

Laboratorio de Desarrollo de Hardware

Memoria de Prácticas

Arduino

Ángel López Santiago

Contenido

Objetivos	2
¿Qué es Arduino?	2
Hardware usado	2
Estructura general de un programa en Arduino.....	3
Ejercicio 1: Variación sobre el ejemplo blink	3
Ejercicio 2: Control de LED por puerto serie	5
Ejercicio 3: Control de una bombilla mediante un relé	6
Ejercicio 4: Control de un LED mediante interrupciones	6
Ejercicio 5: Control de la posición de un servo mediante un potenciómetro	7
Ejercicio 6: Control posición de un servo desde PC.....	8
Ejercicio 7: Control de la velocidad de un motor mediante potenciómetro	9
Ejercicio 8: Imprimir en LCD datos desde puerto serie	11
Ejercicio 9: Imprimir temperatura por pantalla LCD.....	12
Ejercicio 10: Contador en display de 7 segmentos.....	14
Conclusión.....	18

Objetivos

Nuestro objetivo es Conocer la plataforma arduino, sus características, sus variantes, sus modos de programación, así como conocer una serie de componentes básicos de hardware típicos de aplicaciones de sistemas empotrados.

Para ello realizaremos ejemplos básicos de funcionamiento sobre arduino y desarrollaremos otros ejemplos de uso de arduino manejando diversos componentes hardware.

¿Qué es Arduino?

Arduino es una plataforma de desarrollo de hardware abierta (open Hardware) basada en software y hardware flexibles y fáciles de usar. Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores (sensores de temperatura, humedad, presión, etc.) y puede afectar aquello que le rodea controlando luces, motores y otros tipos de actuadores.

Arduino está basado inicialmente en microcontroladores de 8 bits AVR, aunque con alguna versión basada en ARM de 32 bits.

Entre las ventajas que no ofrece Arduino podemos encontrar un entorno de desarrollo (IDE) fácil de manejar y un amplio conjunto de bibliotecas de manejo de periféricos que nos ofrece la propia plataforma Arduino y la gran comunidad de desarrolladores.

En las prácticas usaremos la plataforma Arduino Uno.

Hardware usado

En las prácticas usaremos componentes habituales en el desarrollo hardware como resistencias, leds, pulsadores, interruptores, cables, etc. Así como:

- **Potenciómetro.** Basado en una resistencia variable. Tiene tres terminales, dos de ellos conectados a GND y VCC respectivamente, de manera que con la resistencia variable la salida del tercer terminal puede cambiarse de forma mecánica.
- **Sensor de temperatura (LM36GZ).** Tiene tres terminales; Dos, de alimentación y el tercero de señal. Varía la tensión en función de la temperatura.
- **Puente H.** Permite intercambiar la polaridad en las señales de salida con señales de control de entrada.
- **Relé.** Funciona como interruptor controlado por señal eléctrica (generalmente el control es de una señal electrónica de pocos voltios, mientras la salida puede ser señal de electricidad).

- **Motor de corriente continua.** Una tensión de continua activa el giro del motor. Gira en dos sentidos según la polaridad de la corriente.
- **Servo.** Básicamente es un motor DC con circuitería de control ya incluida. La función del servomotor es girar hasta colocarse en un ángulo determinado y mantenerse en esa posición mientras se desee. Suelen tener un ángulo de giro de 0 a 180grados. El ángulo de giro en el que se posicionan depende del tamaño del pulso que se envíe por la señal de control.

