



Subdirección Académica

Departamento de Sistemas y Computación

Ingeniería en Sistemas Computacionales

Semestre: Enero - Junio 2017

Materia: Sistemas Programables (3SC8A)

Nombre del tema:

Practica 13

Nombre del alumno:

Salcedo Morales José Manuel (13211419)

Nombre del catedrático:

Ingeniero Luis Alberto Mitre Padilla

Índice

1	Introducción	3
2	Componentes utilizados	3
3	Marco Teórico	4
4	Desarrollo	5
4.1	Imagenes	7
4.2	Diseño	11
4.3	Codigo	12
5	Conclusión	17

1 Introducción

En esta practica se hara de uso de software de aplicacion para hacer una comunicacion en conjunto con un Arduino.
En este caso, la aplicacion hecha en el lenguaje de programacion C#.

2 Componentes utilizados

- Arduino
- Cables Jumper
- Fuente de alimentacion para arduino
- Resistencias
 - 1KΩ
 - 2KΩ
- Microswitch
- Led's

3 Marco Teórico

- Arduino: Arduino se refiere a una plataforma o placa de electrónica de código abierto y al software utilizado para programarlo. Arduino está diseñado para hacer la electrónica más accesible a los artistas, diseñadores, aficionados y a cualquiera interesado en la creación de objetos interactivos o entornos. Un tablero de Arduino se puede comprar pre-ensamblado o, porque el diseño de hardware es de código abierto, construido a mano. De cualquier manera, los usuarios pueden adaptar las tablas a sus necesidades, así como actualizar y distribuir sus propias versiones.
- C#: Es un híbrido de C y C++, es un lenguaje de programación de Microsoft desarrollado para competir con el lenguaje Java de Sun. C# es un lenguaje de programación orientado a objetos utilizado con servicios Web basados en XML en la plataforma .NET y diseñado para mejorar la productividad en el desarrollo de aplicaciones web.
- LED: Un diodo emisor de luz (LED) es un dispositivo semiconductor que emite luz visible cuando una corriente eléctrica pasa a través de ella. La luz no es particularmente brillante, pero en la mayoría de los LED es monocromática, que ocurre en una sola longitud de onda.

