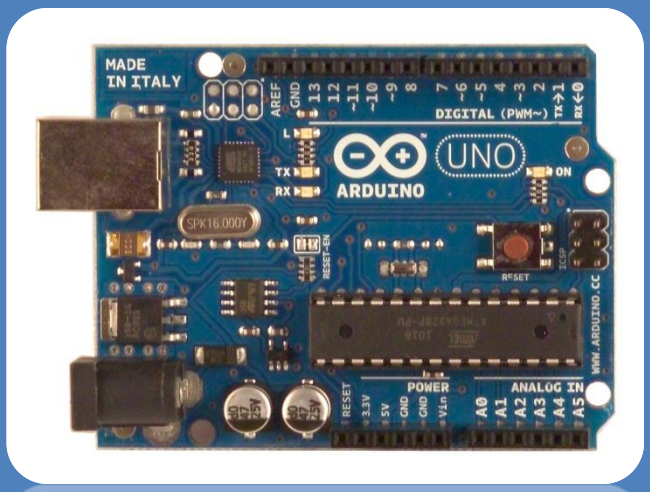

ARDUINO UNO

Manual

Universidad del Valle de México
Campus Tuxtla. Ingeniería Mecatrónica Mayo 2011

Introducción

En los últimos años se han desarrollado innumerables aplicaciones en las que es necesario contar con los conocimientos básicos en el desarrollo y programación de microcontroladores. La mayoría de los sistemas programables actuales tienen por lo menos un microcontrolador encargado del control operativo del sistema.

Existen en el mercado muchos fabricantes de microcontroladores, por mencionar algunos: MICROCHIP, ATMEGA, MOTOROLA entre otras. Estos fabricantes proveen del software especializado para la programación de sus microcontroladores y otorgan gran cantidad de información para el usuario.

Actualmente, ARDUINO, una empresa italiana, ha desarrollado placas microcontroladas educativas con grandes prestaciones. Esta placa posee microcontroladores ATMEGA encargados del control de la placa. Hay disponibles gran cantidad de proyectos que se han desarrollado a través de esta noble interfaz.

ARDUINO UNO

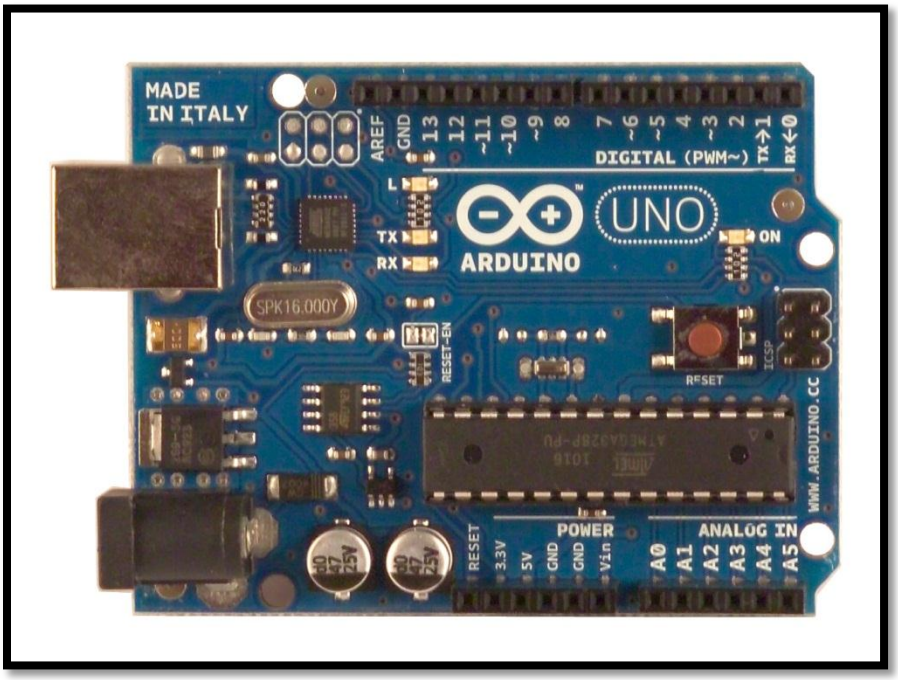

Figura 1
(Parte delantera de la tarjeta ARDUINO UNO)