Estructura general de un programa en Arduino

La estructura general de un programa en Arduino es propia de cualquier sistema empujado ejecutando una aplicación en modo stand-alone.

```
void setup() {
    // put your setup code here, to run once:
}

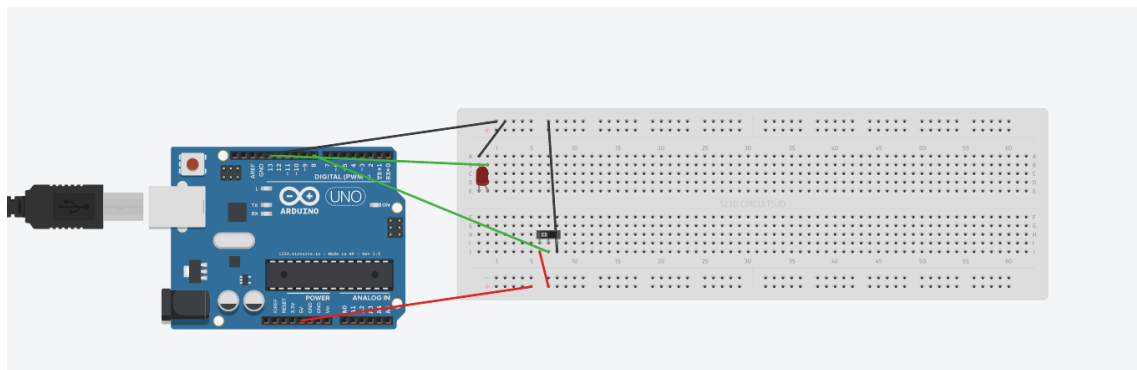
void loop() {
    // put your main code here, to run repeatedly:
}
```

En setup inicializaremos las variables globales y los periféricos. Sólo se ejecuta una vez.

En loop pondremos el cuerpo principal del programa que se repetirá siempre.

Ejercicio 1: Variación sobre el ejemplo blink

En este ejercicio controlaremos el encendido/apagado de un led con un switch/pulsador conectado a VCC y GND.



El código usado para este ejercicio es:

```
// Pin 13 has an LED connected on most Arduino boards.

// give it a name:

int led = 13;

int push = 8;

int pulsado;

// the setup routine runs once when you press reset:

void setup() {

// initialize the digital pin as an output.
pinMode(led, OUTPUT);

// initialize the digital pin as an input.
pinMode(push, INPUT);

}


// the loop routine runs over and over again forever:

void loop() {

pulsado = digitalRead(push);

if(pulsado == 1){

digitalWrite(led, LOW); // turn the LED on (HIGH is the voltage level)

}

else{

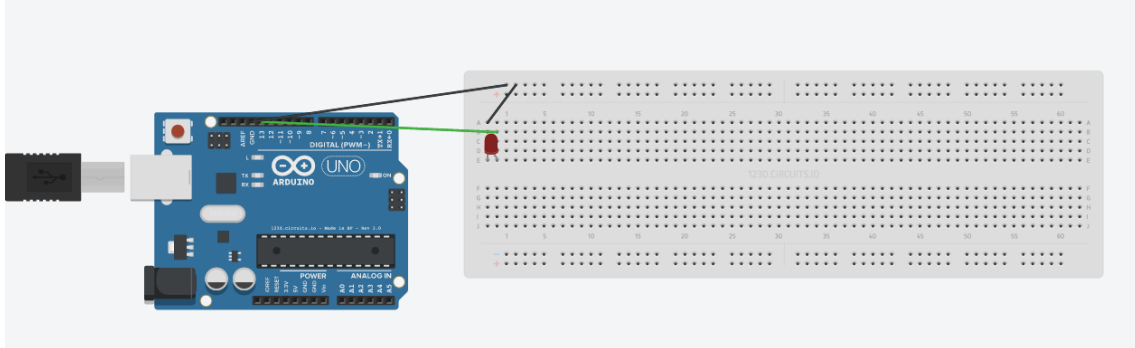
digitalWrite(led, HIGH); // turn the LED off by making the voltage LOW

}

}
```

Ejercicio 2: Control de LED por puerto serie

Este ejercicio es una variación del ejercicio anterior en el que controlaremos el LED haciendo uso del puerto serie mediante un programa en python.



