tipo web, los controles que se despliegan en Measurement Studio son los siguientes (Ver figura 10.61):

- Puntero
- WaveformGraph
- ScatterGraph
- DigitalWaveformGraph
- ComplexGraph
- Legend
- Gauge
- Knob
- Meter
- Slide
- Tank
- Thermometer
- NumericEdit
- Switch
- Led
- AutoRefresh
- NetworkVariableDataSource

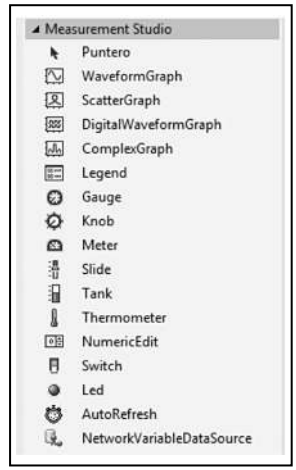


Figura 10.61 Controles de monitoreo y de comunicación.

La figura 10.62 explica la aplicación de monitoreo y control.

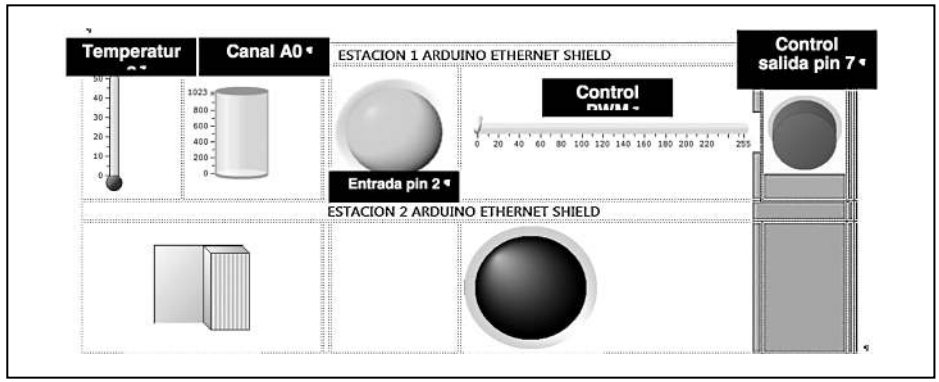


Figura 10.62 Explicación de la aplicación.

La figura 10.63 muestra el control NetworkVariableDataSource.

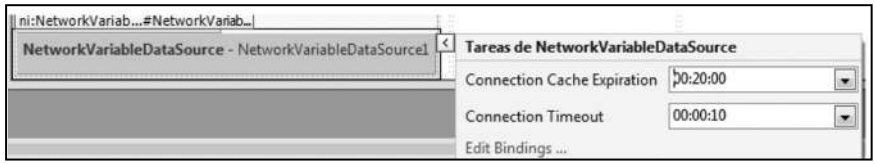


Figura 10.63 El control Network VariableDataSource.

Hacer clic en la opción Edit Bindings; aparecerá la ventana de la figura 10.64. Es necesario abrir este menú porque permite enlazar los controles con los tags creados en el servidor OPC.

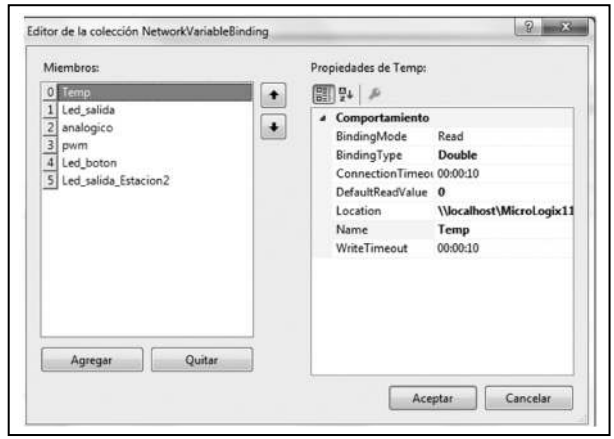


Figura 10.64 Ventana de Edit Bindings.

En la configuración Location, buscar la ruta de la tag configurada previamente en el Servidor OPC. Por ejemplo, la ruta de la configuración de enlace para leer la tag de temperatura es:

```
\\localhost\MicroLogix1100\OPC1\Arduino Modbus\Arduino_
Ethernet\Temp
```

Para cada uno de los controles se debe configurar el vínculo de dónde se encuentra cada tag configurada.

El control **AutoRefresh** (ver figura 10.65) permite a la aplicación estar actualizando la página cada cierto tiempo.

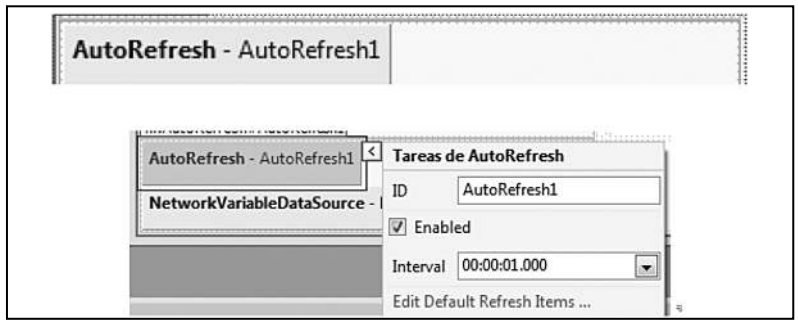


Figura 10.65 El control AutoRefresh.

Activar la opción Enabled (ver figura 10.66); luego, editar los controles que se van a estar actualizando cada cierto tiempo. La figura 10.66 muestra, en el recuadro estos controles.

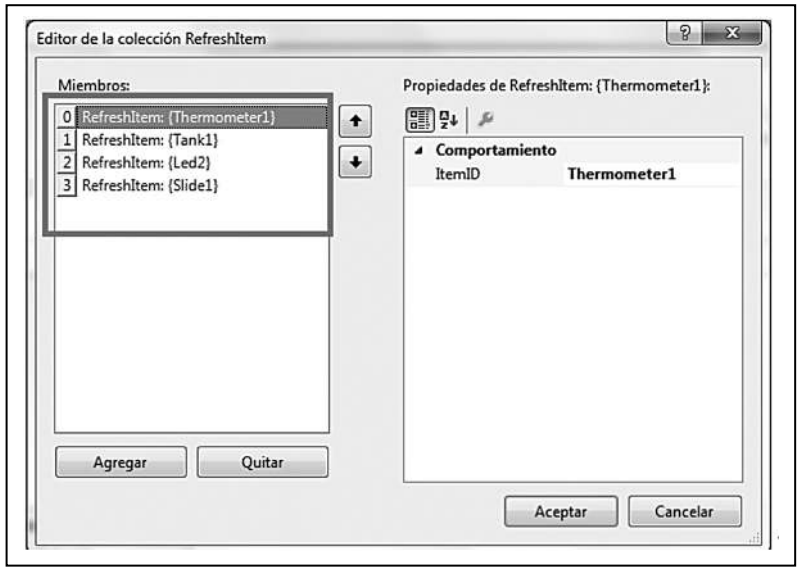


Figura 10.66 Controles para actualizar periódicamente.

El código de la aplicación es el siguiente:

```
Protected Sub AutoRefresh1_Refresh(sender As Object, e As
RefreshEventArgs) Handles AutoRefresh1.Refresh
```

Los valores de la propiedad Bindings se refieren al orden en que se configuran en el control NetworkVariableDataSource. En este caso:

```
        Thermometer1.Value = Ctype(NetworkVariableData-
Source1.Bindings(0).GetValue(), Double)
        Tank1.Value = NetworkVariableDataSource1.Bin-
dings(2).GetValue()
        Led2.Value = NetworkVariableDataSource1.Bin-
dings(4).GetValue

        If Led2.Value = True Then
            Led2.Value = True
        Else
            Led2.Value = False
        End If
    End Sub


Protected Sub Switch1_StateChanged(sender As Object, e As
ActionEventArgs) Handles Switch1.StateChanged

        If Switch1.Value = True Then
            NetworkVariableDataSource1.Bindings(1).SetVa-
lue("1")
```

```
        Else
            NetworkVariableDataSource1.Bindings(1).SetVa-
lue("0")
        End If
    End Sub


Protected Sub Slide1_AfterChangeValue(sender As Object, e
As AfterChangeNumericValueEventArgs) Handles Slide1.Af-
terChangeValue

        NetworkVariableDataSource1.Bindings(3).SetVa-
lue(Slide1.Value)
    End Sub


Protected Sub Switch2_StateChanged(sender As Object, e As
ActionEventArgs) Handles Switch2.StateChanged

        If Switch2.Value = True Then
            NetworkVariableDataSource1.Bindings(5).SetVa-
lue("1")
            Led1.Value = True
        Else
            NetworkVariableDataSource1.Bindings(5).SetVa-
lue("0")
            Led1.Value = False
        End If
    End Sub
```

La figura 10.67 muestra la aplicación en funcionamiento; es decir, la página web en la que se tiene más de un Arduino para monitoreo en el mismo sitio.

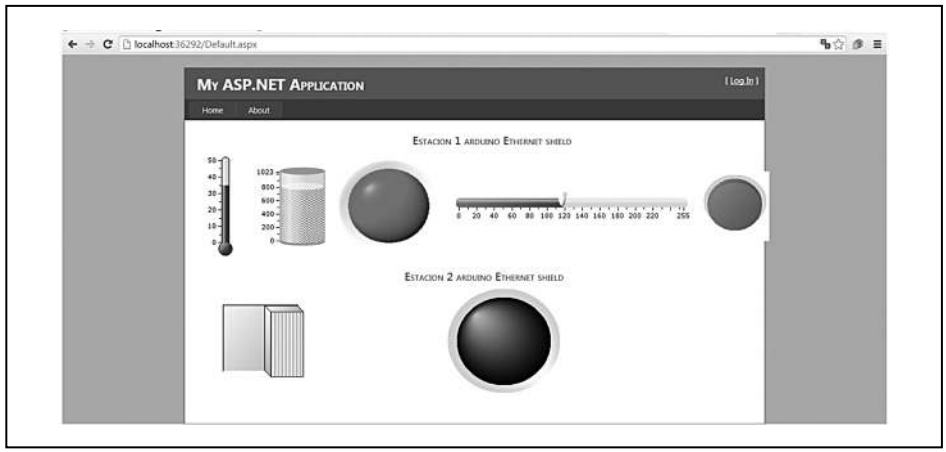


Figura 10.67 Así funciona la aplicación.

10.13 Resumen

En este capítulo se explicó cómo crear una aplicación cliente OPC utilizando la tecnología de .NET. La ventaja de esta aplicación es la configuración de una tarjeta Arduino conectada a una red industrial, donde la Arduino puede interactuar en una red con otros dispositivos industriales, como controladores lógicos programables, a través del servidor OPC, y tener múltiples conexiones de dispositivos. Otra ventaja es que se pueden monitorear y controlar varios dispositivos desde una misma pantalla HMI. Esto permite control absoluto. Además, utilizando las herramientas vistas en capítulos anteriores, se pueden desarrollar proyectos de mayor complejidad para controlar dispositivos de forma remota desde una página web.

Aprender a realizar aplicaciones donde interactúen todas las tecnologías, como bases de datos, reportes o interfaces, permitirá desarrollar proyectos aplicables en el área de la Mecatrónica y, como ya se vio, para redes industriales y sistemas SCADA.

En el siguiente capítulo se desarrollarán aplicaciones para monitorear la localización y el rastreo de dispositivos a través de la implementación de módulos GPS y GSM/GPRS.