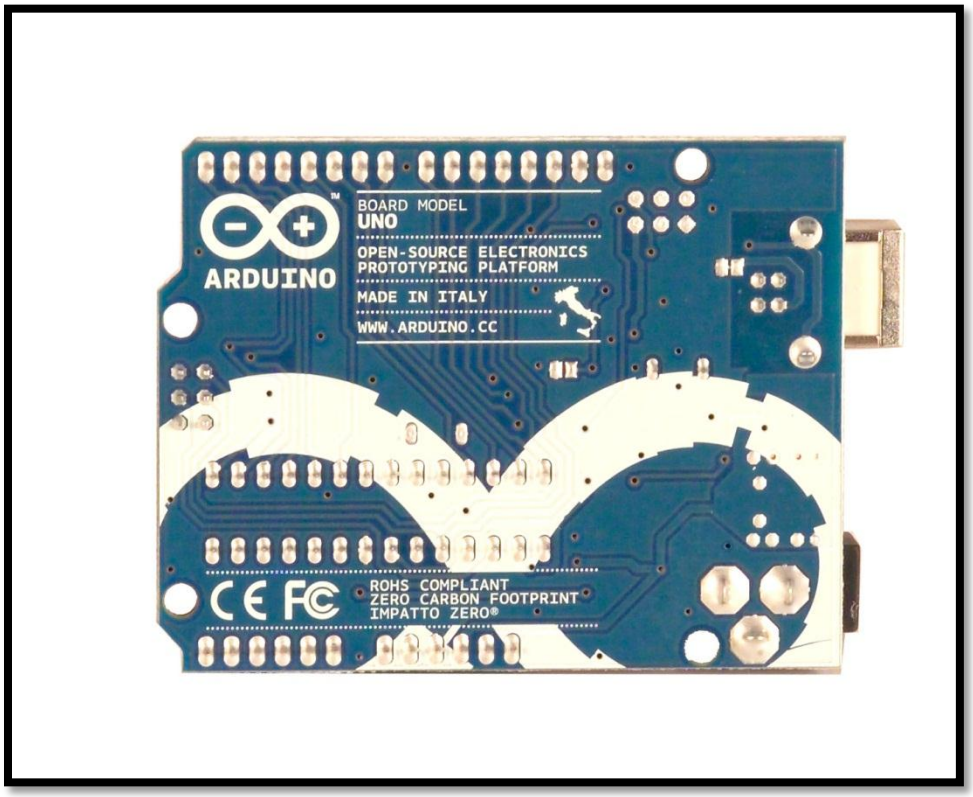

Figura 2
(Parte posterior de la tarjeta ARDUINO UNO)

GENERALIDADES

La Arduino UNO es una tarjeta educativa basada en el microcontrolador ATmega328. Tiene 14 pines de entrada/salida digitales (de los cuales 6 puede utilizarse como salidas PWM), 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP y un botón “reset”. Contiene todo lo necesario para programar y comenzar a usar el microcontrolador; simplemente conéctelo a un equipo con un cable USB o de alimentación con un adaptador AC a DC o batería.

La UNO difiere de todas las placas anteriores puesto que no utiliza el chip de controlador de FTDI USB a puerto serie. En cambio, cuenta con el Atmega8U2 programado como un convertidor de USB a puerto serie.

ESPECIFICACIONES.

Microcontrolador	ATmega328
Voltaje Operativo	5V
Voltaje de entrada(recomendado)	7-12V
Voltaje de entrada (limites)	6-20V
Digitales I/O Pins	14 (6 Como salidas PWM)
Entradas Analógicas Pins	6
Corriente CD por I/O Pin	40 mA
Corriente CD por 3.3V Pin	50 mA
Memoria Flash	32 KB (ATmega328) of which 0.5 KB usado para el cargador de programa.
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de Reloj	16 MHz

ALIMENTACIÓN

La tarjeta ARDUINO UNO, puede ser conectada directamente al Puerto USB de su computador y puede obtener de ahí la tensión y la corriente necesaria para bajas cargas. La placa ARDUINO también puede operar con voltajes externos regulados, pudiéndola alimentar a través de las terminales Vin y GND.
Los pines de alimentación son los siguientes:

- **VIN.** Pin por el cual se puede introducir una fuente regulada de 5 volts.
- **5V.**Esta es la alimentación regulada al momento de conectar una pila de mayor voltaje al conector tipo plug.
- **3V3.** Voltajes generados por la placa a través de los pines indicados de tres volts.
- **GND**. Pines de Tierra o masa.

MEMORIA

El ATmega328 tiene 32 KB de memoria flash (con 0.5 KB usado para el pre quemador). También tiene 2 KB of SRAM y 1 KB of EEPROM (la cual puede ser leída y escrita por la EEPROM).

ENTRADAS Y SALIDAS DIGITALES

Cada uno de los 14 pines de conexiones que posee la Arduino, pueden ser configuradas como entradas y como salidas digitales.

Con las funciones para entradas/salidas digitales, podemos hacer la configuración en la programación

- pinMode() : Indica al programa el modo entrada o salida del terminal.
- digitalWrite(): Escribe un bit en el terminal seleccionado.
- digitalRead(): Lee un valor digital desde el terminal seleccionado.

Las entradas/salidas digitales operan con 5 volts. Cada pin de entrada y salida puede suministrar hasta **20 mA de corriente**. Algunas entradas/salidas tienen funciones especiales de comunicación serial: 0 (RX) y 1 (TX). Usadas para la comunicación con la computadora, por lo cual se sugiere la restricción de su uso.