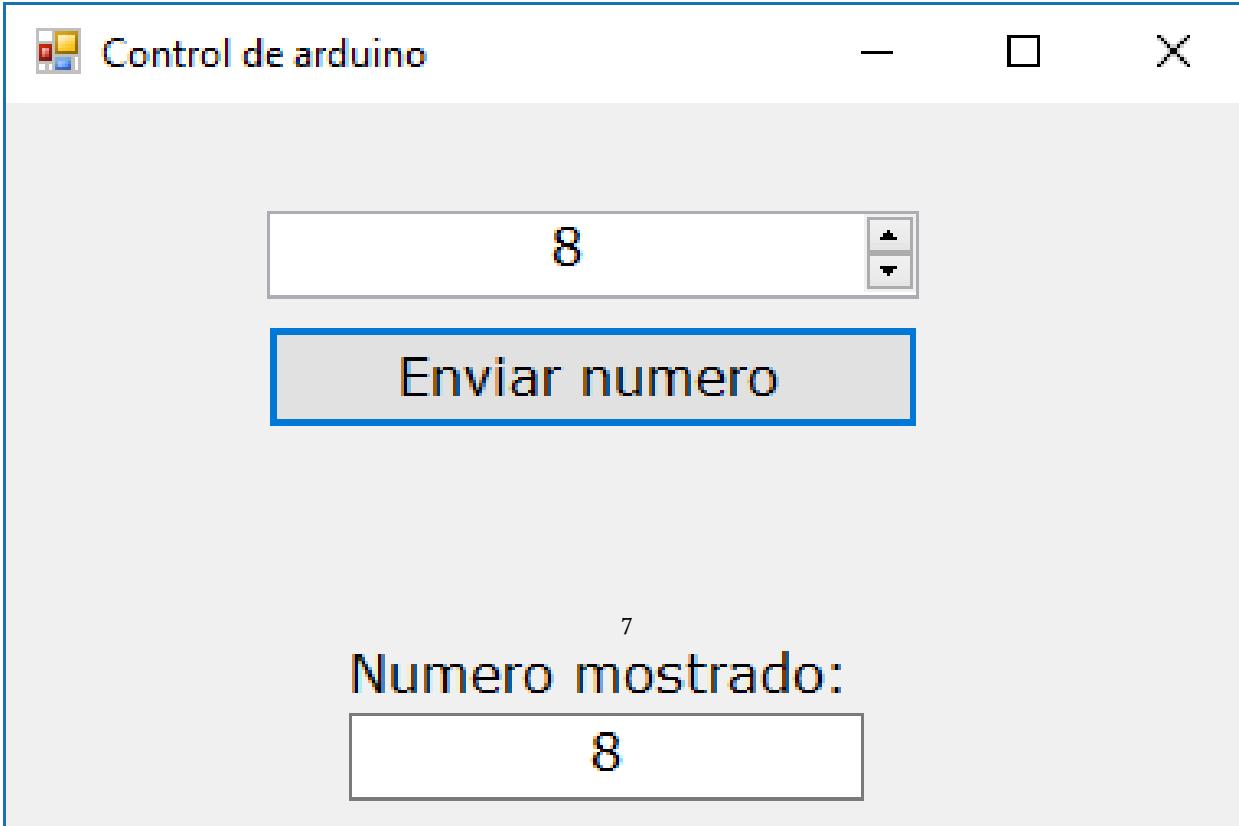
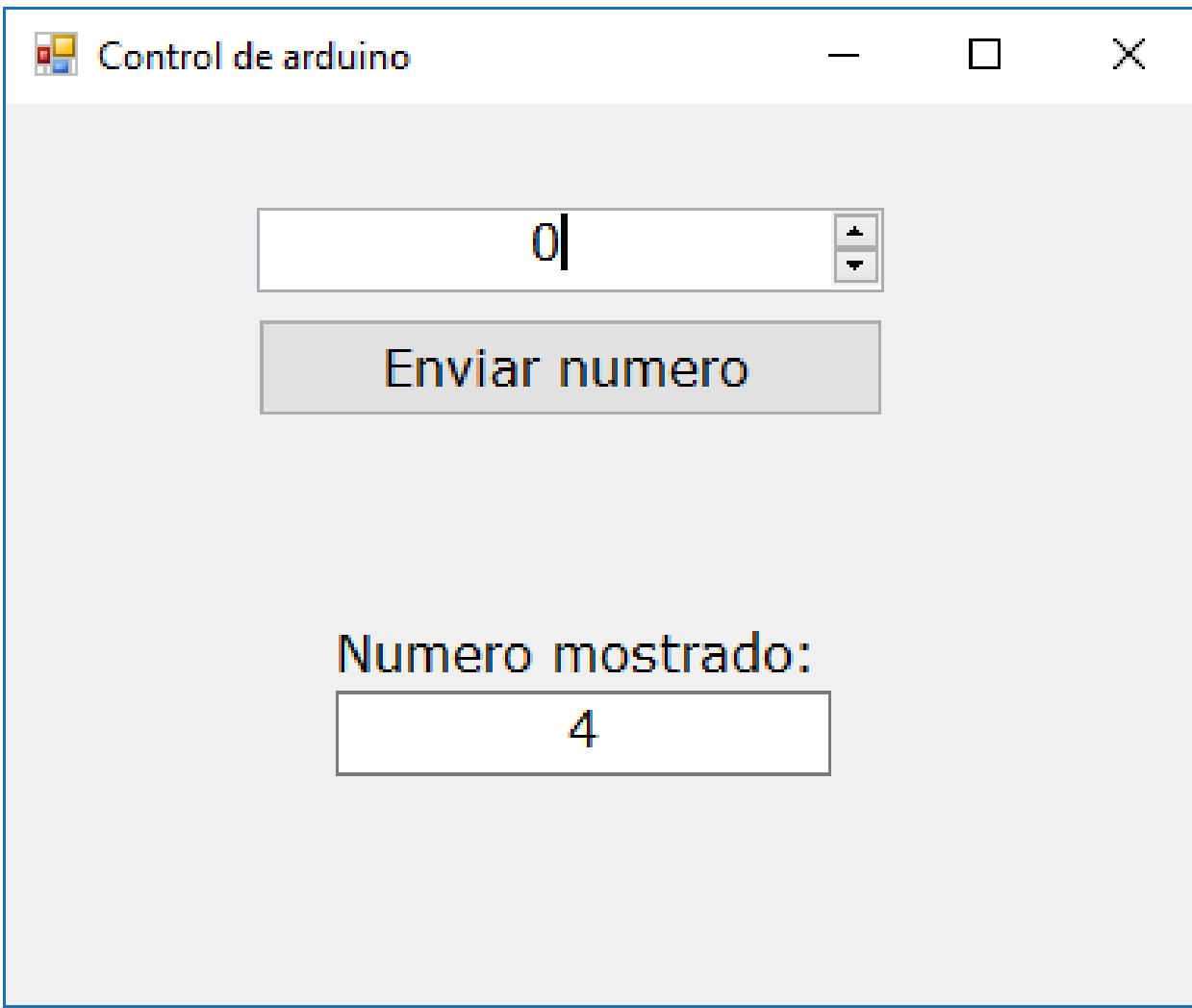
4 Desarrollo