10.14 Problemas

1. Cree una interfaz de monitoreo para una estación meteorológica con sensor de luz, de temperatura LM35 y de presión barométrica.
2. Implemente una interfaz de control con botones y encendido de tres relevadores, cada uno con su foco de potencia.
3. Desarrolle una interfaz para controlar el sentido de un motor de corriente directa con tres botones (adelante, atrás y detener).
4. Implemente una interfaz para controlar el sentido de un motor de corriente directa con cajas de texto para determinar la posición.
5. Realice una interfaz para control de la velocidad de un motor de corriente directa mediante el control slider.
6. Cree una interfaz para control de la posición de un servomotor con botones y con el control slider.
7. Desarrolle una interfaz para monitoreo con sensor de movimiento PIR; si se detecta un movimiento, debe enviar por modbus una activación.
8. Desarrolle un proyecto de tipo web para controlar un salida conectando un relevador, al menos configurar dos tarjetas Arduino conectadas en red.
9. Crear un proyecto de tipo web para monitorear las lecturas de sensores de temperatura y humedad; configure al menos dos tarjetas Arduino que sean monitoreadas en la misma página.
10. Planee un proyecto de tipo web para guardar las lecturas de los sensores en una base de datos.

11. Desde una aplicación de escritorio cree un servicio web para consultar las lecturas generadas.
12. Cree un proyecto de tipo web con al menos tres páginas, controle y monitoree un dispositivo en cada una.
13. Implemente una página web que permita graficar en tiempo real las lecturas de un sensor de temperatura y humedad.
14. Crear una aplicación que registre las lecturas de un sensor de temperatura y humedad en una micro SD.
15. Genere una aplicación de escritorio por modbus que lea los valores guardados y que los muestre en pantalla de monitoreo.
16. Conecte un sensor de huella digital que cuando detecte la huella correcta envíe la respuesta a una página web.

Capítulo 11

Rastreador móvil por medio de GSM/GPRS y GPS

Objetivos

En este capítulo se aplicarán conocimientos de comunicación entre GPS, GSM y GPRS utilizando el sistema de Arduino para crear módulos de rastreo remoto de dispositivos en movimiento.

11.1 Introducción

En este capítulo se desarrollarán proyectos relacionados con la comunicación entre módulos GPS (sistema de posicionamiento global), el módulo GSM (comunicación por mensajes de texto) y GPRS (servicio general de paquetes vía radio). Además, estos tres módulos se integrarán a la tarjeta Arduino para desarrollar algunos proyectos relacionados con el rastreo remoto de dispositivos en movimiento, ya sea vehículos o cualquier otro dispositivo.

Los proyectos a desarrollar en este capítulo son los siguientes:

- Recibir coordenadas del módulo GPS y mostrarlas en una pantalla LCD.
- Capturar una ubicación a través de Google Maps.
- Enviar coordenadas capturadas por medio de mensaje de texto.
- Rastrear coordenadas y enviarlas a Freeboard por medio de Dweet.io.
- Enviar coordenadas por medio de la red de datos GPRS.

11.2 Requerimientos de software y hardware

El hardware que se necesita para desarrollar los proyectos anteriormente mencionados es el siguiente:

- Arduino UNO
- Shield GPS de Spark Fun
- Shield Ethernet
- Shield GSM/GPRS SIM 900
- Módulo SIM de alguna compañía de telefonía celular
- Pantalla LCD (16x2)

Con respecto al software, se requiere de la Librería: TinyGPS.h para la configuración de la comunicación del módulo GPS.

11.3 Configuración del hardware

A continuación se describen los pasos para preparar el módulo que capturará las coordenadas por el módulo GPS.

1) Conectar el Shield GPS (ver figura 11.1) a la tarjeta Arduino.
2) Conectar la pantalla LCD a la tarjeta.

11.4 Módulo GPS para recibir coordenadas

Figura 11.1 El escudo GPS de Spark Fun.

Para configurar la comunicación del módulo GPS, ubicar el pin de configuración que se muestra en la figura 11.2.

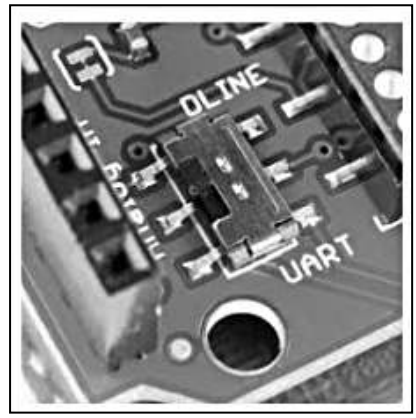


Figura 11.2 Pin de comunicación del módulo GPS.

Se podrá constatar que el modo DLINE permite que la Arduino se comunique con el módulo a través de la librería software Serial en cualquier pin digital. Asimismo, el modo UART permite que el módulo GPS se comunique con la Arduino por los pines 0 y 1.

Para este proyecto se utilizará el modo UART para establecer la comunicación. Sin embargo, es importante saber que, para descargar el programa en la tarjeta, hay que mover el jumper a modo DLINE y, después de hacer la descarga, cambiar el jumper a modo UART.

Posteriormente, hay que cargar el siguiente programa a la placa Arduino:

```
#include <TinyGPS.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

// crear instancia del módulo GPS
TinyGPS gps;
void getgps(TinyGPS &gps);

void setup()
{
Serial.begin(4800);
lcd.begin(16,2);
}

void getgps(TinyGPS &gps)
{
float latitude,longitude;

gps.f_get_position(&latitude,&longitude);

lcd.setCursor(0,0);
lcd.print(«Lat:»);
lcd.print(latitude,5);
lcd.print(« «);
lcd.setCursor(0,1);
lcd.print(«Long:»);
lcd.print(longitude,5);
lcd.print(« «);
delay(3000);
lcd.clear();
}


void loop()
{
byte a;
if (Serial.available() > 0 )
{
a = Serial.read();
if(gps.encode(a))
{
getgps(gps);
}
}
}
```

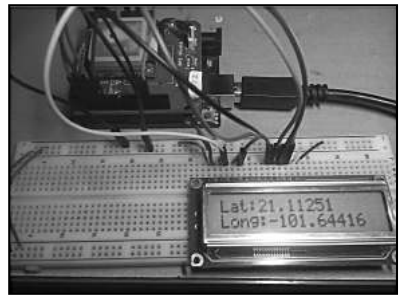


Figura 11.3 Pantalla LCD mostrando latitud 21.11251 y longitud -101.64416.

Para que esta ubicación se despliegue en Google Maps, hay que teclear la siguiente dirección en Internet: https://www.google.com.mx/maps. (Ver figura 11.4.)

Figura 11.4 Tecleando coordenadas en Google Maps.

De tal forma, en la página de Internet se mostrará la ubicación exacta de estas coordenadas, como lo muestra la figura 11.5.

Figura 11.5 Coordenadas recibidas del GPS en la pantalla LCD y Google Maps.

11.5 Comunicación entre el módulo GSM/GPRS y el módulo GPS

En el capítulo 9 se describió la puesta en marcha del módulo GSM/GPRS (ver figura 11.6). Ahora se utilizará este módulo para enviar los datos obtenidos del módulo GPS a través de la red de mensajes de texto GSM.

Figura 11.6 Módulo GSM/GPRS.

En este proyecto, el módulo GSM/GPRS enviará, cada cierto tiempo, la ubicación del GPS a un teléfono celular mediante un mensaje de texto SMS. (Ver figura 11.7.)

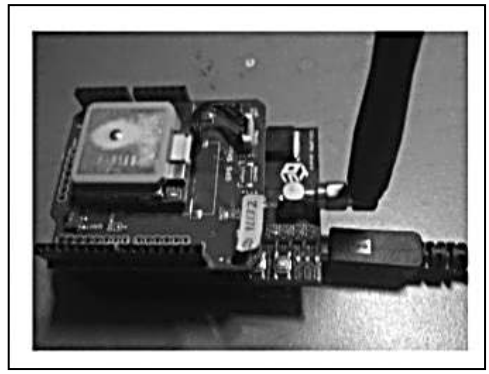


Figura 11.7 Módulo GPS y GSM/GPRS trabajando en conjunto.

La instrucción para enviar el mensaje de texto es la siguiente:

```
#include <TinyGPS.h>
#include <SoftwareSerial.h>
```

```
SoftwareSerial cell(2,3);

float la,lon;

TinyGPS gps;
void getgps(TinyGPS &gps);

void setup() {
 //comando para encender el módem SIM900
 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2000);
 digitalWrite(8,LOW);
 delay(3000);

 //inicia la comunicación serial con el módem SIM900 a
19200
 cell.begin(19200);

 // Open serial communications
 Serial.begin(4800);
}

void loop()
{
  byte a;
if (Serial.available() > 0 ) // if there is data coming
into the serial line
{
a = Serial.read();
if(gps.encode(a))
{
getgps(gps);
envioSMS();
delay(30000);   //tiempo entre cada envío de mensaje
}
}
}
void getgps(TinyGPS &gps)
{
gps.f_get_position(&la,&lon);
}

void envioSMS()
{
    cell.print("AT+CMGF=1\r");
    delay(100);
    cell.println("AT+CMGS=\"4772236196\"");
    delay(100);
    cell.println(«Latitud:»);
    cell.println(la,5);
    cell.println(«Longitd:»);
```

```
        cell.println(lon,5);
        delay(100);
        cell.println((char)26);
        delay(100);
    }
```

La figura 11.8 muestra cómo un teléfono celular recibe un mensaje SMS del módulo GSM.

Figura 11.8 Mensaje con la ubicación del GPS.

11.6 Monitoreo remoto y rastreador móvil

En este proyecto, primeramente se realizan pruebas utilizando el módulo Ethernet (ver figura 11.9) para enviar las coordenadas al sitio de Internet de las cosas llamado Freeboard.

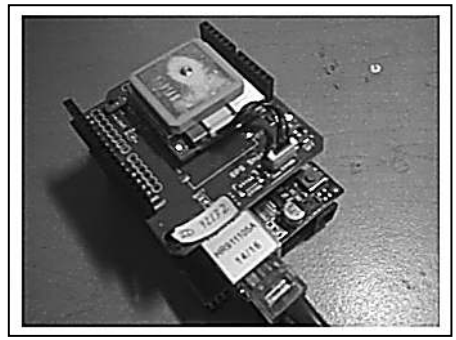


Figura 11.9 Módulo Ethernet y GPS trabajando en conjunto.

La instrucción que se siguió para enviar los datos al sitio remoto es la siguiente:

```
#include <SPI.h>
#include <Ethernet.h>
#include <TinyGPS.h>

float la,lon;

TinyGPS gps;
void getgps(TinyGPS &gps);

byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xFE, 0x40 };
IPAddress ip(192,168,1,150);
EthernetClient client;

char server[] = "www.dweet.io";

void setup() {
   Serial.begin(4800);

    if (Ethernet.begin(mac) == 0) {
    Serial.println("La Dirección IP se asigna por DHCP");
    Ethernet.begin(mac, ip);
  }

    Serial.print("IP address: ");
  Serial.println(Ethernet.localIP());

  delay(1000);
  Serial.println("Conectando...");
}

void loop()
{
  byte a;
if (Serial.available() > 0 )
{
a = Serial.read();
```

```
if(gps.encode(a))
{
getgps(gps);
httpRequest();
}
}
}

void getgps(TinyGPS &gps)
{
gps.f_get_position(&la,&lon);
}



void httpRequest(){
  String lati = String((float)la,5);
  String longi = String((float)lon,5);

  if (client.connect(server, 80)) {
    if (client.connected()) {
      Serial.println("conectado");

      client.println("POST /dweet/for/gps?la=" + lati +
"&lon=" + longi + " HTTP/1.1");
      client.println("Host: www.dweet.io");
      client.println("Connection: close");
      client.println();
    }
    else {
      Serial.println("fallo la conexion");
    }
      while (client.connected()) {
      while (client.available()) {
      char c = client.read();
      Serial.print(c);
      }
    }
    if (!client.connected()) {
      Serial.println();
      Serial.println("desconectado.");
      client.stop();
    }
  }
  delay(5000);
}
```

La figura 11.10 muestra los datos en la página de Freeboard que se enviaron de forma remota a través del módulo Ethernet Shield.

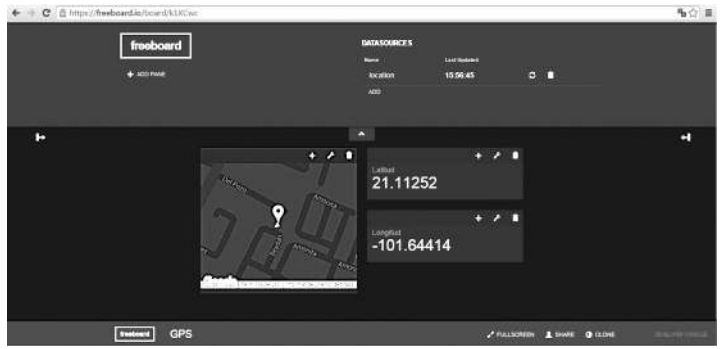


Figura 11.10 Datos del GPS en la página de Freeboard.