El código usado es:

```
int led = 13;

void setup () {

  pinMode(led, OUTPUT); //LED 13 como salida

  Serial.begin(9600); //Inicializo el puerto serial a 9600 baudios

}

void loop () {

  if (Serial.available()) { //Si está disponible

    char c = Serial.read(); //Guardamos la lectura en una variable char

    if (c == 'H') { //Si es una 'H', enciendo el LED

      digitalWrite(led, HIGH);

    } else if (c == 'L') { //Si es una 'L', apago el LED

      digitalWrite(led, LOW);

    }

  }

}
```

Nuestro programa en Python

```
import serial

arduino = serial.Serial('/dev/ttyACM0', 9600)

print("Starting!")

while True:

    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    if comando == 'H':

        print('LED ENCENDIDO')
    elif comando == 'L':

        print('LED APAGADO')

arduino.close() #Finalizamos la comunicacion
```

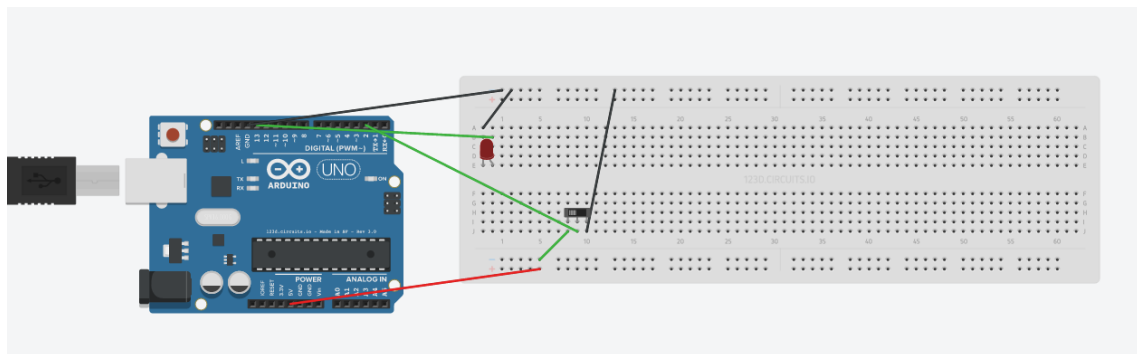
Ejercicio 3: Control de una bombilla mediante un relé

Este ejercicio es una variación del ejercicio anterior en el que controlaremos el relé haciendo uso del puerto serie mediante un programa en Python.

Se hace uso del mismo código arduino y Python que en el ejercicio anterior.

Ejercicio 4: Control de un LED mediante interrupciones

Variante del ejercicio 1 en el que controlaremos el LED mediante interrupciones.



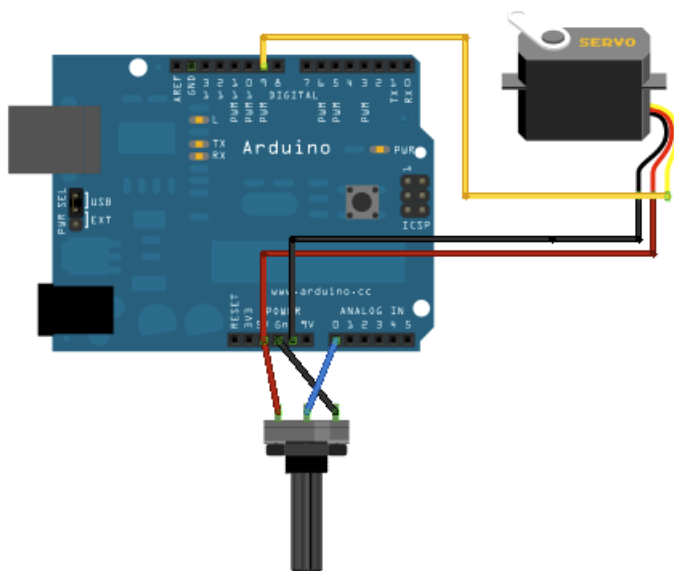
Disponemos de este código arduino:

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}
void loop() {
  digitalWrite(ledPin, state);
}
void blink() {
  state = !state;
}
```

Al pulsar el pulsador detectamos cierto rebote en el LED.

Ejercicio 5: Control de la posición de un servo mediante un potenciómetro

Para este ejercicio haremos uso de la librería servo de arduino.



Disponemos de este código:

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo

int potpin = 0; // analog pin used to connect the potentiometer

int val; // variable to read the value from the analog pin

void setup() {

    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop() {

    val = analogRead(potpin); // reads the value of the
//potentiometer (value between 0 and 1023)

    val = map(val, 0, 1023, 0, 180); // scale it to use it with the //servo
(value between 0 and 180)

    myservo.write(val); // sets the servo position //according
to the scaled value

    delay(15); // waits for the servo to get //there
}
```

Ejercicio 6: Control posición de un servo desde PC

Variación del ejercicio anterior haciendo uso de un programa en Python para controlar la posición del servo. Disponemos de este código en arduino:

```
#include <Servo.h>

int val = 0; //Variable de entrada del Serial

Servo servo; //Creamos un objeto Servo de nombre... servo

void setup()
{
    Serial.begin(9600); //Iniciamos el serial
    servo.attach(3); //Conectamos el servo al pin digital 3
}

void loop()
{
    if(Serial.available() > 0) //Detecta si hay alguna entrada por
//serial
    {
        val = Serial.parseInt();
        if(val != 0)
        {
            servo.write(val); //Mueve el servo a la posición entrada
//(excepto si es 0)
        }
        delay(500);
    }
}
```

Disponemos también de este código en Python:

```
import serial

arduino = serial.Serial('/dev/ttyACM0', 9600)

print("Starting!")

while True:

    angulo = raw_input('Introduce un angulo: ') #Input
    arduino.write(angulo) #Mandar un comando hacia Arduino

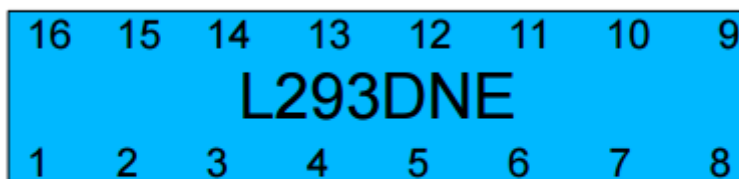
arduino.close() #Finalizamos la comunicacion
```

Ejercicio 7: Control de la velocidad de un motor mediante potenciómetro

En este ejercicio controlaremos la velocidad y el sentido de un motor de corriente continua haciendo uso de un potenciómetro.