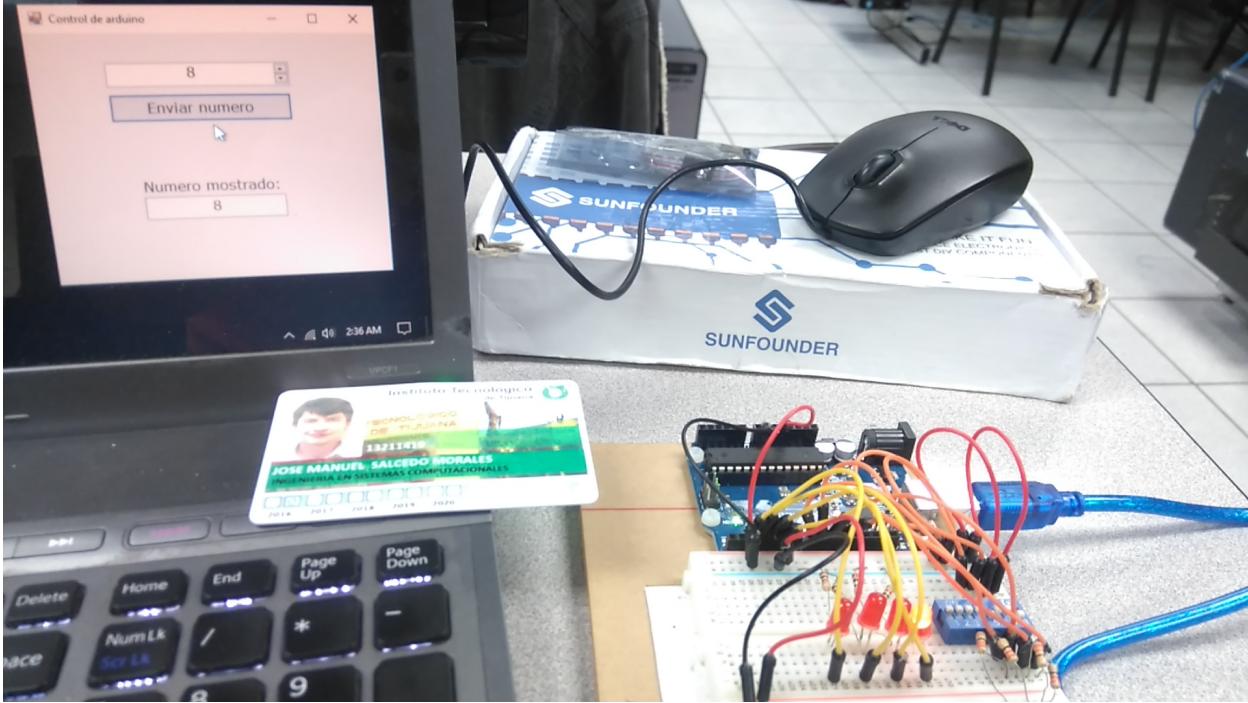
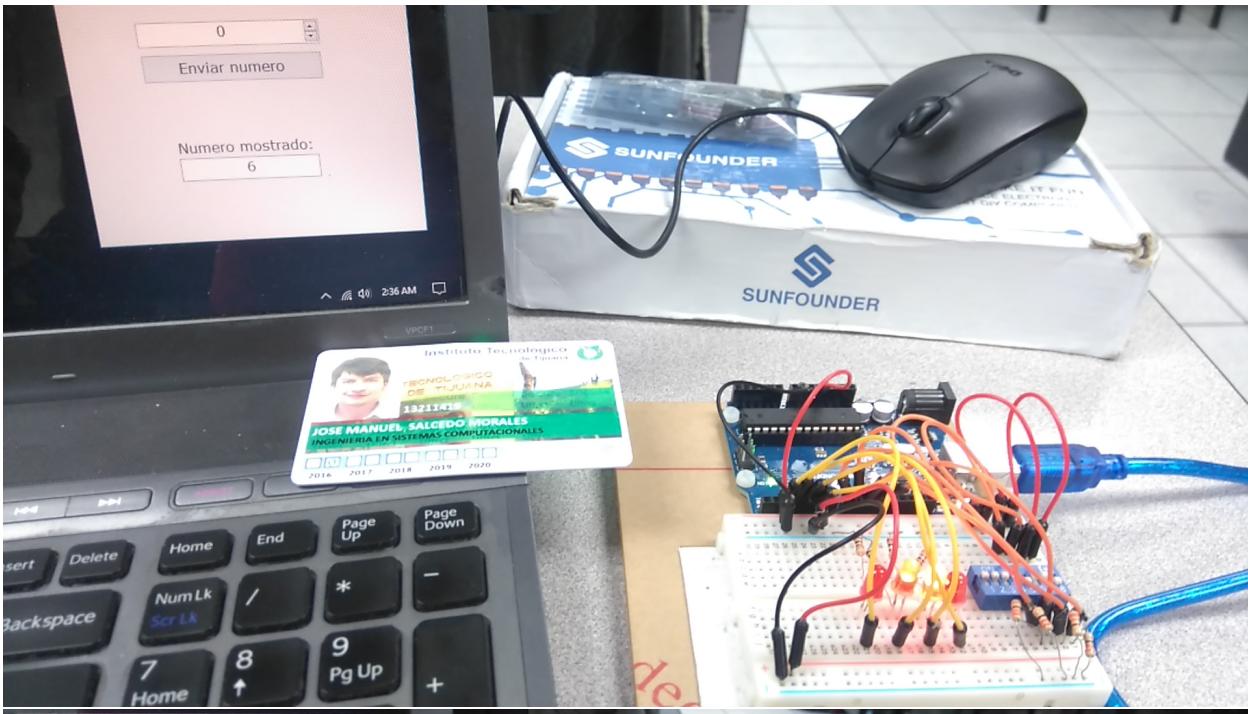
En cuanto a la aplicacion, esta controla el ingreso de un numero de 0 a 15 hacia el arduino. Subsecuentemente, el arduino se encarga de desplegar en binario en los led's el numero a representar.