11.7 Rastreador remoto

En este punto se explicará cómo desarrollar la página de monitoreo para enviar las coordenadas del módulo GPS al sitio Freeboard.io utilizando como medio de comunicación el Shield GSM/GPRS y la red de datos GPRS. (Ver figura 11.11.)

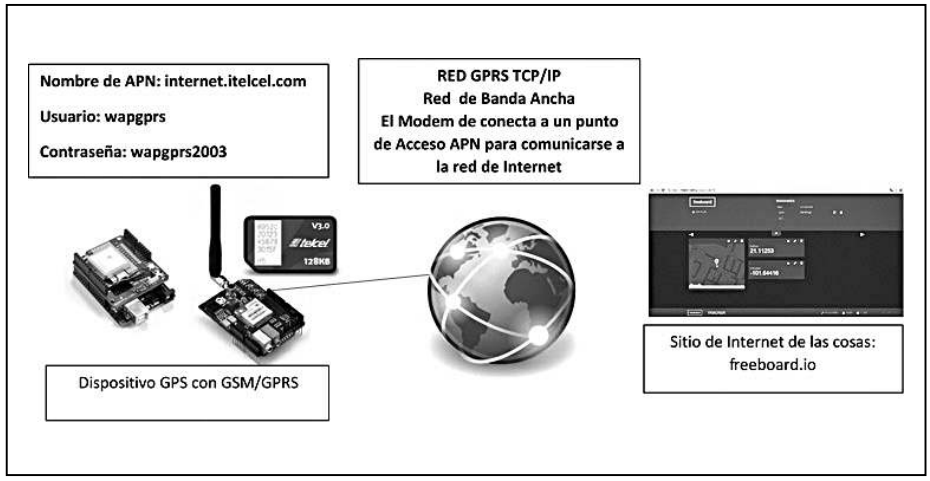


Figura 11.11 Monitoreo remoto mediante red GSM/GPRS.

Los pasos que se llevaron a cabo para realizar la comunicación entre todos estos elementos, con el fin de lograr el monitoreo remoto, son los que se describen a continuación:

1) Conectarse a la red de datos GPRS. Para ello, hay que saber cuáles son los datos del APN (*Access Point Name)* o nombre del punto de acceso. Todo dispositivo

móvil necesita un APN para acceder a una red de datos basada en el Servicio General de Radio por Paquetes (GPRS, por sus siglas en inglés: *General Packet Radio Service*). En este caso se utilizaron los datos que se muestran más abajo.

```
Nombre: Telcel INTERNET
APN: internet.itelcel.com
Proxy: dejar en blanco
Puerto: dejar en blanco
Nombre de Usuario: wapgprs
Contraseña: wapgprs2003
Servidor: dejar en blanco
MMSC: dejar en blanco
Proxy de MMS: dejar en blanco
Puerto de MMS: dejar en blanco
MCC: 334
MNC: 020
Tipo de autenticación: PAP
Tipo de APN: default
```

Para este proyecto solo se necesitan los datos del nombre del punto de acceso APN, el usuario y la contraseña.

2) Hacer la programación. El programa de la aplicación es el siguiente:

```
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial mySerial(2,3);
#include <TinyGPS.h>

float la,lon;

TinyGPS gps;
void getgps(TinyGPS &gps);

void setup()
{
 mySerial.begin(19200);
 Serial.begin(4800);
 delay(500);
}

void loop()
{
 byte a;
if (Serial.available() > 0 )
{
a = Serial.read();
if(gps.encode(a))
{
getgps(gps);
tcp();
delay(10000);
```

```
}
}
}


void getgps(TinyGPS &gps)
{
gps.f_get_position(&la,&lon);
}

//función para comunicación GPRS
  void tcp(){
  String lati = String((float)la,5);
  String longi = String((float)lon,5);

 mySerial.println("AT+CGATT?"); //Establece comunicación
con la red GPRS
 delay(100);
 ShowSerialData();

 //verificar el usuario y contraseña del Chip de acuerdo a
la compañía de telefonía
 mySerial.println("AT+CSTT=\"internet.itelcel.
com\",\"wapgprs\",\"wapgprs2003\"");   //Configuración de
parámetros de Punto de acceso
 delay(1000);
 ShowSerialData();

 mySerial.println("AT+CIICR");                //Se obtie-
ne la conexión de la red
 delay(300);
 ShowSerialData();

 mySerial.println("AT+CIFSR");                //Busca la
dirección IP de la red
 delay(2000);
 ShowSerialData();

 mySerial.println("AT+CIPSPRT=0");
 delay(3000);
 ShowSerialData();

 //Parámetros hacia donde se van a enviar los datos di-
rección IP y el Puerto del servidor
 mySerial.println("AT+CIPSTART=\"TCP\",\"dweet.
io\",\"80\"");
 delay(2000);
 ShowSerialData();
 Serial.println();

 //Se envía la cadena
 mySerial.println("AT+CIPSEND");
 delay(4500);
 ShowSerialData();
```

```
 //Petición GET para enviar los datos
 mySerial.print("POST /dweet/for/gprs?");
 delay(500);
 ShowSerialData();

 mySerial.print("latitud=");
 mySerial.print(lati);
 delay(10);
 ShowSerialData();

 mySerial.print("&longitud=");
 mySerial.print(longi);
 delay(10);
 ShowSerialData();

 mySerial.print(" HTTP/1.1\r\n");                    //
cabecera del protocolo HTTP GET
 delay(500);
 ShowSerialData();

 mySerial.print("Host: www.dweet.io\r\n");
 delay(500);
 ShowSerialData();

 mySerial.print("Connection: close");             //se
cierra la conexión
 mySerial.print("\r\n");
 mySerial.print("\r\n");
 delay(500);
 ShowSerialData();

 mySerial.println((char)26);         //envío de los datos
es igual a control-z
 delay(500);
 mySerial.println();
 ShowSerialData();

 mySerial.println("AT+CIPCLOSE");  //se cierra la conexión
 delay(100);
 ShowSerialData();
}

void ShowSerialData()
{
 while(mySerial.available() != 0)
   Serial.write(mySerial.read());
}
```

Dentro de la programación, hay que establecer una función para enviar el texto. El formato de instrucción es el siguiente:

```
//función para enviar cadena de texto por GPRS mediante
TCP/IP cuando de presiona la tecla se envía la cadena
void mensajeGPRS()
{
 mySerial.println("AT+CGATT?");
//Establece comunicación con la red GPRS
 delay(100);
 ShowSerialData();
 mySerial.println("AT+CSTT=\"internet.itelcel.
com\",\"wapgprs\",\"wapgprs2003\"");         //Configura-
ción de parámetros de Punto de acceso
 delay(1000);
 ShowSerialData();
 mySerial.println("AT+CIICR");
//Se obtiene la conexión de la red
 delay(300);
 ShowSerialData();
 mySerial.println("AT+CIFSR");
//Se obtiene la dirección IP Local
 delay(2000);
 ShowSerialData();
 mySerial.println("AT+CIPSPRT=0");
 delay(3000);
 ShowSerialData();
 mySerial.println("AT+CIPSTART=\"t-
cp\",\"187.205.38.28\",\"80\"");                  //Pará-
metros hacia donde se va a enviar la cadena
 delay(2000);
 ShowSerialData();
 mySerial.println("AT+CIPSEND");
//Se envía la cadena
 delay(4000);
 ShowSerialData();

 //envío de cadena de texto
 String mensaje = "getstring.php?cadena=PRUEBA DE ENVÍO
DE MENSAJE POR GPRS";
 mySerial.println(mensaje);
 delay(500);
 ShowSerialData();
 mySerial.println((char)26);
 delay(5000);
 mySerial.println();
 ShowSerialData();
 mySerial.println("AT+CIPCLOSE");
//cierra la conexión
 delay(100);
 ShowSerialData();
}
```

La figura 11.12 muestra cómo se ve la aplicación en Freeboard.

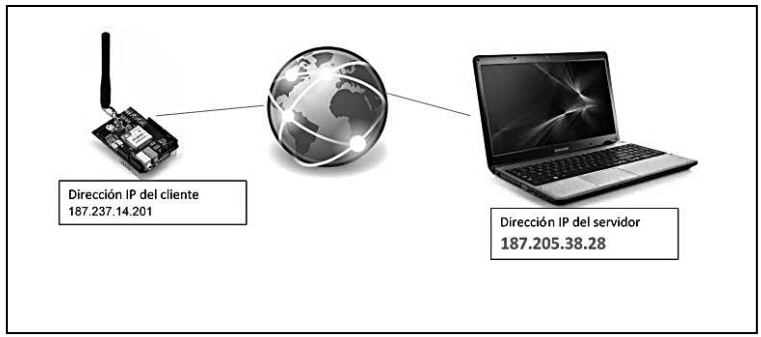


Figura 11.12 Aplicación de monitoreo remoto en Freeboard.

3) Establecer el comando para la conexión con el servidor. (Ver figura 11.13.)

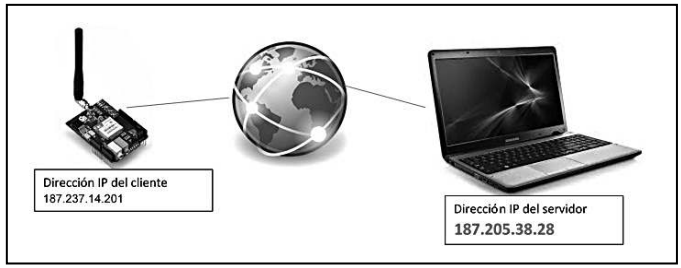


Figura 11.13 El equipo cliente se conecta a la red de Internet y envía el dato al servidor web.

```
mySerial.println("AT+CIPSTART=\"tcp\",\"187.205.38.28\",\"80\"");
```

El comando para que se envíe la cadena de texto es el siguiente:

```
mySerial.println("AT+CIPSEND");
```

El dato enviado es:

```
String mensaje = "getstring.php?cadena=PRUEBA DE ENVÍO DE
MENSAJE POR GPRS";

mySerial.println(mensaje);
```

4) Se crea el sitio remoto elaborando un panel de control (dashboard) en Freeboard. Hay que diseñar el panel remoto donde estarán mostrando las coordenadas del módulo GPS. Primeramente, se debe crear una fuente de datos (Data Source) con el nombre "gprs" en el campo Thing name (nombre del objeto) y de Tipo (type) Dweet.io, como se puede apreciar en la figura 11.14.

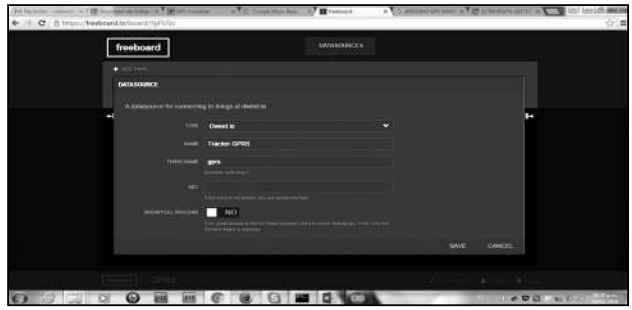


Figura 11.14 Crear el sitio remoto en Freeboard.

5) Crear un widget haciendo clic en Add pane. Se desplegará la pantalla que se muestra en la figura 11.15.

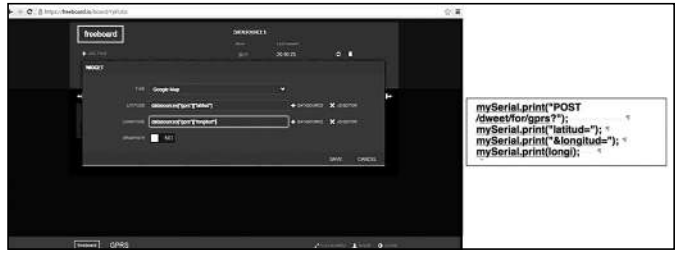


Figura 11.15 Crear un widget de las coordenadas.