PWM: 3, 5, 6, 9, 10, y 11. Proveen señales cuadradas de 8-bits controlado por la función analogWrite ().

ESQUEMA

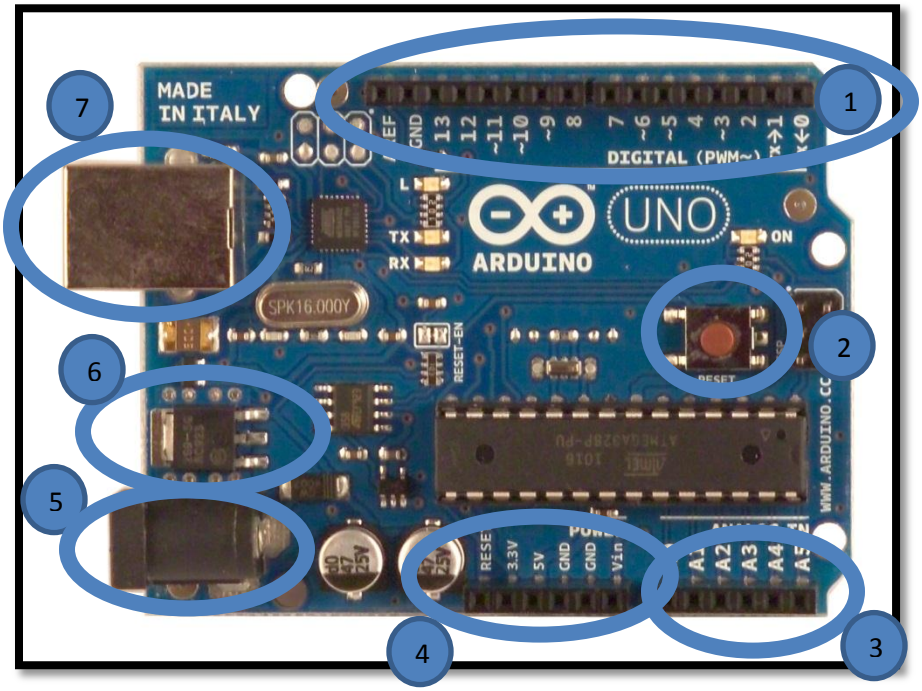

Figura 3
(Partes importantes de la tarjeta ARDUINO)

En la figura tres, podemos observar resaltadas las partes que es necesario conocer para el correcto conexionado de la tarjeta.

1. Línea de comunicaciones pines 0 y 1 no se usan. Entradas/salidas digitales pines del 2 al 13.
2. Botón “reset” de la tarjeta (Permite el re-inicio de la misma).
3. Línea de Entradas analógicas (De la A0 a la A5).
4. Línea de alimentación. En estos pines podemos encontrar Vin, GND, 5V, 3.3V y reset.
5. Plug de alimentación de la tarjeta (Para voltajes entre 7 a 12 volts máximo).
6. Regulador de voltaje.
7. Conector USB.

PRACTICAS PROPUESTAS (ENTRADAS/SALIDAS DIGITALES)

PRACTICA 1 "*ENCENDER UN LED*"

```
int LED = 4; //Se asigna a una variable el pin

void setup() //Función donde se definen como funcionaran los pines o comunicaciones
 {

pinMode(LED,OUTPUT); //Se asigna a la variable LED como salida en este caso sera el pin 4 funcionara como salida

 }

 void loop() //Función donde se ejecuta el programa
 {
   digitalWrite(LED,1); // EL pin LED se le manda un alto para que encienda
 }
```

PRACTICA 2 *"APAGAR UN LED"*

```
int LED = 4; //Se asigna a una variable el pin

void setup() //Función donde se definen como funcionaran los pines o comunicaciones
 {

pinMode(LED,OUTPUT); //Se asigna a la variable LED como salida en este caso sera el pin 4 funcionara como salida

 }

 void loop() //Función donde se ejecuta el programa
 {
   digitalWrite(LED,0); // EL pin LED se le manda a bajo para que se apague
 }
```

PRACTICA 3 *"BLINKEO DE UN LED"*

Este programa encenderá un led en cierto tiempo y lo apagara en el mismo tiempo, esto visualizara un blinkeo.

```
int LED = 4; //Se asigna a una variable el pin

void setup() //Función donde se definen como funcionaran los pines o comunicaciones
 {
```

```
pinMode(LED,OUTPUT); //Se asigna a la variable LED como salida en este caso sera el pin 4 funcionara como salida
 }

 void loop() //Función donde se ejecuta el programa
 {
```