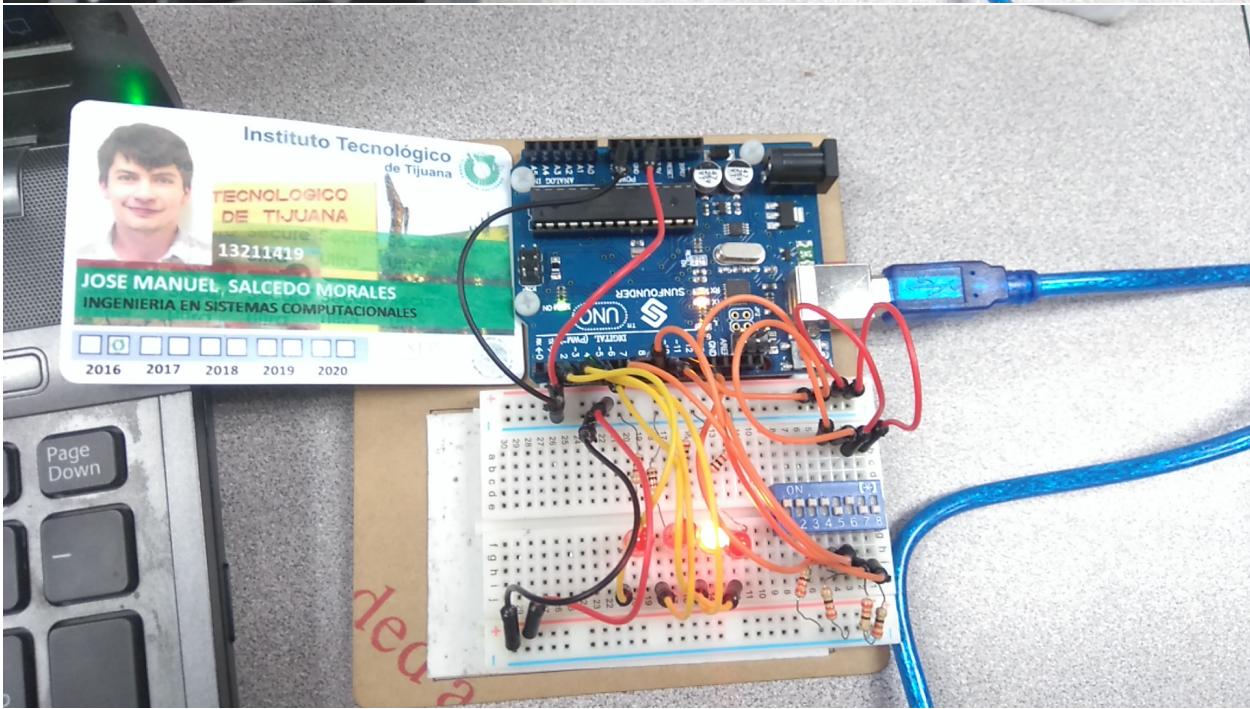
Si se interactua con el switch, se desplegara en pantalla el numero a representar en la aplicacion. De igual forma se representara en los led's de forma binario.