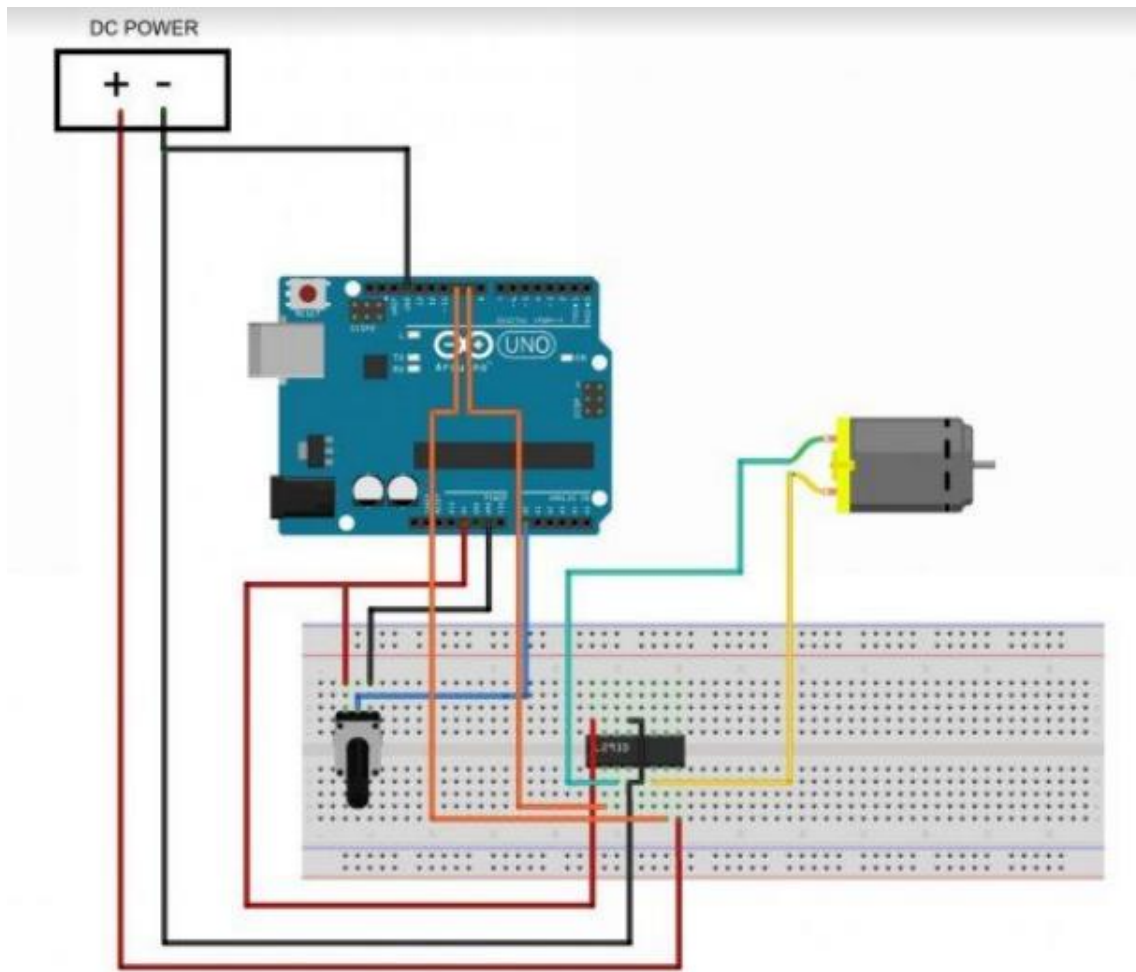
Haremos uso de un circuito de puente H para aportar potencia y cambiar la polaridad del motor.