PRACTICA 4 *"SECUENCIA DE 3 LEDs"*

Este programa encenderá 3 led con la diferencia que uno encenderá después del otro, simulando una serie de luces de navidad.

```
int LED1 = 4; //Se asigna a una variable el pin
int LED2 = 5;
int LED3 = 6;

void setup() //Función donde se definen como funcionaran los pines o comunicaciones
 {
pinMode(LED1,OUTPUT);
pinMode(LED2,OUTPUT);
pinMode(LED3,OUTPUT);
 }

 void loop() //Función donde se ejecuta el programa
 {

   digitalWrite(LED1,1;
   delay(100);
   digitalWrite(LED1,0);
   digitalWrite(LED2,1);
   delay(100);
   digitalWrite(LED2,0);
   digitalWrite(LED3,1);
   delay(100);
   digitalWrite(LED3,0);

 }
```

PRACTICA 5 *"ENCENDER Y APAGAR UN LED CON PUSH-BOTON"*

En este programa se encenderá un led y se apagara por medio de dos botones, el primer botón encenderá el led y el segundo lo apagara. Es necesario 2 push boton y 2 resistencias de 1kohm

```
int LED = 4;
int boton = 5;
int boton2 = 6;

int x=0;

void setup() //Función donde se definen como funcionaran los pines o comunicaciones
  {
pinMode(LED,OUTPUT);
pinMode(boton,INPUT);
pinMode(boton2,INPUT);
  }

  void loop() //Función donde se ejecuta el programa
  {

   if(digitalRead(boton)==1) //Se lee la entrada del pin botón y se compara
    {
      x=1; //Se asigna un valor a la variable x
    }

   if(digitalRead(boton2)==1) //Se lee la entrada del pin botón y se compara esta condición
es la contraria a la anterior
    {
      x=0; //Se asigna un valor a la variable x
    }

if(digitalRead(boton)==1 && digitalRead(boton2)==1) //En caso de que se presionen los 2
botones
{
      x=0; //Se asigna un valor a la variable x
    }

if(x==1) //Si la variable x es igual a 1 enciende el led
{
    digitalWrite(LED,1); // EL pin LED se le manda un alto para que encienda
}

if(x==0) //Si la variable x es igual a 0 apaga el led
{
    digitalWrite(LED,0); // EL pin LED se le manda un alto para que encienda
}

  }
```

PRACTICA 6 *"USO DE LA FUNCIÓN println"*

En esta practica se visualizara en el monitor de Arduino la impresión de un texto con el comando Serial.println, pero para esto se tiene que inicializar la comunicación serial con Arduino para esto existe la función Serial.begin(9600);

```
void setup()
{
  Serial.begin(9600);
  Serial.println("  HOLA MUNDO ");
}
void loop()
{}
```

PRACTICA 7 "CONTADOR CON PUSH-BOTON"

Hay diferentes formas de hacer un contador, en este caso se hará por medio de un pulsador que incrementara el valor de una variable y este valor se visualizara en el monitor de Arduino.

```
int conta=0; //Inicializacion del contador

  void setup()
  {
    Serial.begin(9600); //Comunicacion Serial arduino
    pinMode(4,INPUT); //Se define el pin 4 como entrada
  }

  void loop()
  {
    if(digitalRead(4)==1) //Condicion si es presionado el pulsador
    {
      conta=conta+1; //incremento de la variable
      delay(200); //Se coloca un ratardo para que cuando se presione el pulsador este no incremente de mas

    }
    if(digitalRead(4)==0) //Cuando el boton es soltado mantiene el valor de la variable
      conta=conta;
    Serial.println(conta,DEC); //Impresion de la variable en el monitor de arduino
    delay(300);
  }
```

PRACTICA 8 *"USANDO CICLO FOR "*

Ejemplo de un ciclo For e imprimir los valores en el monitor de Arduino, para esto hay que conocer como funciona dicho ciclo:

```
int i; //Se inicializa una variable que servira en el ciclo for
void setup()
{
  Serial.begin(9600); //Inicialización de la comunicación serial con arduino
}
void loop()
{
  for(i=0;i<=5;i++) //Ciclo for. La primer sentencia es donde se inicializa la variable con un valor, la segunda sentencia es para indicar la condición para que el ciclo siga, y por ultimo es el incremento de la variable.
  {
    Serial.print("Valor "); //Impresión de la palabra "Valor" en el monitor de arduino
    Serial.println(i,DEC); //en este se imprime el valor de la variable "i" y la referenciamos que es un valor decimal con "DEC"
    delay(400); //Se pone un retraso para que a la hora de visualizar los valores de la variable sean mas lentos
    Serial.println(); //Impresión de un renglón vacio
  }
  delay(800); //Retraso para la nueva ejecución del ciclo
}
```

ENTRADAS ANALÓGICAS

La tarjeta Arduino UNO posee 6 entradas analógicas, que pueden ser leídas a partir de una tensión de entrada de 0 volts hasta 5 volts. La resolución del convertidor analógico/digital es de 8 bits.

La resolución del convertidor puede ser modificada a través de la tensión de referencia denominada Vref en el placa Arduino.