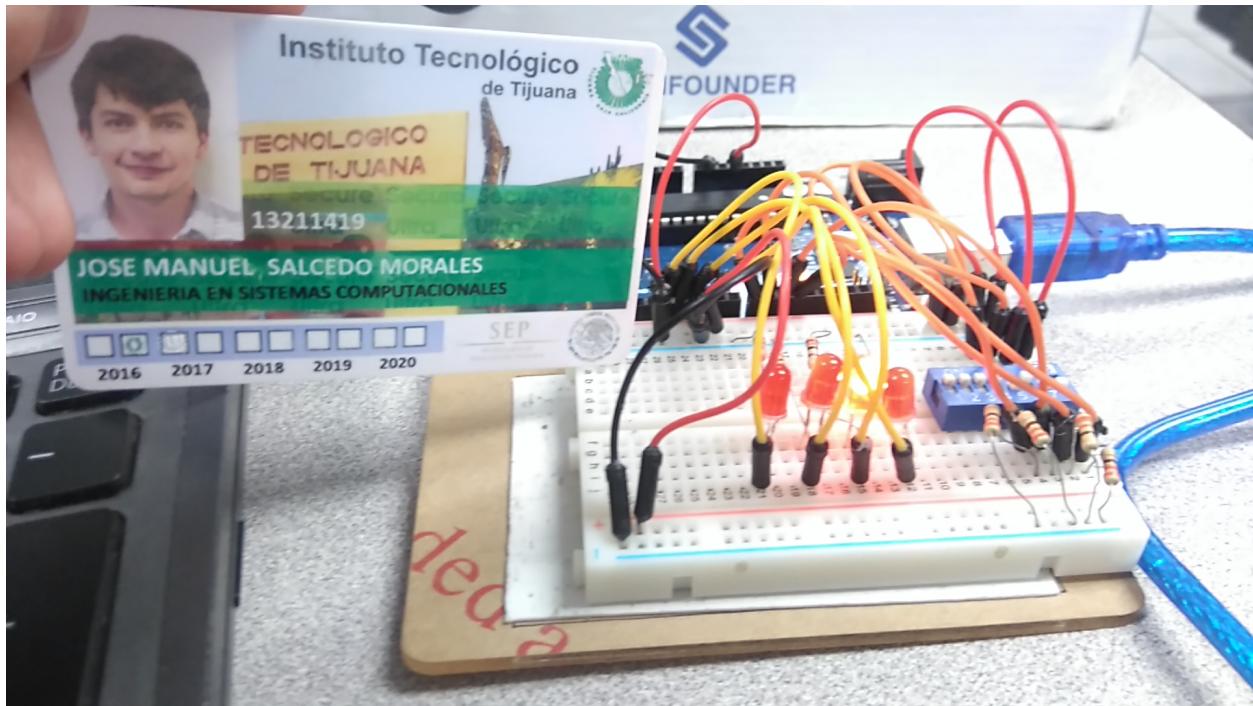
Toda la comunicacion por medio de un puerto USB.