Es importante considerar que los nombres de los parámetros están totalmente relacionados con los datos que se configuraron en el sketch de Arduino.

6) Una vez que se carga el sketch a la tarjeta, ejecutar el monitor serial, donde aparece el proceso del envío de los datos. El proceso de conexión (ver figura 11.16) es:

```
OK
AT+CSTT="internet.itelcel.com", "wapgprs", "wapgprs2003"

ERRORAT+CIICR

ERROR
AT+CIFSR

10.98.97.183
AT+CIPSPRT=0

OK
AT+CIPSTART="TCP", "dweet.io", "80"

OK

CONNECT OK
```

Figura 11.16 Proceso de conexión.

7) Entonces se enviarán los datos del módulo GPS (ver figura 11.17):

```
POST /dweet/for/gprs?latitud=21.11255&longitu-
de=-101.64414 HTTP/1.1
```

Figura 11.17 Envío de los datos del módulo GPS.

La figura 11.18 muestra cómo se ven los datos en la pantalla del Dashboard.

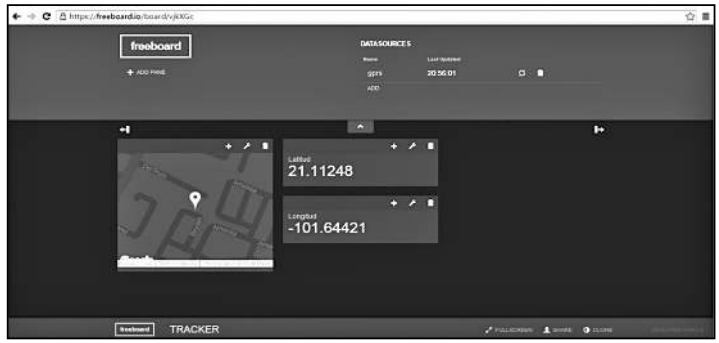


Figura 11.18 Dashboard de nuestro rastreador remoto.

11.8 Resumen

En este capítulo se describieron varios proyectos también orientados al Internet de las cosas. En esta ocasión, se utilizaron los módulos de comunicación que también se pueden añadir a la tarjeta Arduino, GPS y GPRS. Se explicó la puesta en marcha y configuración del módulo GPS para rastrear las coordenadas de ubicación, mostrarlas en una pantalla LCD y reenviarlas a un mapa de Google Maps, que se puede apreciar en un sitio web.

Asimismo, se desarrolló el proyecto utilizando otro módulo muy importante para la aplicación de redes de datos y comunicación: el Shield GSM/GPRS. En la primera parte del capítulo se utilizó el modo GSM para enviar las coordenadas a través de un mensaje de SMS. Posteriormente, las coordenadas se enviaron de forma remota al sitio web.

Para el envío por SMS resultó crucial aprender a integrar el Shield Ethernet; gracias a él se pueden enviar los datos al sitio del Internet de las cosas llamado Dweet.io; así se pudo monitorear remotamente en tiempo real. Esto está limitado a que el dispositivo se encuentre conectado por medio de un cable, lo cual puede no ser práctico, pero funcionó a modo de prueba. Finalmente, se desarrolló un proyecto donde se utilizó el modo GPRS para enviar las coordenadas y ubicar el dispositivo en el mapa. Cabe señalar que el módulo GPRS permite estar conectado a Internet en cualquier ubicación sin tener que conectar un cable a la red de datos. Esta forma es más práctica, pues el dispositivo puede permanecer en movimiento, ya que en cualquier lugar se conectará al sitio Dashboard remoto.

En la vida cotidiana, esta aplicación permitirá desarrollar proyectos de rastreo de dispositivos que se encuentren, por ejemplo, en taxis, camiones de carga, ambulancias, etc. en tiempo real y con ello saber en qué lugar se encuentran.

En el siguiente capítulo se desarrollará un proyecto con un robot móvil, el cual se podrá rastrear y controlar inalámbricamente y, para aumentar la complejidad, el control se hará por medio de reconocimiento de voz.

11.9 Problemas

1. Cree una aplicación de tipo policía espía, que cuando detecte un movimiento guarde la voz mediante un sensor de sonido y envíe una respuesta a una página web.
2. Implemente una aplicación con un sensor de movimiento PIR, que cuando detecte el movimiento envíe un mensaje de texto SMS.
3. Realice lo mismo que el proyecto anterior pero ahora con una llamada telefónica.
4. Desarrolle un proyecto que permita el acceso a una casa con el sensor de huella digital; si es correcta la huella debe enviar el mensaje de texto "Acceso concedido", si no es correcto debe enviar un mensaje de texto para activar la chapa.
5. Cree una aplicación web de tipo ASP.NET que envíe los datos de temperatura y humedad por medio de GSM/GPRS.
6. Guarde los datos leídos en una base de datos en red del proyecto anterior.
7. Active dos cargas de potencia (relevador y foco) desde una página ASP.NET, hágalo a través de GSM/GPRS.
8. Implemente una aplicación en dweet.io y Freeboard (Dashboard) que permita monitorear la temperatura, humedad, luz y detección de sensor de movimiento.
9. Cree alertas mediante envío de correo electrónico del ejemplo anterior.
10. Implemente el proyecto del rastreador en una bicicleta móvil, agregue un sensor de movimiento a través de Freeboard.
11. Desarrolle el proyecto del rastreador en una patineta; agregue una pantalla LCD a través de Freeboard.
12. Cree el proyecto del rastreador en una carriola de niño; agregue sensores de temperatura y humedad a través de Freeboard.
13. Implemente el proyecto del rastreador a través de un reloj de mano; agregue un sensor de pulso cardiaco a través de Freeboard.
14. Realice una aplicación de monitoreo con GPS para rastreador de un automóvil; validar la latitud y longitud con base en los movimientos, determinar un rango determinado y, si el dispositivo se sale de ese rango, enviar un mensaje de texto por SMS; esto se puede emplear para colocar este dispositivo en un automóvil con el objetivo de determinar si el auto fue robado.
15. Lleve a cabo un proyecto de monitoreo por medio de GSM/GPRS y GPS que permita rastrear la trayectoria del móvil, de tal manera que el dispositivo avise cuando el automóvil se encuentra en una zona prohibida; si detecta la zona, debe enviar un mensaje de texto SMS.

Capítulo 12

Robot controlado inalámbricamente

Objetivos

En este capítulo se explicará cómo configurar y programar la tarjeta Arduino Yun para controlar a un robot inalámbricamente. Esto se podrá hacer estableciendo una comunicación inalámbrica a la que se tenga acceso vía Wi-Fi y que se pueda controlar mediante la voz.

12.1 Introducción

En este último capítulo del libro se explicará cómo utilizar la tarjeta Arduino Yun en un campo completamente diferente: la robótica. De tal forma, se aprenderá a conectar motores de corriente continua, así como a construir un robot móvil propio cuyo cerebro sea la Arduino Yun y que tenga un sensor de distancia para que se le controle inalámbricamente mediante Wi-Fi usando una simple interfaz web. También se podrá obtener una visualización en directo de las mediciones realizadas por el robot; por ejemplo, la distancia que el sensor ultrasónico mide en frente del robot.

12.2 Construcción del robot móvil

Para construir un robot móvil que tiene el Arduino Yun como su "cerebro" y para controlarlo por completo a través de la red Wi-Fi desde una computadora o un dispositivo móvil (como un teléfono inteligente o una tableta), primeramente debe programarse un sketch Arduino para el robot que recibirá órdenes y enviará datos de regreso, así como programar una interfaz gráfica en la computadora.

De esta manera, si se quisiera construir más aplicaciones complejas en el futuro, sólo se tendría que cambiar el software que se ejecuta en la computadora y dejar el robot intacto.

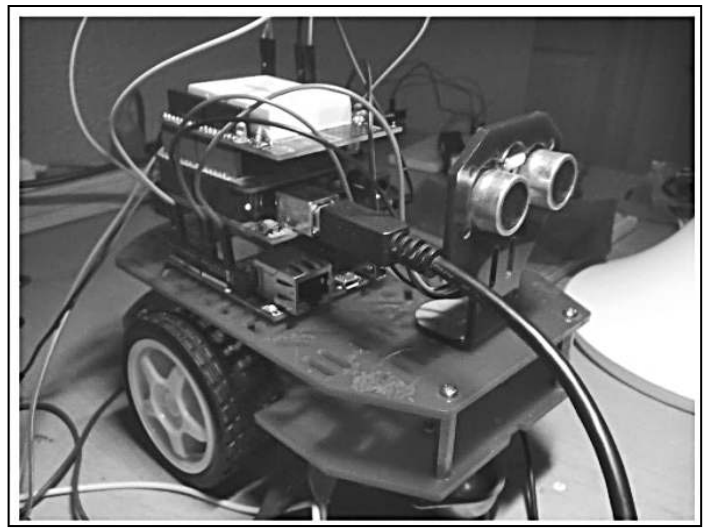


Figura 12.1 Robot móvil.

La arquitectura del proyecto está conformada por una tarjeta Arduino YUN, cuya función principal será comunicarse con la tarjeta Arduino UNO, con el fin de enviar y recibir los comandos de control de los motores y de los datos de la distancia que mide

el sensor ultrasónico. La Arduino UNO recibirá las instrucciones y las enviará al shield de motores. A su vez, los motores del carrito se conectan al shield de motores y el shield del protoboard permitirá hacer conexiones especiales para alimentar los módulos y suministrar el voltaje de 5 volts y GND.

La alimentación del sistema la recibe la tarjeta Arduino UNO con una pila de 9 volts conectada por medio del Jack. Para fines de prueba, se conectó el cable USB de la computadora para alimentación de la tarjeta Arduino UNO.

12.3 Requerimientos de software y hardware

Para el ensamblado del robot móvil se ocuparon diferentes partes mecánicas y eléctricas que a continuación se describen (Ver figuras 12.2 a 12.5):

- Base para carrito
- Soportes
- Dos llantas para carrito
- Una rueda loca
- Dos motoreductores
- Batería de 9 volts
- Arduino UNO
- Arduino YUN
- Shield de motores
- Shield para protoboard
- Un sensor ultrasónico
- Cables macho-macho

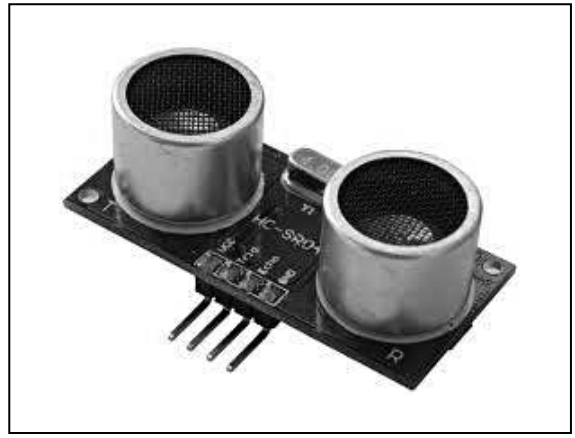


Figura 12.2 Sensor ultrasónico.

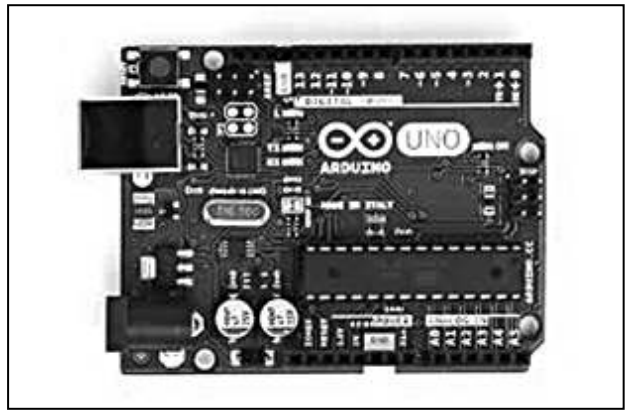


Figura 12.3 Tarjeta Arduino UNO.