+5v



+5v pwm1 M1 GND GND M2 pwm2 **+9v**

Circuito puente H



Disponemos del siguiente código arduino:

```
int pin2=9; //Entrada 2 del L293D
int pin7=10; //Entrada 7 del L293D
int pot=A0; //Potenciómetro
int valorpot; //Variable que recoge el valor del potenciómetro
int pwm1; //Variable del PWM 1
int pwm2; //Variable del PWM 2
void setup(){
  //Inicializamos los pins de salida
  pinMode(pin2,OUTPUT);
  pinMode(pin7, OUTPUT);
}
```

```

void loop(){
//Almacenamos el valor del potenciómetro en la variable
valorpot=analogRead(pote);

//Como la entrada analógica del Arduino es de 10 bits, el rango va de 0 a
//1023.

//En cambio, la salidas del Arduino son de 8 bits, quiere decir, rango entre 0
//a 255.

//Por esta razón tenemos que mapear el número de un rango a otro usando este
//código.

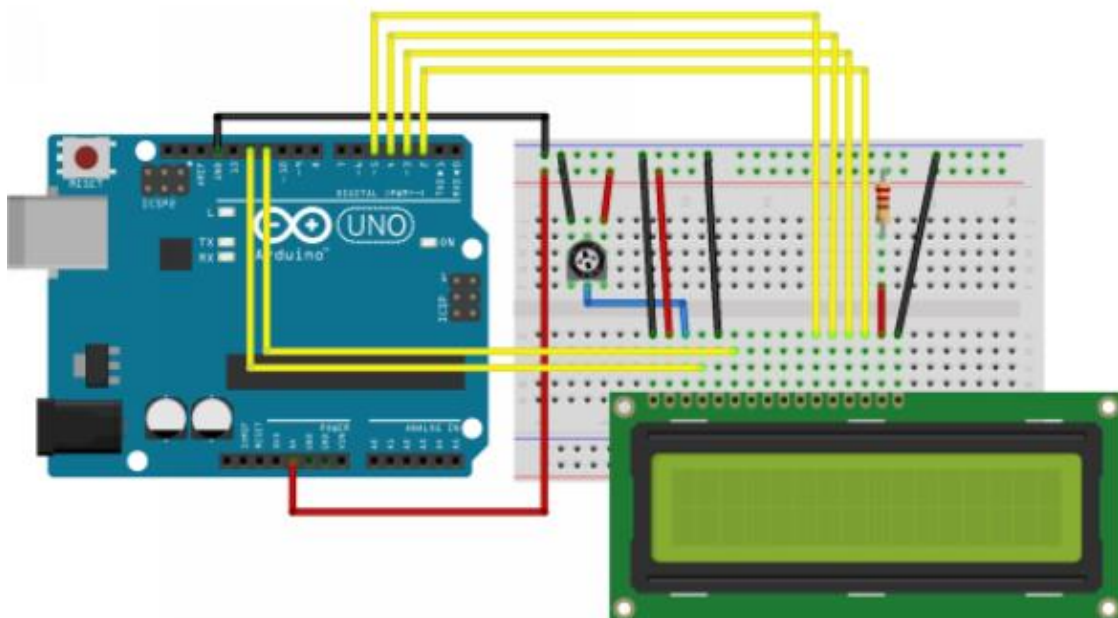
pwm1 = map(valorpot, 0, 1023, 0, 255);
pwm2 = map(valorpot, 0, 1023, 255, 0); //El PWM 2 esta invertido respecto al
//PWM 1

//Sacamos el PWM de las dos salidas usando analogWrite(pin,valor)
analogWrite(pin2,pwm1);
analogWrite(pin7,pwm2);
}

```

Ejercicio 8: Imprimir en LCD datos desde puerto serie

En este ejercicio vamos a imprimir por una pantalla LCD conectada a nuestra placa arduino texto enviado desde nuestro PC a través del puerto serie haciendo uso de un programa en Python.



Usaremos también un potenciómetro para controlar el brillo de la pantalla.

Disponemos de este código arduino usando la librería LCD:

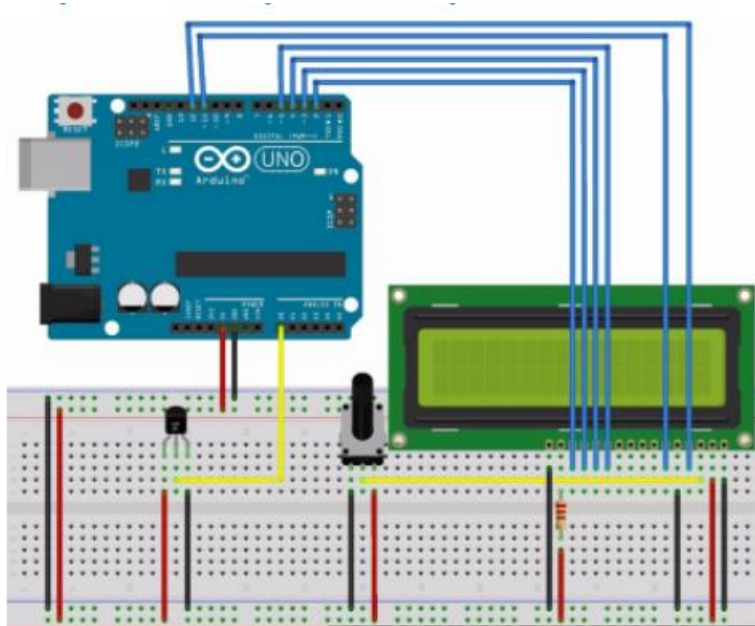
```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup() {
    lcd.begin(16,2);
    Serial.begin(9600);
}
void loop() {
    if(Serial.available()) {
        delay(100);
        lcd.clear();
        while(Serial.available() > 0) {
            lcd.write(Serial.read());
        }
    }
}
```

Disponemos del siguiente código Python:

```
import serial
arduino = serial.Serial('/dev/ttyACM0', 9600)
print("Starting!")
while True:
    comando = raw_input('Introduce palabra: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    print(comando);
arduino.close()
```

Ejercicio 9: Imprimir temperatura por pantalla LCD

Empleando el sensor de temperatura tmp36GZ imprimiremos por la pantalla LCD la temperatura ambiente.