4.1 Imagenes

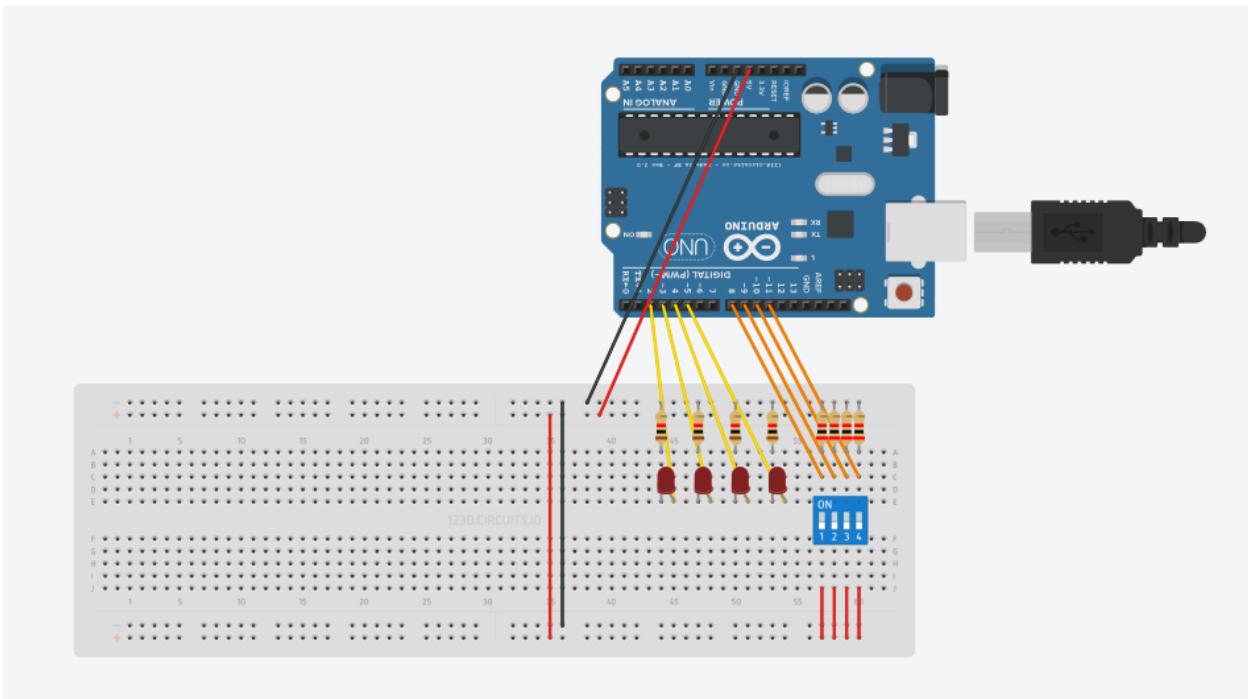








4.2 Diseño



4.3 Código

Código C#

```
using System;
using System.IO.Ports;
using System.Threading;
using System.Windows.Forms;

namespaceCodigo13
{
    public partial class FormaPrincipal : Form
    {
        private static readonly string PUERTO_ARDUINO = "COM4";
        private static readonly int BAUDIOS = 9600;
        SerialPort puertoArduino;

        public FormaPrincipal()
        {
            InitializeComponent();

            // Inicializar puerto a escribir.
            InicializarPuerto(PUERTO_ARDUINO, BAUDIOS);
        }

        private void InicializarPuerto(string puertoSistemaOperativo, int baudios)
        {
            puertoArduino = new SerialPort(puertoSistemaOperativo, baudios);
            try
            {
                // Reiniciar puerto.
                puertoArduino.DtrEnable = true;

                // Abrir puerto.
                puertoArduino.Open();

                // Preparar para cada de lectura de informacion del puerto.
                Thread hilo = new Thread(LeerHilo);
                hilo.Start(puertoArduino);
            }
            catch (Exception exception)
            {
                System.Diagnostics.Debug.WriteLine(exception.Message.ToString());
            }
        }

        private void LeerHilo(object contexto)
        {
            SerialPort puertoSerial = contexto as SerialPort;

            while (puertoSerial.IsOpen)
            {
                string datoLeido = puertoSerial.ReadLine();

                try
                {
                    this.BeginInvoke(new EventoLineaRecibida(LineaRecibida), datoLeido);
                }
                catch (Exception exception)
                {
                    System.Diagnostics.Debug.WriteLine(exception.Message.ToString());
                }
            }
        }

        private delegate void EventoLineaRecibida(string linea);
        private void LineaRecibida(string linea)
        {
            // Remover el caracter especial.
            linea = linea.Replace("\r", "");

            try
            {
                byte numero = 0;
                byte.TryParse(linea, out numero);
                elementoArduinoNumeroActual.Text = numero.ToString();
            }
            catch (Exception exception)
            {
                System.Diagnostics.Debug.WriteLine(exception.Message.ToString());
            }
        }

        private void CerrarPuerto(SerialPort puerto)
        {
            if (puerto != null)
                puerto.Close();
        }

        private void FormaPrincipal_FormClosing(object sender, FormClosingEventArgs e)
        {
            // Cerrar puerto antes de salir de ejecucion.
            CerrarPuerto(puertoArduino);
        }

        private void elementoEnviarNumeroArduino_Click(object sender, EventArgs e)
        {
            // Capturar numero a enviar.
            byte numeroCapturado = (byte)elementoNumeroArduino.Value;

            // Escribir numero a arduino.
        }
    }
}
```

```
if (numeroCapturado >= 0 && numeroCapturado <= 255)
{
    // Convertir a formato apropiado para enviar a Arduino.
    byte[] numeroEnvio = BitConverter.GetBytes(numeroCapturado);
    try
    {
        puertoArduino.Write(numeroEnvio, 0, 1);
    }
    catch (Exception exception)
    {
        System.Diagnostics.Debug.WriteLine(exception.Message.ToString());
    }
}
}
```