PRACTICA 9 *"LECTURA ANALOGICA"*

Con este programa se puede leer valores analógicos del pin A0 de Arduino. Este programa podría leer el valor que manda un potenciómetro con la diferencia que lo leería en valor a bits, a su vez este programa imprimirá el valor en el monitor de Arduino.

```
int valor; //Inicialización de la variable que representara el valor del potenciómetro

void setup()
{
```

```
  Serial.begin(9600);
}
void loop()
{
  valor=analogRead(A0); Con la sentencia analogRead(A0) leerá el valor de la entrada analógica del pin A0 y lo guardara en la variable valor.

  Serial.println(valor,DEC); //Sentencia que imprime el valor de la variable.
  delay(400); //Retardo con el fin de que se logre observar el valor de la variable ya que este programa se ejecutara de manera cíclica, es decir, estará leyendo el valor del pin A0.
}
```

PRACTICA 10 *"ASIGNACIÓN DE UN RANGO ANALOGICO"*

Con la finalidad a observar cambios con los valores, se hará una referencia con el valor analógico, es decir se estará leyendo continuamente la entrada analógica A0 y a este se comparara con unas condiciones. La primera es de que si el valor analógico obtenido es menor al de la referencia de la condición apagara el Led y si es mayor lo encenderá.

```
int valor; //Inicialización de la variable que representara el valor del potenciómetro
int led=4; //Asignación del pin

void setup()
{
  Serial.begin(9600);
  pinMode(led,OUTPUT);
}
void loop()
{
  valor=analogRead(A0);
  Serial.println(valor,DEC);
  delay(400);
  if(valor < 500)
          digitalWrite(led,0);
if(valor >= 500)
          digitalWrite(led,1);
}
```

Nota: El valor máximo que puede entregar la entrada analógica es de 1023 bits.

Modulación por ancho de pulsos

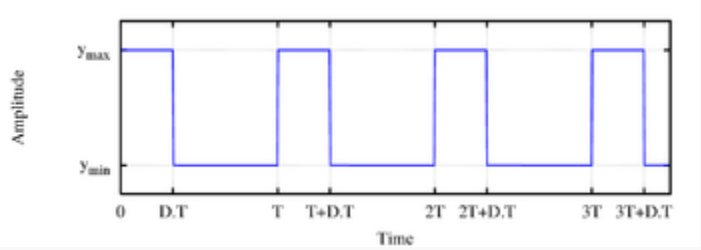

Fig. 1: una señal de onda cuadrada de amplitud acotada (y_{min},y_{max}) mostrando el ciclo de trabajo D.

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de *pulse-width modulation*) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función

La frecuencia utilizada por la señal PWM es de 500 HZ. La llamada a la function analogWrite(), proporciona un PWM con una escala de 0 a 255 valores digitales. Por ejemplo: si se hace la llamada de la siguiente forma analogWrite(255), se obtiene un ciclo útil del 100% de la señal PWM.

PRACTICA 11 *"ANCHO DE PULSO MODULADO (PWM)"*

En este programa se visualizara de forma ciclica la intensidad luminosa de un led, aumentara su intensidad y después disminuira hasta que se apague.Cave destacar que la salida máxima de PWM que contiene Arduino UNO es de 255 bits.

```
int A=0; //Inicialización de una variable que representara el PWM
int B=5; //Inicialización de una variable para el incremento del PWM
```

```
void setup()
{
  pinMode(3,OUTPUT); //Asignación del pin 3 como salida
}

void loop()
{
  analogWrite(3,A); //sentencia que manda el PWM al pin 3
  A=A+B; //Incremento de la variable
  if(A==255) //En caso de que la variable A sea igual a 255 se asigna un nuevo valor a
este
    {A=0;}
  if(A==255 || A==0) // para que se observe de forma inversa el PWM se hace un
decremento
  {B=-B;}
}
```

PRACTICA 12 *"CONTROL DE LUMINOSIDAD CON PWM"*

En las practicas anteriores se ha aprendido a leer valores digitales y analógicos, asignar variables, imprimir valores o palabras en el monitor de Arduino. En esta practica trata de leer un valor analógico y este convertirlo a una salida PWM con el fin de controlar la intensidad luminosa de un led.

```
int pwm; //Variable para el valor del PWM
int led=4; //Pin 4
int valor; //Variable para la lectura analogica
void setup()
{
  Serial.begin(9600); //Comunicacion serial arduino
  pinMode(led,OUTPUT); //asignacion de salida al pin led(4)
}
void loop()
{
  valor=analogRead(A0); //Lectura analogica
  pwm=(valor*255)/1023; //Conversion de bits a PWM
  Serial.print("Analogico - ");
  Serial.println(valor,DEC); //Impresion del valor analogico
  delay(100);
  Serial.println();
  Serial.print("PWM - ");
```

```
  Serial.println(pwm,DEC); //impresion del valor PWM
    delay(200);
analogWrite(led,pwm); //enviar PWM a pin led
}
```

Nota: Esta practica también podría controla la velocidad de un motor, con la diferencia que tendría que llevar un circuito extra entre la salida PWM y el motor. Un ejemplo de este seria a un puente H (L293b), la señal PWM se mandaría al enable de este integrado.