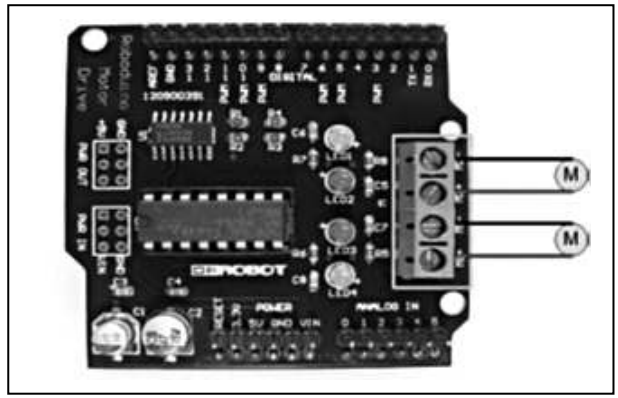


Figura 12.4 Shield de motores.

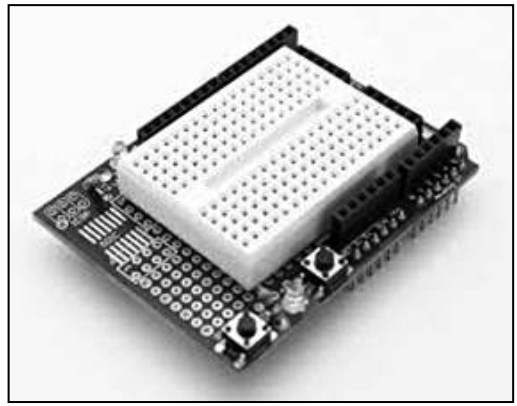


Figura 12.5 Shield con protoboard.

12.4 Configuración del hardware

Se deben ensamblar todos los elementos paso a paso, cuidadosamente.

La figura 12.6 muestra cómo ensamblar la tarjeta Arduino YUN a la base del carrito.

Figura 12.6 Tarjeta YUN montada en el carrito que será el cuerpo del robot.

Después, hay que ensamblar la tarjeta Arduino UNO a la placa de Arduino YUN, como lo muestra la figura 12.7:

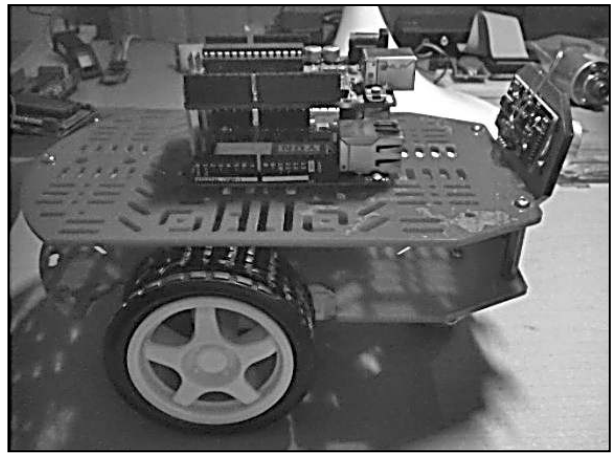


Figura 12.7 Montando la tarjeta Arduino UNO a la tarjeta Arduino YUN.

Luego, se debe ensamblar el shield de motores a las placas Arduino, como se aprecia en la figura 12.8:

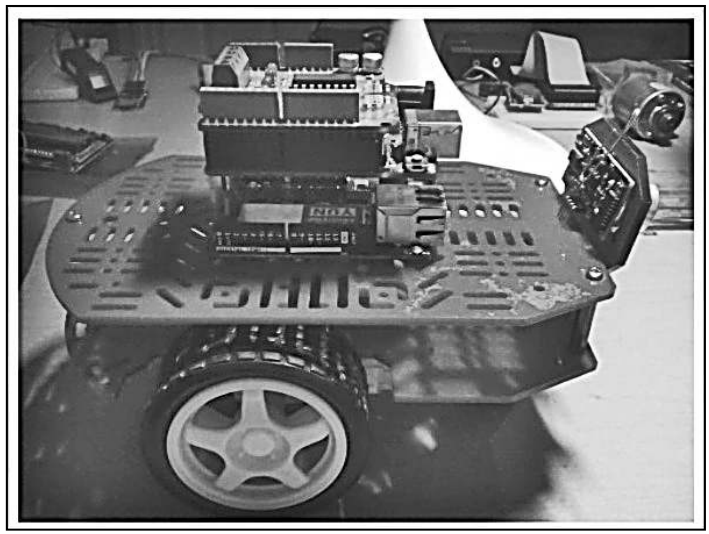


Figura 12.8 Ensamble del shield de motores a las placas.

Después, hay que conectar los motores al módulo de su shield, como lo muestra la figura 12.9:

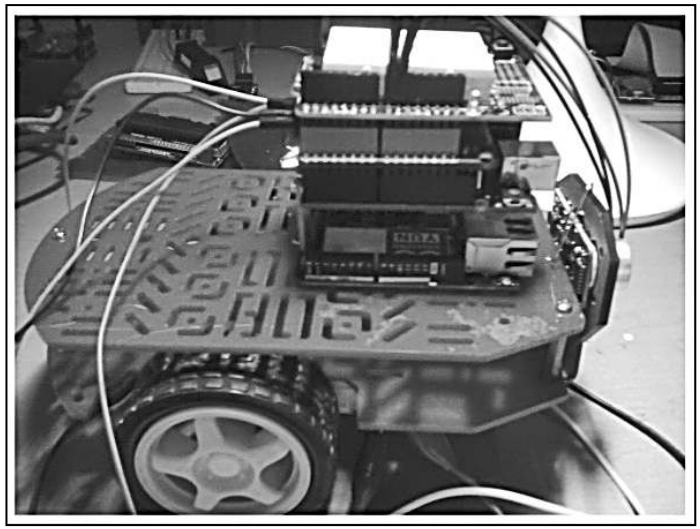


Figura 12.9 Conexión de los motores al módulo de shield de motores.

De tal forma, la figura 12.10 muestra todos los elementos conectados.

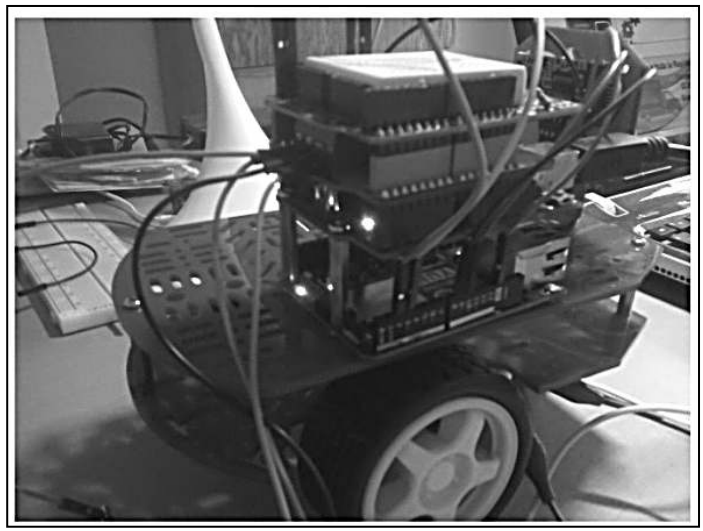


Figura 12.10 Las tarjetas, el carrito, el shield de motores y los motores conectados.

En la figura 12.11 se presenta el diagrama de conexión entre ambas tarjetas a través de la comunicación I2C.

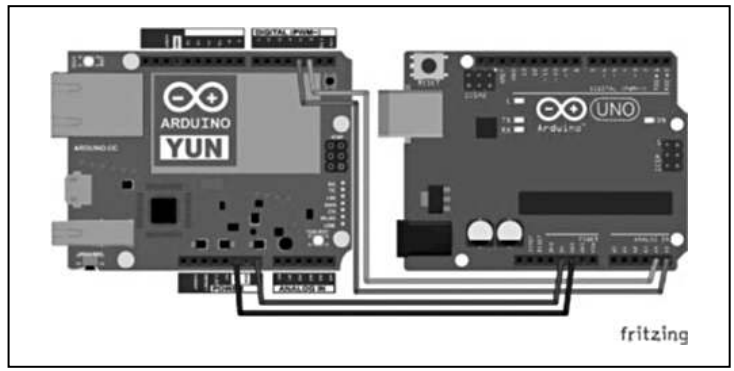


Figura 12.11 Diagrama de conexión entre YUN y UNO.

En la figura 12.12 se puede apreciar el cableado.

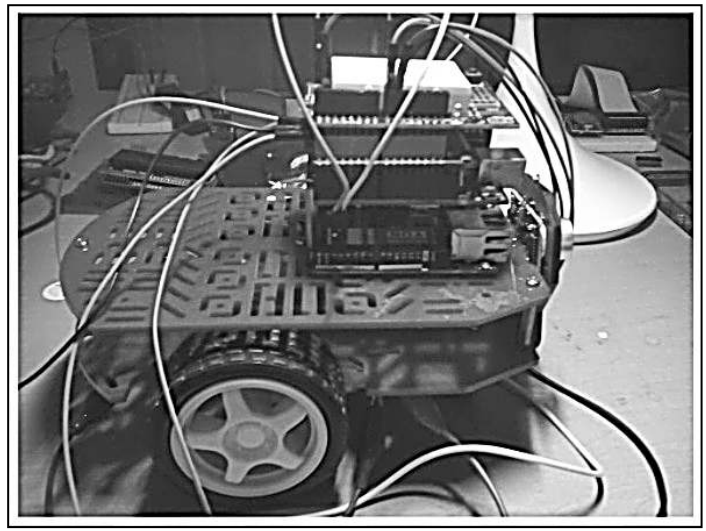


Figura 12.12 Cableado de las conexiones.

Ahora es posible conectar el sensor ultrasónico. Para ello, conectar el pin Trigger al pin número 10 de la placa Arduino YUN, el pin Echo se conecta al pin 10 de la misma tarjeta, VCC a voltaje de 5 volts y tierra GND. (Ver la figura 12.13.)

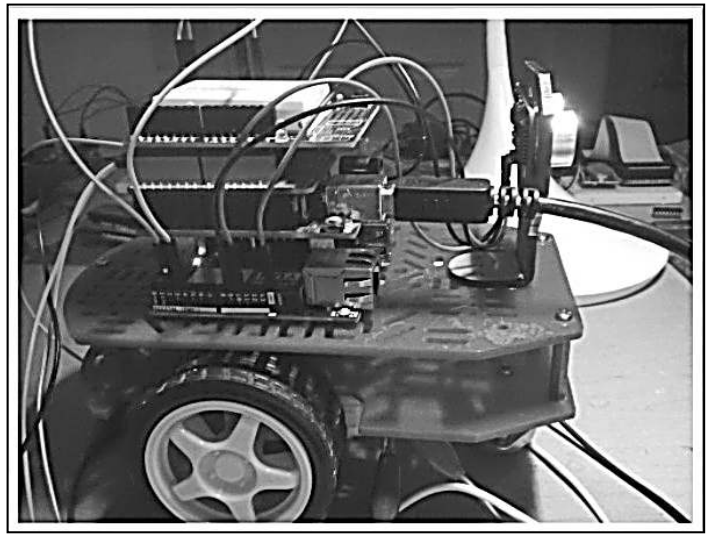


Figura 12.13 Conexión del sensor ultrasónico.

En la figura 12.14 se puede apreciar cómo se ve el móvil terminado y armado.

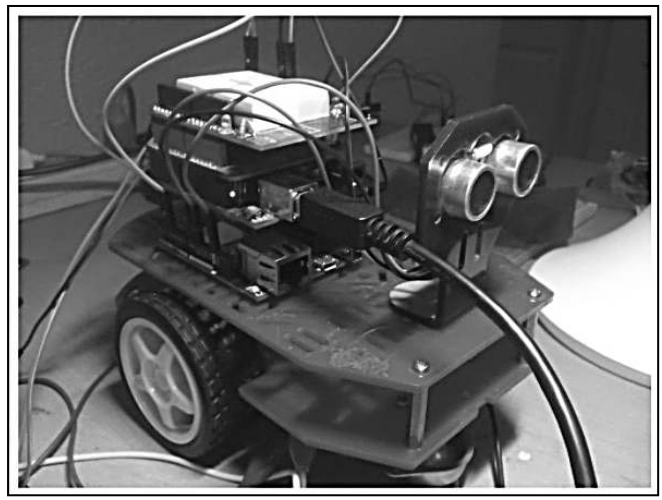


Figura 12.14 El robot móvil completamente armado y conectado.