Código Arduino

```
// obtener un numero en binario.
String ObtenerNumeroEnBinario(unsigned long numero)
{
    // Maximo de bytes a usar para un numero binario.
    const int BYTES_BINARIO_MAXIMO = 32;

    // Definir texto del numero binario a retornar.
    String numeroBinario = "";

    // Si el numero es igual o menor a 0, definir el numero como 0.
    if (numero <= 0) {
        numeroBinario = "0";
    } else {
        // Repetir por cada byte posible en un numero binario.
        for (int numeroByte = BYTES_BINARIO_MAXIMO; numeroByte >= 0; numeroByte--) {
            // Obtener el numero de potencia de acuerdo al numero del byte actual.
            const unsigned long numeroPotenciaActual =
                (unsigned long)(round(pow(2, numeroByte)));

            // Si el resultado de la potencia obtenida es menor o igual al numero
            // a convertir, a esta posicion de byte le pertenece un 1.
            if (numeroPotenciaActual <= numero) {
                numeroBinario += "1";

                // Restar la potencia actual al numero para no repetir el mismo byte.
                numero = numero - numeroPotenciaActual;

                // Si el numero no es 1, si hay numeros en el texto binario agregar un 0.
            } else if (numeroBinario != "") {
                numeroBinario += "0";
            }
        }
    }

    return numeroBinario;
}

// Prender una colección de led's de acuerdo a un numero,
// representandolo de forma binaria.
void RepresentarNumeroEnBinarioEnPines(unsigned int numero,
                                         const unsigned int colecciónPines[], const unsigned int cantidadPines)
{
    // Obtener numero decimal en binario textual.
    const String numeroBinario = ObtenerNumeroEnBinario((unsigned long)numero);

    // Obtener la cantidad de digitos en el numero binario.
    const unsigned int cantidadDigitos = numeroBinario.length();

    // Iniciar desde el ultimo digito.
    unsigned int digitoActual = cantidadDigitos - 1;
    // Recorrer todos los pines existentes.
    for (unsigned int pinActual = 0; pinActual < cantidadPines; pinActual++) {
        // Capturar el pin a encender.
        const unsigned int pinEncender = colecciónPines[pinActual];

        // Continuar si el numero del digito actual es menor
        // a la cantidad de pines permitidos a usar.
        // Y el pin actual es menor a la cantidad de digitos a representar.
        if (digitoActual < cantidadPines && pinActual < cantidadDigitos) {
            // Capturar el digito actual del numero binario.
            const char numeroDigito = numeroBinario.charAt(digitoActual);

            // Prender o apagar el pin de acuerdo al equivalente del numero binario.
            if (numeroDigito == '1') {
                digitalWrite(pinEncender, HIGH);
            } else if (numeroDigito == '0') {
                digitalWrite(pinEncender, LOW);
            }

            // Operar en un digito hacia la izquierda.
            digitoActual -= 1;
        } else {
            digitalWrite(pinEncender, LOW);
        }
    }
}

// Obtener un numero en binario.
unsigned long ObtenerNumeroEnDecimal(String numeroBinario)
{
    // Definir numero decimal a retornar.
    unsigned long numeroDecimal = 0;

    // Obtener la cantidad de digitos en el numero binario.
    const byte cantidadDigitos = numeroBinario.length();

    // Repetir por cada digito en el numero binario.
    for (short digito = cantidadDigitos - 1; digito >= 0; digito--) {
        // Capturar el digito actual del numero binario.
        const char numeroDigito = numeroBinario.charAt(digito);

        // Agregar el numero decimal de acuerdo al digito actual si esta activado.
        if (numeroDigito == '1') {
            numeroDecimal = numeroDecimal + ((unsigned long)(round(pow(2, cantidadDigitos - (digito + 1)))));
        }
    }
}

return numeroDecimal;
}
```

```

// CONSTANTES.
const unsigned int CANTIDAD_LEDS = 4;
const unsigned int pinLed[] = {2, 3, 4, 5};
const unsigned int CANTIDAD_SWITCHES = 4;