PRACTICA 13 *"MANDAR DATOS CON EL MONITOR"*

Este programa es de encender leds pero enviando datos al puerto serial de Arduino. Los pines a activar serán del pin 3 al 13 ya que los otros funcionan como comunicación. También la forma en que Arduino lee los datos es diferente ya que no lee el dato en forma decimal sino en ASCII entonces se tiene que hacer una conversión de este para poder seleccionar que pin activar. Otro detalle de este es que no lee palabras sino bits, entonces hay que tener CUIDADO al enviar los datos ya que de preferencia se deben de enviar UNO POR UNO.

```
int dato; //Variable para la lectura del puerto serial
  int x=0;
  int y=0;
  void setup()
  {
    Serial.begin(9600); //Comunicacion con el puerto
    for(int pin = 3 ; pin <= 13 ; pin++) //Este ciclo asignara a los pines 3 al 13 como salida
    {
      pinMode(pin,OUTPUT);
    }
  }
  void loop()
  {
    if(Serial.available()>0) //Esta condicion es para eliminar ruido de la entrada serial
    {
      dato=Serial.read(); //Con la sentencia Serial.read(); se lee los datos seriales
      if(dato>='3' && dato<='9') //Con esta condicion se seleccionara que pin entre 3 y 9
para encender o apagar
      {
        dato=dato-'0'; //Con este se hace la conversion de ASCI a decimal
        y=1+x; //Con este es para hacer que conmuete entre encendido y apagado
        x=y;
        digitalWrite(dato,y); //Con esta condicion manda activar que pin a encender o apagar
      }
```

```
    if(y==1) //Con este es para hacer que conmuete entre encendido y apagado
    {
      x=-x;
    }
   }
 }
```

PRACTICA 14 *"CONVERSION DE SISTEMA A BINARIO – OCTAL – HEXADECIMAL – ASCII - DECIMAL"*

Este programa tiene como finalidad mostrar el valor de algún dato que se aya enviado al puerto serial de Arduino y convertirlo a los diferentes sistemas de numeración.

```
int valor; //Variable que asignaran a la lectura
  int dato;

  void setup()
  {
    Serial.begin(9600); //Comunicacion con arduino
    Serial.print("Dato  "); //Impresion de titulos
    Serial.print("  BIN  ");
    Serial.print("  OCT  ");
    Serial.print("  HEX");
    Serial.println("  ASCII");
}
void loop()
  {
    if(Serial.available()>0) //Condicion para que solo lea cuando reciba datos del teclado
    {
      valor=Serial.read(); //Asignacion de la lectura serial a una variable
      dato=valor; //Guardar el valor en otra variable
      valor=valor-'0'; //Conversion a decimal del dato en ASCII
      Serial.print(valor,DEC); //Impresion del valor en decimal
      Serial.print("        ");
      Serial.print(valor,BIN); //Impresion del valor en binario
      Serial.print("       ");
      Serial.print(valor,OCT); //Impresion del valor en octal
      Serial.print("      ");
      Serial.print(valor,HEX); //Impresion del valor en hexadecimal
      Serial.print("      ");
      Serial.println(dato); //Impresion del valor en ASCII
    }
  }
```

EJEMPLOS DE APLICACIÓNES

PRACTICA 15 *"CONTROL DE LLENADO DE UN TANQUE"*

Con este programa se tiene el control del llenado de un tanque, leyendo el valor del potenciometro y comparar este valor para encender o apagar el llenado.

```
int sensor;

 void setup()
 {
   Serial.begin(9600);
   pinMode(3,OUTPUT);
 }

 void loop()
 {
   sensor=analogRead(A0);
   delay(200);

   if(sensor>=800) //Esta condicion indica que el tanque se esta llenando
   {

     digitalWrite(3,0); //Detiene el llenado del tanque
   }

   if(sensor<=50) //Condicion que se esta vaciando el tanque
   {
     digitalWrite(3,1);
   }
 }
```

PRACTICA 16 *"DETECCION DE COLOR ROJO – VERDE – AZUL"*

Por medio de entradas analógicos que serán enviadas de un sensor infrarrojo (QRD) se podra detectar si la pieza es de color rojo, verde o azul. Hay que tener en cuenta que que la luz que se encuentre en el ambiente variara en la detección de este.