12.5 Comunicación inalámbrica

La comunicación se realiza de forma inalámbrica conectando la tarjeta Arduino YUN a la red inalámbrica (Wi-Fi) y configurando la tarjeta como punto de acceso. Hay que seguir estos tres pasos:

a) Desde la computadora, conectarse a la red como punto de acceso de nuestro Arduino YUN, como se ve en la figura 12.15.

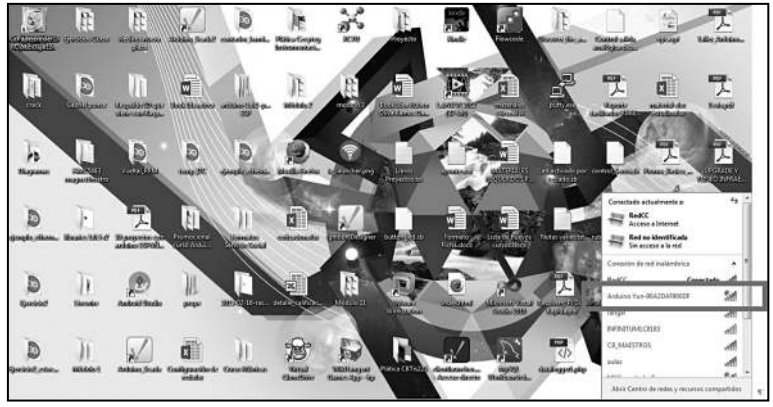


Figura 12.15 Configurar la red Wi-Fi.

b) Acceder al panel de la tarjeta Arduino tecleando: http:Arduino.local, como se ve en la figura 12.16.

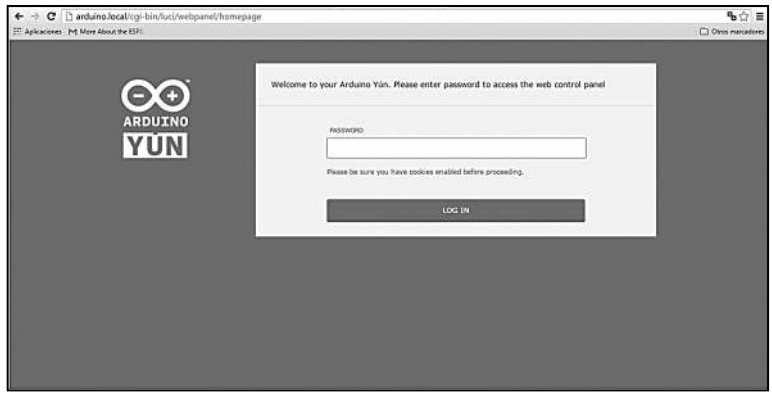


Figura 12.16 Entrar al panel de Arduino en http:Arduino.local.

c) Una vez que se tiene acceso, la plataforma nos muestra la configuración de la tarjeta. (Ver figura 12.17.)

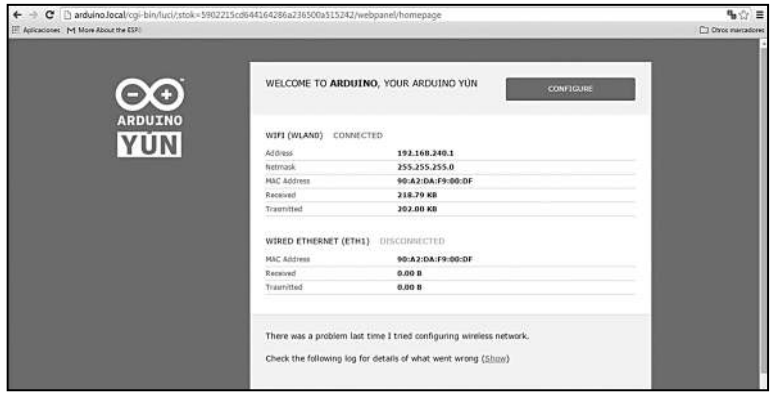


Figura 12.17 Configurar la tarjeta Arduino YUN.

12.6 Programación de los módulos

En esta sección se presentarán los códigos para cada módulo.

12.6.1 Código para la tarjeta Arduino UNO. Módulo 1.

```
#include <Wire.h>

int speed_motor1 = 6;
```

```
int speed_motor2 = 5;
int direction_motor1 = 7;
int direction_motor2 = 4;

void setup()
{
  // Set pin 4,5,6,7 to output mode
  for(int i=4;i<=7;i++)
  {
    pinMode(i, OUTPUT);
  }

  Wire.begin(4);
  Wire.onReceive(receiveEvent);

  Serial.begin(9600);
}



void loop()
{
  delay(100);
}

// Función que se ejecuta cuando el Arduino YUN envía da-
tos
void receiveEvent(int howMany)
{
  // Lectura de datos enviados

  int pwm1 = Wire.read();
  Serial.print(pwm1);
  char c = Wire.read();
  Serial.print(c);

  int dir1 = Wire.read();
  Serial.print(dir1);
  c = Wire.read();
  Serial.print(c);

  int pwm2 = Wire.read();
  Serial.print(pwm2);
  c = Wire.read();
  Serial.print(c);

  int dir2 = Wire.read();
  Serial.println(dir2);

  // Aplicación de los comandos enviados
  send_motor_command(speed_motor1,direction_motor1,pwm1,-
dir1);
  send_motor_command(speed_motor2,direction_motor2,pwm2,-
dir2);
```

```
}

// Función para los comandos
void send_motor_command(int speed_pin, int direction_pin,
int pwm, boolean dir)
{
  analogWrite(speed_pin,pwm);
  digitalWrite(direction_pin,dir);
}
```

12.6.2 Código para la tarjeta Arduino UNO. Módulo 2.

```
#include <Wire.h>
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>

int trigPin = 10;   // Se elige el pin para Trig
int echoPin = 9;    // Se elige el pin para Echo
long duration, cm;  // Variables que se utilizarán

YunServer server;

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Wire.begin();

  Bridge.begin();

  server.listenOnLocalhost();
  server.begin();
}

void loop()
{
   // El emisor se activa al recibir un impulso HIGH de
al menos 10 milisegundos.
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // La duración es el tiempo en microsegundos que tarda
la señal

  duration = pulseIn(echoPin, HIGH);
```

```
  // Convertimos el tiempo en distancia
  cm = (duration / 58);

YunClient client = server.accept();

  if (client) {
    // Process request
    process(client);

   client.stop();
  }

  delay(50);

}

// Inicia proceso de las funciones con RESP API
void process(YunClient client) {

  // lectura de comando
  String command = client.readStringUntil('/');

  if (command == "robot") {
     robotCommand(client);
  }

}

// comandos enviados
void robotCommand(YunClient client) {

  // inicia la lectura de comandos
  String command = client.readStringUntil('\r');

  if (command == "detener") {
    Wire.beginTransmission(4);
    Wire.write(0);
    Wire.write(",");
    Wire.write(0);
    Wire.write(",");
    Wire.write(0);
    Wire.write(",");
    Wire.write(0);
   Wire.endTransmission();
  }



   if (command == "vizquierda") {
    Wire.beginTransmission(4);
    Wire.write(75);
    Wire.write(",");
    Wire.write(0);
```

```
      Wire.write(",");
      Wire.write(75);
      Wire.write(",");
      Wire.write(1);
      Wire.endTransmission();
    }

    if (command == "vderecha") {
      Wire.beginTransmission(4);
      Wire.write(75);
      Wire.write(",");
      Wire.write(1);
      Wire.write(",");
      Wire.write(75);
      Wire.write(",");
      Wire.write(0);
      Wire.endTransmission();
    }

    if (command == "atrás") {
      Wire.beginTransmission(4);
      Wire.write(255);
      Wire.write(",");
      Wire.write(1);
      Wire.write(",");
      Wire.write(255);
      Wire.write(",");
      Wire.write(1);
      Wire.endTransmission();
    }

    if (command == "adelante") {
      Wire.beginTransmission(4);
      Wire.write(255);
      Wire.write(",");
      Wire.write(0);
      Wire.write(",");
      Wire.write(255);
      Wire.write(",");
      Wire.write(0);
      Wire.endTransmission();
    }

     // solicita la distancia con el sensor ultrasónico
     if (command == "distancia") {
       client.print("La distancia enfrente del robot es:
");
       client.print(cm);
       client.println(" cm.");
    }
  }
```

12.7 Prueba de los comandos desde el navegador web

Es necesario correr una prueba de los comandos desde Internet. En esta sección se mostrarán las imágenes de los comandos motor adelante, motor atrás, detener motor y medir distancia.

La figura 12.18 muestra el comando motor adelante: 192.168.240.1/arduino/robot/adelante

Figura 12.18 Comando motor adelante.

La figura 12.19 muestra el comando motor atrás: 192.168.240.1/arduino/robot/atras.

Figura 12.19 Comando motor atrás.

De igual forma, la figura 12.20 muestra el comando detener motor: 192.168.241.1/arduino/robot/detener

Figura 12.20 Comando detener motor.

Por último, la figura 12.21 muestra el comando medir distancia: 192.168.204.1/arduino/robot/distancia

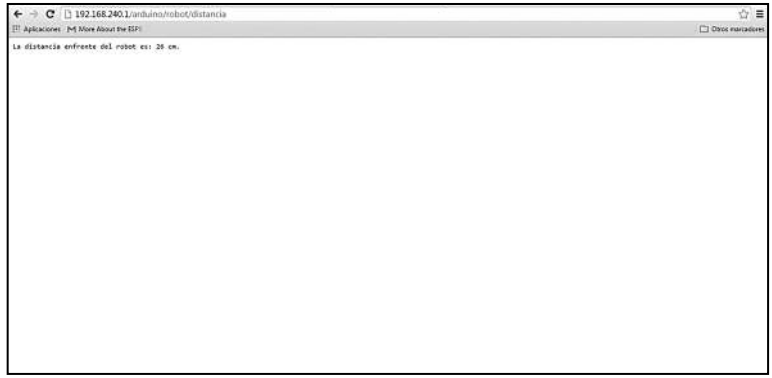


Figura 12.21 Comando medir distancia.

12.8 Interfaz hombre-máquina

En esta etapa del proyecto se debe realizar una página web con las herramientas utilizadas en capítulos anteriores con ASP, gracias a la cual se envían los comandos a través de los botones.

La figura 12.22 nos ayuda a entender cómo funciona esta herramienta. Nótese que hay botones para mover el robot hacia adelante, hacia atrás, para dar vuelta a la derecha, a la izquierda, para medir la distancia y para detenerlo.

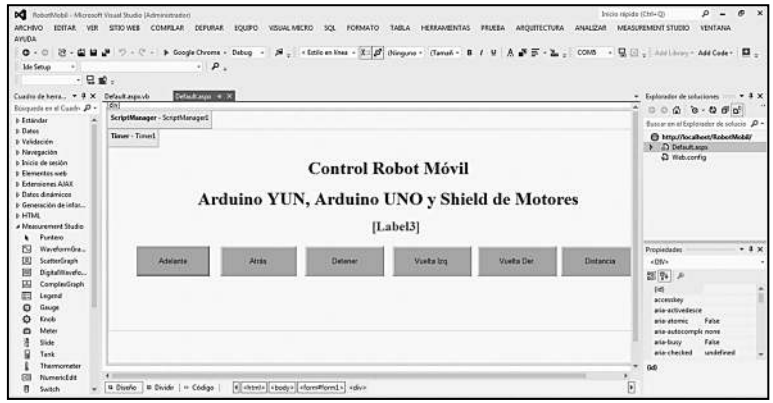


Figura 12.22 Crear página web con la herramienta ASP.