const unsigned int pinSwitch[] = {8, 9, 10, 11};

void setup()
{
    // Inicializar lectura de serial.
    Serial.begin(9600);

    // Definir pines de salida para leds.
    unsigned int cantidadPinesLed = sizeof(pinLed);
    for (byte led = 0; led < cantidadPinesLed; led++) {
        pinMode(pinLed[led], OUTPUT);
    }

    // Definir pines de entrada para switch.
    unsigned int cantidadPinesSwitch = sizeof(pinSwitch);
    for (byte switchPin = 0; switchPin < cantidadPinesSwitch; switchPin++) {
        pinMode(pinSwitch[switchPin], INPUT);
    }
}

// VARIABLES GLOBALES.
byte datoEntrante = 255;
byte datoEntranteAnterior = 255;
bool estadosSwitchAnterior[] = {true, true, true, true};
byte valorNumero = 0;

void loop()
{
    if (Serial.available() > 0) {
        // Leer un dato enviada desde el dispositivo conectado.
        datoEntrante = Serial.read();
    }

    // Definir si cambio el dato enviado.
    bool datoEntranteDistinto = false;
    if (datoEntrante != datoEntranteAnterior) {
        datoEntranteDistinto = true;
    }
    datoEntranteAnterior = datoEntrante;

    // Obtener cantidad de pines del switch.
    unsigned int cantidadPinesSwitch = CANTIDAD_SWITCHES;

    // Leer los valores del switch si el dato entrante es distinto.
    bool valoresSwitch[cantidadPinesSwitch];
    bool valoresDistintosSwitch = false;
    if (datoEntranteDistinto == false) {
        for (byte switchPin = 0; switchPin < cantidadPinesSwitch; switchPin++) {
            valoresSwitch[switchPin] = digitalRead(pinSwitch[switchPin]);
            //Serial.println("Pin " + String(switchPin) + ": " + String(valoresSwitch[switchPin]));
        }

        // Determinar si hubo un cambio de estado.
        if (valoresSwitch[switchPin] != estadosSwitchAnterior[switchPin]) {
            valoresDistintosSwitch = true;
        }
        // Guardar los valores actuales del switch.
        estadosSwitchAnterior[switchPin] = valoresSwitch[switchPin];
    }
}

//Serial.println(valoresDistintosSwitch);
// Si el switch no tiene valores distintos,
// continuar si hay conexion serial detectada.
if (valoresDistintosSwitch == false && datoEntranteDistinto == true)
{
    // Guardar el dato como numero a representar.
    valorNumero = datoEntrante;
}
else if (valoresDistintosSwitch == true) {
    //Serial.println("Valores distintos.");
    // Convertir los valores del switch a binario.
    String numeroBinario = "";

    // Recorrer cada pin del switch para determinar el numero binario a representar.
    for (short switchPin = cantidadPinesSwitch - 1; switchPin >= 0; switchPin--) {
        // Capturar estado del switch actual.
        bool estadoPin = valoresSwitch[switchPin];

        // Si el estado del pin esta prendido, agregar 1.
        // De lo contrario, agregar 0.
        if (estadoPin == true) {
            numeroBinario = numeroBinario + "1";
        } else if (estadoPin == false) {
            numeroBinario = numeroBinario + "0";
        }
    }

    // Definir el numero a representar.
    valorNumero = (byte)(ObtenerNumeroEnDecimal(numeroBinario));
}

Serial.println(valorNumero, DEC);

// Representar el dato capturado en los pines apropiados si hubo cambio de numero.
if (valoresDistintosSwitch || datoEntranteDistinto) {
    RepresentarNumeroEnBinarioEnPines((unsigned int) valorNumero,
                                      pinLed, CANTIDAD_LEDS);
}

```

}

5 Conclusión

Por si solo, el Arduino proporciona una buena manera para controlar componentes electronicos. Combinado con software para control, la usabilidad y control se extiende a mejores posibilidades.

Referencias

- [1] What is Arduino? - Definition from Techopedia. (n.d.). Retrieved March 26, 2017, from <https://www.techopedia.com/definition/27874/arduino>
- [2] Staff, -. W. (n.d.). C#. Retrieved May 11, 2017, from http://www.webopedia.com/TERM/C/C_sharp.html
- [3] Rouse, M. (n.d.). Light-emitting diode (LED) . Retrieved February 13, 2017, from <http://whatis.techtarget.com/definition/light-emitting-diode-LED>