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

 }

 void loop()
 {
```

```
  sensor=analogRead(A0);
  delay(200);

  if(sensor>=600 && sensor<700) //Rojo
  {
    Serial.println("Color Rojo");
    delay(100);
  }

  if(sensor>=500 && sensor<600) //Azul
  {
    Serial.println("Color Azul");
    delay(200);
  }

  if(sensor>=400 && sensor<500) //verde
  {
    Serial.println("Color Verde");
    delay(200);
  }
}
```

PRACTICA 17 ***"INVERTIR GIRO DE UN MOTOR DC"***

Con la ayuda de un circuito integrado L293d se invertirá el giro de un motor DC, este cambio se hara por medio de un push-boton.

```
int y=0;
  int x=0;
  int z=1;
  int A_1=4;
  int A_2=5;
  int boton=6;

  void setup()
  {

    pinMode(A_1,OUTPUT); //Se define el pin A_1 como salida
    pinMode(A_2,OUTPUT); //Se define el pin A_2 como salida
    pinMode(boton,INPUT); //Se define el pin boton como entrada
  }

  void loop ()
  {
```

```
   if(digitalRead(boton)==1) //Con esta condicion se hara la conmutacion de cambio de
giro del motor
   {
     y=1+x;
     x=y;
   }
   if(x==1)
   {
     x=-x;
   }
   z=z+x;

   if(digitalRead(boton)==0) //Cuando deje de presionar el boton mandara las salidas
correspondientes
   {
     digitalWrite(A1,y);
     digitalWrite(A2,z);

   }
   z=1;
 }
```

PRACTICA 18 ***"CONTE DEL 0 AL 9 CON DISPLAY"***

En esta practica será necesario usar un display de 7 segmentos cátodo. También se vera el uso de la creación de funciones, esta se definirán para que al programa principal se vayan llamando cada uno.

```
int num;
  int a=2;
  int b=3;
  int c=4;
  int d=5;
  int e=6;
  int f=7;
  int g=8;

  void setup()
  {
    Serial.begin(9600);
    for(int x = 2 ; x <= 8 ; x++)
    {
      pinMode(x,OUTPUT);
    }
  }
```

void cero() //En este se crea la primera función que cera la del numero cero como se observa se mandan salidas ya que el display es un juego de leds y nada mas se activan las necesarias para que se observe el numero cero.

```
{
  digitalWrite(g,0);
  digitalWrite(a,1);
  digitalWrite(b,1);
  digitalWrite(c,1);
  digitalWrite(d,1);
  digitalWrite(e,1);
  digitalWrite(f,1);
}

void uno()
{
  digitalWrite(g,0);
  digitalWrite(a,0);
  digitalWrite(d,0);
  digitalWrite(e,0);
  digitalWrite(f,0);
  digitalWrite(b,1);
  digitalWrite(c,1);
}

void dos()
{
  digitalWrite(g,1);
  digitalWrite(a,1);
  digitalWrite(d,1);
  digitalWrite(e,1);
  digitalWrite(f,0);
  digitalWrite(b,1);
  digitalWrite(c,0);
}

void tres()
{
  digitalWrite(g,1);
  digitalWrite(a,1);
  digitalWrite(d,1);
  digitalWrite(e,0);
  digitalWrite(f,0);
  digitalWrite(b,1);
  digitalWrite(c,1);
```

```
}

void cuatro()
{
  digitalWrite(g,1);
  digitalWrite(a,0);
  digitalWrite(d,0);
  digitalWrite(e,0);
  digitalWrite(f,0);
  digitalWrite(b,1);
  digitalWrite(c,1);
}

void cinco()
{
  digitalWrite(g,1);
  digitalWrite(a,1);
  digitalWrite(d,1);
  digitalWrite(e,0);
  digitalWrite(f,1);
  digitalWrite(b,0);
  digitalWrite(c,1);
}

void seis()
{
  digitalWrite(g,1);
  digitalWrite(a,1);
  digitalWrite(d,1);
  digitalWrite(e,1);
  digitalWrite(f,1);
  digitalWrite(b,0);
  digitalWrite(c,1);
}
void siete()
{
  digitalWrite(g,0);
  digitalWrite(a,1);
  digitalWrite(d,0);
  digitalWrite(e,0);
  digitalWrite(f,0);
  digitalWrite(b,1);
  digitalWrite(c,1);
}
```

```
 void ocho()
 {
   digitalWrite(g,1);
   digitalWrite(a,1);
   digitalWrite(d,1);
   digitalWrite(e,1);
   digitalWrite(f,1);
   digitalWrite(b,1);
   digitalWrite(c,1);
 }
 void nueve()
 {
   digitalWrite(g,1);
   digitalWrite(a,1);
   digitalWrite(d,0);
   digitalWrite(e,0);
   digitalWrite(f,1);
   digitalWrite(b,1);
   digitalWrite(c,1);
 }
```

void loop() //En esta parte es donde se mandan a llamar las funciones de los números, para que se logre observar los cambios de los números, se colocaron retardos después de cada llamada, este conteo será cíclico es decir contara del cero al nueve y se reiniciara nuevamente en cero