12.8.1 Código de la aplicación de los botones en la página web

A continuación se presenta el código de la aplicación para todos los botones:

```
Imports System.Net
Imports System.IO

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button2_Click(sender As Object, e As
EventArgs) Handles Button2.Click
        Dim myWebClient As New WebClient()
        Dim postStream1 As Stream = myWebClient.OpenWri-
te("http://192.168.240.1/arduino/robot/adelante")
        postStream1.Close()
    End Sub

    Protected Sub Button3_Click(sender As Object, e As
EventArgs) Handles Button3.Click
        Dim myWebClient As New WebClient()
        Dim postStream2 As Stream = myWebClient.OpenWri-
te("http://192.168.240.1/arduino/robot/atras")
        postStream2.Close()
    End Sub



Protected Sub Button4_Click(sender As Object, e As Even-
tArgs) Handles Button4.Click
        Dim myWebClient As New WebClient()
        Dim postStream3 As Stream = myWebClient.OpenWri-
te("http://192.168.240.1/arduino/robot/detener")
        postStream3.Close()
    End Sub
```

```
    Protected Sub Button6_Click(sender As Object, e As
EventArgs) Handles Button6.Click
        Dim myWebClient As New WebClient()
        Dim postStream4 As Stream = myWebClient.OpenWri-
te("http://192.168.240.1/arduino/robot/vizquierda")
        postStream4.Close()
    End Sub

    Protected Sub Button5_Click(sender As Object, e As
EventArgs) Handles Button5.Click
        Dim myWebClient As New WebClient()
        Dim postStream5 As Stream = myWebClient.OpenWri-
te("http://192.168.240.1/arduino/robot/vderecha")
        postStream5.Close()
    End Sub

    Protected Sub Button7_Click(sender As Object, e As
EventArgs) Handles Button7.Click
        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.240.1/arduino/robot/distancia")
        Label3.Text = lectura
    End Sub
End Class
```

12.8.2. Actualización de datos desde el Page Load

A continuación se presenta el código para que los datos se actualicen desde el Page Load, pues la función que lee la distancia del sensor debe estarse actualizando siempre, ya que es la que detecta la distancia del móvil, es la página principal y es el punto donde la página web se va actualizar va a cargar todas las funciones iniciales.

```
Protected Sub Page_Load(sender As Object, e As EventArgs)
Handles Me.Load

        Dim myWebClient As New WebClient()
        Dim lectura As String = myWebClient.DownloadS-
tring("http://192.168.240.1/arduino/robot/distancia")
        Label3.Text = lectura
End Sub
```

12.8.3 Autorefresh con Ajax Script Manager y Timer

A continuación se presenta el código para programar el autorefresh con Ajax Script Manager y con Timer. Es una función de actualización que permite que se refresquen solamente los elementos que se indican; el Timer es el que determina cada cuánto tiempo se va a realizar la actualización; cuando esto sucede se refresca el dato en pantalla y la distancia siempre se va a mostrar en tiempo real.

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFi-
le="Default.aspx.vb" Inherits="_Default" %>
<%@ Import Namespace="System.Net" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; char-
set=utf-8"/>

    <title>Control Robot Móvil</title>
</head>
<body>
    <form id="form1" runat="server">
    <div style="height: 454px; width: 1006px; text-align:
center;">
        <asp:ScriptManager ID="ScriptManager1" runat="-
server"></asp:ScriptManager>

        <asp:UpdatePanel ID="UpdatePanel1" runat="ser-
ver">
            <ContentTemplate>

                <asp:Timer ID="Timer1" runat="server" In-
terval="1000"></asp:Timer>

                <br />
                <asp:Label ID="Label1" runat="server"
Font-Bold="True" style="font-size: xx-large" Text="Con-
trol Robot Móvil"></asp:Label>
                <br />
                <br />
                <asp:Label ID="Label2" runat="server"
Font-Bold="True" style="font-size: xx-large; text-align:
center;" Text="Arduino YUN, Arduino UNO y Shield de Moto-
res"></asp:Label>
                <br />
                <br />
                      &nbs-
p;<asp:Label ID="Label3" runat="server" ForeColor="#-
3333FF" style="font-size: x-large; font-weight: 700"></
asp:Label>
                <br />
                <br />

 <%
 Dim myWebClient As New WebClient()
 Dim lectura As String =    myWebClient.DownloadStrin-
g("http://192.168.240.1/arduino/robot/distancia")
Label3.Text = lectura
 %>

                <asp:Button ID="Button2" runat="server"
Height="53px" Text="Adelante" Width="130px" BackColor="#-
```

```
FF9900" BorderColor="#FF9900" style="margin-right: 0px;
margin-top: 4px" />
                    
                <asp:Button ID="Button3" runat="server"
Height="53px" Text="Atrás" Width="136px" BackColor="#-
FF9900" style="margin-top: 0px" />
                   
                <asp:Button ID="Button4" runat="server"
Height="53px" Text="Detener" Width="141px" BackColor="#-
FF9900" />
                   
                <asp:Button ID="Button6" runat="server"
BackColor="#FF9900" Height="53px" Text="Vuelta Izq." Wid-
th="141px" />
                  
                <asp:Button ID="Button5" runat="server"
BackColor="#FF9900" Height="53px" Text="Vuelta Der." Wid-
th="141px" />
                 
                <asp:Button ID="Button7" runat="server"
BackColor="#FF9900" Height="53px" Text="Distancia" Wid-
th="141px" />
                <br />
                <br />

                <br />
                <br />

                <br />
                <br />
            </ContentTemplate>
         </asp:UpdatePanel>

  
        <br />
        <br />

        <br />

        <br />

   </div>
    </form>
</body>
</html>
```

12.8.4 Envío de comandos desde el sitio web

A través de las siguientes imágenes se describirá cómo se envían los comandos al robot inalámbricamente desde el sitio web. La figura 12.23 muestra la pantalla de inicio del sitio.

Figura 12.23 Pantalla de inicio del sitio web del robot.

Asimismo, la figura 12.24 muestra la pantalla de inicio del sitio web con una actualización automática.

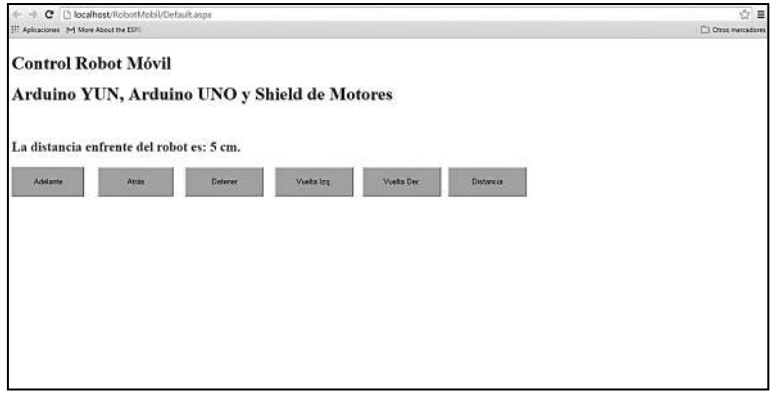


Figura 12.24 Actualización automática del sitio web.

Así, desde cualquier navegador se puede teclear el sitio web para acceder a él y poder controlar el robot móvil de forma inalámbrica.

12.9 Control mediante la voz

En el punto anterior se controló el robot móvil de forma inalámbrica desde una página web, utilizando la comunicación Wi-Fi, a través de la tarjeta Arduino YUN. Ahora, en este apartado se desarrollará un proyecto para controlar al robot móvil a través de la voz, para lo cual se activará un módulo de reconocimiento de voz. (Ver figura 12.25.)

Este módulo de reconocimiento de voz funciona según las siguientes acciones:

- Desde la placa Arduino se envían los comandos de configuración hacia el módulo.
- Se graban los mensajes para el control del dispositivo.
- Los mensajes quedan guardados en el módulo.
- Desde la tarjeta Arduino se leen los mensajes previamente guardados.
- Se crean las condiciones para efectuar el control requerido.

12.9.1 Configuración del módulo de reconocimiento de voz

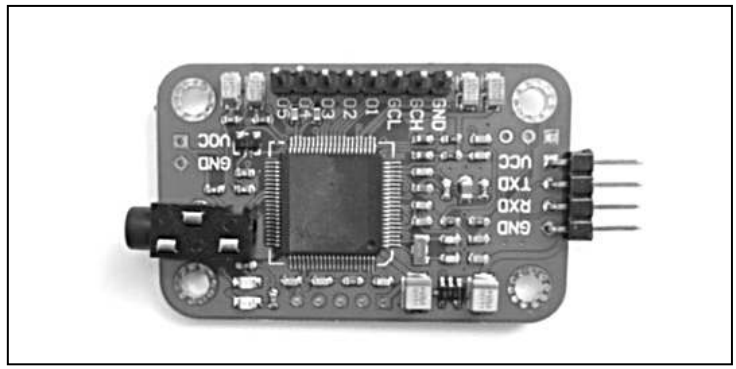


Figura 12.25 Módulo de reconocimiento de voz.

En esta sección se explicará cómo configurar el módulo que reconoce las instrucciones de voz para mover al robot.

Cabe aclarar que el módulo puede grabar hasta 15 comandos de instrucciones divididos en tres grupos; en cada grupo se permiten guardar 5 comandos. Se deben seguir los comandos en formato hexadecimal que se presentan en la siguiente tabla.

Tabla de comandos y códigos de configuración del módulo

Comando en formato Hexadecimal	Descripción del comando
0x11	Comienza el guardado de los mensajes en el grupo 1.
0x12	Comienza el guardado de los mensajes en el grupo 2.
0x13	Comienza el guardado de los mensajes en el grupo 3.

Comando en formato Hexadecimal	Descripción del comando
0x21	Importa grupo 1 y está listo para las instrucciones de voz.
0x22	Importa grupo 2 y está listo para las instrucciones de voz.
0x23	Importa grupo 3 y está listo para las instrucciones de voz.
0x36	Cambia a modo común.
0x37	Cambia a modo compacto.

12.9.2 Grabar comandos de texto

Para establecer los comandos de voz, se debe seguir un procedimiento específico. Desde la placa Arduino, enviar los comandos al módulo de reconocimiento de voz: a través de la comunicación serial, el pin TX del módulo se conecta al pin RX del Arduino; el pin RX del módulo de conecta al pin TX del Arduino. Los pines de alimentación del módulo deben alimentarse desde el mismo programa; el pin VCC del módulo se conecta al pin 8 del Arduino y el pin GND del módulo se conecta el pin 9 del Arduino.

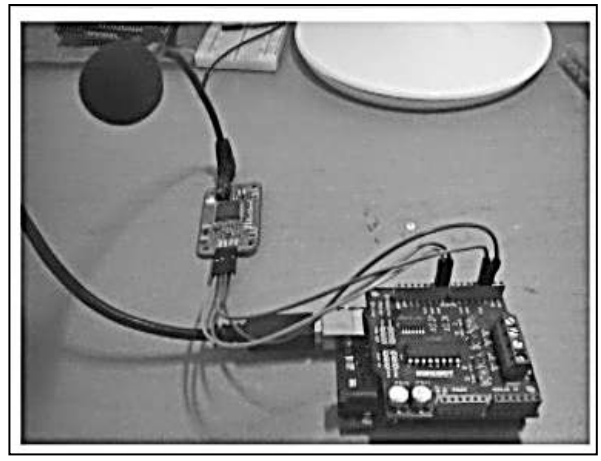


Figura 12.26 Conexión del módulo de reconocimiento de voz.

A continuación se configura el shield de motores. (Ver figura 12.27.)

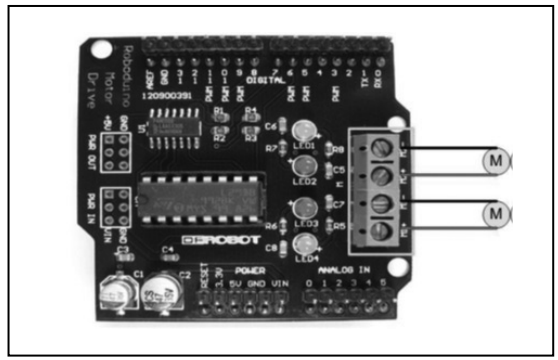


Figura 12.27 Shield para motores de DFRobot.

Los pines de configuración son los siguientes:

Pin	Función
Digital 4	Control de dirección motor 2
Digital 5	Control PWM motor 2
Digital 6	Control PWM motor 1
Digital 7	Control de dirección motor 2

La figura 12.28 muestra cómo se ven estos elementos ya conectados.

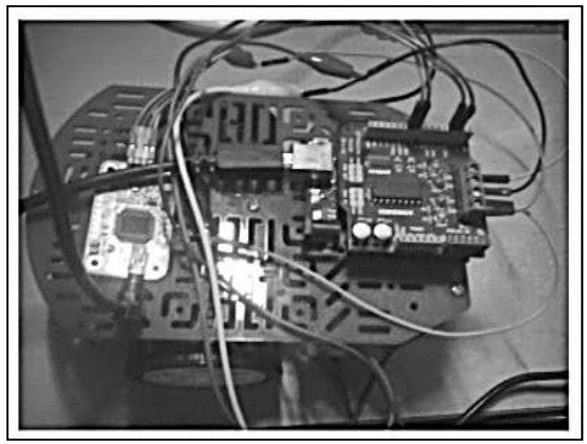


Figura 12.28 Shield de motores configurado.

Los comandos para guardar los mensajes se envían a través de comandos de forma serial desde la placa Arduino de la siguiente manera:

```
//Configuramos el módulo de reconocimiento de comandos de
voz
  pinMode(vozVcc,OUTPUT);
  pinMode(vozGnd,OUTPUT);
  digitalWrite(vozVcc,HIGH);
  digitalWrite(vozGnd,LOW);
  vozSerial.begin(9600);
  delay(200);

  //Cambiamos al modo compacto del módulo de voz
  vozSerial.write(0xAA);
  vozSerial.write(0x37);
  delay(200);

  //Importamos el grupo 1 y estamos listos para los co-
mandos de voz
  vozSerial.write(0xAA);
  vozSerial.write(0x21);
```

El comando para iniciar el guardado de los comandos en el grupo 1 es:

```
case 'G':
          vozSerial.write(0xAA);
          vozSerial.write(0x11);
          break;
```

El comando para importar los comandos del grupo 1 es:

```
case 'R':
          //Importamos el grupo 1 y estamos listos para
los comandos de voz
          vozSerial.write(0xAA);
          vozSerial.write(0x21);
          break;
```

Estos comandos se envían desde el monitor serial para cargarle las instrucciones. Posteriormente, los códigos de respuesta durante el guardado de los comandos son:

```
case 0xE0:
        Serial.println("ERROR!");
        break;

      case 0x40:
        Serial.println("START");
        break;

      case 0x41:
        Serial.println("No voice");
```

```
        break;

      case 0x42:
        Serial.println("Again");
        break;

      case 0x43:
        Serial.println("Too loud");
        break;

      case 0x44:
        Serial.println("Different");
        break;

      case 0x45:
        Serial.println("Finish One");
        break;

      case 0x46:
        Serial.println("Group Finished");
        break;
```

Cuando se inicializa el guardado de los comandos aparece la leyenda START. Si la voz es correcta, aparecerá la leyenda AGAIN y si lo guardado es correcto, aparecerá el mensaje Finish One. Esto significa que el primer mensaje se ha guardado correctamente. Se tiene que seguir el mismo procedimiento con las cinco instrucciones; al finalizar el quinto mensaje, aparecerá la leyenda Group Finish. Entonces, se está listo para empezar a mover al robot.

12.9.3 Movimiento con base en los mensajes grabados

Es importante saber que después de que los mensajes se guardaron en el módulo de reconocimiento de voz, de acuerdo a los códigos en formato hexadecimal comentados, se comparan en el programa para darles una funcionalidad.

Si se cumple la condición del mensaje recibido, el programa actúa con las instrucciones establecidas:

```
      case 0x11:                                          //
adelante
        digitalWrite(EN1,255);
        digitalWrite(IN1,HIGH);
        digitalWrite(EN2,255);
        digitalWrite(IN2,HIGH);
        Serial.println("Motor adelante");
        break;


      case 0x12:                                        //
atrás
        digitalWrite(EN1,255);
        digitalWrite(IN1,LOW);
```

```
        digitalWrite(EN2,255);
        digitalWrite(IN2,LOW);
        Serial.println("Motor atras");
        break;

      case 0x13:                                    //detener
        digitalWrite(EN1,0);
        digitalWrite(EN2,0);
        Serial.println("Motor detenido");
        break;
```

A continuación se presenta el código del ejemplo completo:

```
#include <SoftwareSerial.h>
SoftwareSerial vozSerial(2,3);                  //RX,TX

int EN1 = 6;  //control PWM motor 1
int EN2 = 5;  //control PWM motor 2
int IN1 = 7;  //dirección motor 1
int IN2 = 4; //dirección motor 2

#define vozVcc 8
#define vozGnd 9

void setup(){
  pinMode(EN1,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(EN2,OUTPUT);
  pinMode(IN2,OUTPUT);

  Serial.begin(9600);

  //Configuramos el módulo de reconocimiento de comandos
de voz
  pinMode(vozVcc,OUTPUT);
  pinMode(vozGnd,OUTPUT);

  digitalWrite(vozVcc,HIGH);
  digitalWrite(vozGnd,LOW);

  vozSerial.begin(9600);
  delay(200);

  //Cambiamos al modo compacto del módulo de voz
  vozSerial.write(0xAA);
  vozSerial.write(0x37);

  delay(200);

  //Importamos el grupo 1 y estamos listos para los co-
mandos de voz
  vozSerial.write(0xAA);
```

```
  vozSerial.write(0x21);
}
 void loop() {
  byte com = 0;

 if(vozSerial.available())
  {
    com = vozSerial.read();
    switch(com)
    {
      case 0x11:      //adelante
        digitalWrite(EN1,255);
        digitalWrite(IN1,HIGH);
        digitalWrite(EN2,255);
        digitalWrite(IN2,HIGH);
        Serial.println("Motor adelante");
        break;

      case 0x12:     //atrás
        digitalWrite(EN1,255);
        digitalWrite(IN1,LOW);
        digitalWrite(EN2,255);
        digitalWrite(IN2,LOW);
        Serial.println("Motor atras");
        break;

      case 0x13:      //detener
        digitalWrite(EN1,0);
        digitalWrite(EN2,0);
        Serial.println("Motor detenido");
        break;

      case 0xE0:
        Serial.println("ERROR!");
        break;

      case 0x40:
        Serial.println("START");
        break;

      case 0x41:
        Serial.println("No voice");
        break;

      case 0x42:
        Serial.println("Again");
        break;

      case 0x43:
        Serial.println("Too loud");
        break;

      case 0x44:
```

```
          Serial.println("Different");
          break;

        case 0x45:
          Serial.println("Finish One");
          break;

        case 0x46:
          Serial.println("Group Finished");
          break;

         //default:
         // Serial.println("invalid entry");
         // break;
      }
    }
    if (Serial.available())
    {
      char input = Serial.read();
      if(input != -1)
      {
        switch(input)
        {
          case 'G':
            vozSerial.write(0xAA);
            vozSerial.write(0x11);
            break;

          case 'R':
            //Importamos el grupo 1 y estamos listos para
los comandos de voz
            vozSerial.write(0xAA);
            vozSerial.write(0x21);
            break;
         }
      }
    }
}
```

12.10 Integración de ambas tecnologías

Si en este proyecto se pretendiera controlar el carrito móvil tanto de forma inalámbrica como por voz, se podría implementar una solución para que ambas partes interactúen: agregar otro módulo de comunicación bluetooth para enviar los comandos de forma inalámbrica a través de la implementación de otras tecnologías de comunicación inalámbrica.

12.11 Resumen

En este último capítulo se presentaron cuestiones interesantes para implementar las diversas tecnologías de Arduino y se ha estudiado a profundidad cómo combinar diferentes tarjetas para realizar diferentes funciones con distintos dispositivos.

El control desde una página web en Visual Basic .NET con ASP.NET es otro punto a resaltar sobre lo aprendido a lo largo de los proyectos, pues desde una página web se tiene acceso a los datos de la placa Arduino YUN y se comunica con los otros elementos del sistema.

Para aumentar la complejidad de este proyecto, se puede añadir un shield GPS y un shield GSM/GPS, con el objetivo de darle la funcionalidad de rastreador remoto móvil y el control del mismo desde cualquier lugar, a través de la comunicación GPRS desde una página de Internet, puesto que el dispositivo se moverá a diferentes lugares y requiere del servicio de datos para enlace a Internet. También es posible añadirle al robot móvil una cámara web para crear un robot espía, o bien, colocar más sensores para rastrear al dispositivo, por ejemplo, un sensor giroscopio o un acelerómetro (para identificar la posición y los movimientos). Sin duda, con las herramientas y principios aprendidos se pueden realizar proyectos con aplicaciones para diferentes sectores, tales como el industrial, de control y de monitoreo, así como en el área de la Robótica, la Mecatrónica, las Redes de Datos, etcétera, y se podrán crear otros proyectos aplicando la creatividad y los principios adquiridos.

Y, si bien los conocimientos logrados y los proyectos creados implican grandes experiencias de aprendizaje, nunca es tarde para instruirse en algo nuevo y seguir adelante.

12.12 Problemas

1. Lleve a cabo el control de un relevador por voz.
2. Realice el control de encendido y apagado de las luces de una casa por voz.
3. Diseñe la apertura de una chapa electrónica por voz.
4. Elabore un control de riego automático por medio de voz.
5. Realice el control del sentido de giro de un motor de corriente directa por voz.
6. Efectúe el control de la velocidad de un motor de corriente directa por voz.
7. Elaborar el control de la posición de un servomotor por medio de voz.
8. Desarrolle el control de un led RGB por voz.
9. Diseñe una solicitud de temperatura y humedad por medio de voz.
10. Desarrollar un sistema de monitoreo de seguridad con cámara USB que trabaje desde una página web, agregue botones para controlar el movimiento de la cámara web con un servomotor.
11. Implemente un robot controlado desde una página web con detección de obstáculos.

12. Diseñe un robot espía que permita tomar video y lo envíe a una página web, que detecte movimiento y que tome fotos; agregue un sensor de movimiento PIR.
13. Diseñar un proyecto que detecte la voz de una persona y que, con base en el tipo de detección, tome una decisión.
14. Implementar un robot que permita ser controlado desde una página web, con botones, utilizando el módulo GSM/GPRS; esto se aplica en caso de que no exista Internet en una red inalámbrica.
15. Realice un sistema de seguridad con una cámara USB controlada desde una página web utilizando el módulo GSM/GPRS; esto se aplica en caso de que no exista Internet en una red inalámbrica.
16. Crear un proyecto de robot policía que cuide la entrada de una casa, con cámara web sensores, comunicación inalámbrica, detector de sonido y detector de presencia.

Bibliografía

- www.arduino.cc/tutorials
- *C Programming for Arduino*, Julien Bayle, Pack Publishing
- *Arduino Robotic Projects*, Richard Grimmett.
- *Arduino Home Automation: The builder Pack*, Marco Schwartz.
- *Internet of Things with Arduino Blueprints*, Pradeeka Seneviratne.
- *Arduino Development Cookbook*, Cornel Amariei.
- *Arduino for Secret Agents*, Marco Schwartz.
- *Arduino Networking*, Marco Schwartz.
- *Arduino Home Automation Projects*, Marco Schwartz.
- *Arduino Computer Vision Programming*, Özen Özkaya, Giray Yıllıkçı.
- *Internet of things with Arduino Yun*, Marco Schwartz.
- *Building Wireless Sensor Networks Using Arduino*, Matthijs Kooijman.
- *Arduino Essentials*, Francis Perea.
- *Building a Home Security System with Arduino*, Jorge R. Castro.
- *Arduino Electronics Blueprints*, Don Wilcher.
- *Aprenda practicando visual basic usando visual studio 2012*, Felipe Ramírez.
- *Aprenda SQL server 2012*, Francisco Salazar.
- *Sistemas SCADA*, Aquilino Rodríguez.

Índice analítico

D

E

M

N

O

P

R

S

T

V

W