```
 {
   delay(500);
   cero();
   delay(400);
   uno();
   delay(400);
   dos();
   delay(400);
   tres();
   delay(400);
   cuatro();
   delay(400);
   cinco();
   delay(400);
   seis();
   delay(400);
   siete();
   delay(400);
   ocho();
```

```
  delay(400);
  nueve();
 }
```

PRACTICA 19 ***"CONTROL DE MOTOR A PASOS"***

El motor a pasos a usar es un Unipolar y un C.I. uln2003a. De la misma manera que el anterior programa se crearon funciones ya que para que un motor a pasos pueda observarse el movimiento giratorio es necesario mandarle pulsos y a su vez la configuración de los pulsos va variando para el movimiento.

```
int x;
int vueltas;
int avanzar[]={3,4,5,6};
int encender=8;
void setup()
{
  pinMode(encender,INPUT); //Boton para encender el motor
  pinMode(avanzar[0,1,2,3],OUTPUT); //Arreglo para las activaciones del motor
}
void mov1()
{
  digitalWrite(avanzar[0],1);
  digitalWrite(avanzar[1],0);
  digitalWrite(avanzar[2],0);
  digitalWrite(avanzar[3],0);
}

void mov2()
{
  digitalWrite(avanzar[0],0);
  digitalWrite(avanzar[1],1);
  digitalWrite(avanzar[2],0);
  digitalWrite(avanzar[3],0);
}
void mov3()
{
  digitalWrite(avanzar[0],0);
```

```
  digitalWrite(avanzar[1],0);
  digitalWrite(avanzar[2],1);
  digitalWrite(avanzar[3],0);
}
void mov4()
{
  digitalWrite(avanzar[0],0);
  digitalWrite(avanzar[1],0);
  digitalWrite(avanzar[2],0);
  digitalWrite(avanzar[3],1);
}
void mov5()
{
  digitalWrite(avanzar[0],1);
  digitalWrite(avanzar[1],1);
  digitalWrite(avanzar[2],0);
  digitalWrite(avanzar[3],0);
}
void mov6()
{
  digitalWrite(avanzar[0],0);
  digitalWrite(avanzar[1],1);
  digitalWrite(avanzar[2],1);
  digitalWrite(avanzar[3],0);
}
void mov7()
{
  digitalWrite(avanzar[0],0);
  digitalWrite(avanzar[1],0);
  digitalWrite(avanzar[2],1);
  digitalWrite(avanzar[3],1);
}
void loop()
{
  if(digitalRead(encender)==1)
  {
    x=1;
  }
  if(x==1)
  {
     for(int conta=1; conta<=36; conta++)
        {
          mov1();
          delay(x);
```

```
        mov5();
        delay(x);
        mov2();
        delay(x);
        mov6();
        delay(x);
        mov3();
        delay(x);
        mov7();
        delay(x);
        mov1();
      }
  }
}
```

PRACTICA 20 *"CONTROL DE SERVOMOTOR"*

En practicas anteriores se vieron salidas PWM, en esta practica se volverá a usar ya que para controlar un servomotor es necesario PWM a su vez el control de este será por medio del teclado de la computadora, con la tecla W avanzara y la tecla S retrocederá. Hay que tener en cuenta que los servomotores tienen un ángulo máximo ya que si se sobrepasa este se romperán lo engranes que tiene.

```
#include <Servo.h> //Se carga la libreria de servomotor

  Servo motor; //Se crea la variable que movera el servomotor tomando en cuenta que esta linkeado con la libreria
  int valor;
  int pwm=5;
  void setup()
  {
    Serial.begin(9600);
    //Con estas sentencia se indica la salida PWM, el ancho de pulso min y max del servomotor
    motor.attach(3);
    motor.attach(3,800,2200);
    //Y esta es el valor que se le manda, en este caso es de cero para que el servomotor este en la posicion inicial
    motor.write(0);
  }

  void loop()
  {
    if(Serial.available()>0)
    {
```

```
    valor=Serial.read();
    //Con estas condiciones se seleccionara si avanzara o retrocedera
    if(valor==119) //Avanza
    {
      pwm=pwm+5; //Este es un aumento de 5 en 5 para que se logre observar mas rapido el cambio
      if(pwm>=180) //Esta condicion es para que no sobrepase el angulo maximo del servomotor
      {
        pwm=180;
      }
      motor.write(pwm);
      Serial.println(pwm,DEC);
    }

    if(valor==115) //Retrocede
    {
      pwm=pwm-5;
      if(pwm<=0) //Esta condicion es para que no sobrepase el angulo minimo del servomotor
      {
        pwm=0;
      }
      motor.write(pwm);
      Serial.println(pwm,DEC);
    }

  }
 }
```

REFERENCIA BIBLIOGRAFICA

WWW.ARDUINO.COM