Disponemos del siguiente código:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

int sensorPin = 0;

float valorGradosCent() {
    int dato;
    float resultado;
    dato = analogRead(sensorPin);
    resultado = (5000.0 * dato)/1024;
    resultado = (resultado - 500)/10;
    return resultado;
}

void setup() {
    Serial.begin(9600);
    lcd.begin(16,2);
}
```

```

void loop() {
    lcd.setCursor(0,0);
    int sensorVal = analogRead(sensorPin);
    Serial.print("Valor sensor: ");
    Serial.print(sensorVal);
    float voltaje = (sensorVal/1024.0) * 5.0;
    Serial.print(", Voltios: ");
    Serial.print(voltaje);
    lcd.setCursor(0,1);
    lcd.write("Temperatura: ");
    float temperatura = (voltaje - .5) *100;
    lcd.print(temperatura);
    lcd.setCursor(5,1);
    lcd.write((char)223);
    lcd.setCursor(6,1);
    lcd.write("C");
    delay(2000);
}

```

Ejercicio 10: Contador en display de 7 segmentos

Haremos un contador de 0 a 59 en un display de 7 segmentos.

El código está estructurado de la siguiente forma:

- Función principal (loop()). Basado en un bucle infinito que se ejecuta cada segundo que calculará el valor correcto de cada dígito del display (data0 y data1). Llamará a la función refresh(data0,data1).
- Función refresh (refresh(int data0,int data1)). Se encarga de actualizar los displays 7-seg cada cierto tiempo.
Para activar display1 o display0 se emplean los bits PB0 y PB1 de la placa de expansión (uno activo y el otro desactivo y así alternativamente cada Xms).
Llamará a la función write_data(int dato).
- Función write_data (write_data(int dato)). Se encarga de calcular el valor correcto de representación de un entero en los displays 7-seg y lo escribe en la función mediante write7seg(unsigned char arg).
- Función write7seg (write7seg(unsigned char)). Se encarga de escribir en el display 7-seg el valor de entrada.

Código función principal:

```
int segPins[] = { 0, 1, 2, 3, 4, 5, 6, 7};
int tiempototal= 1000;
int j = 40;
int disp1 =8;
int disp2= 9;
int dat1 = 0;
int dat0 = 0;
int tiempoMax = 60;
void setup() {
    for (int thisseg = 0; thisseg < 8; thisseg++) {
        pinMode(segPins[thisseg], OUTPUT);
    }
    pinMode(disp1, OUTPUT);
    pinMode(disp2, OUTPUT);
}
void loop() {
    int valor;
    while(1){
        for (valor = 0; valor < tiempoMax; valor ++) {
            dat1 = valor / 10;
            dat0 = valor % 10;
            refresh(dat1, dat0);
        }
    }
}
```

Código función refresh:

```
void refresh (int data1, int data0){
    int j=40;
    int tiempo_refresco = tiempototal/(2*j);
    int i;
    for(i=0; i<j;i++){
        delay(tiempo_refresco);
        digitalWrite(disp1, 1);
    }
}
```



```

        digitalWrite(dis2, 0);
        write_data(data1);
        delay(tiempo_refresco);
        digitalWrite(dis1, 0);
        digitalWrite(dis2, 1);
        write_data(data0);
    }
}

```

Código función write_data:

```

void write_data(int arg){
    switch(arg){
        case 0:
            write7seg(0x7e);
            break;
        case 1:
            write7seg(0x30);
            break;
        case 2:
            write7seg(0x6d);
            break;
        case 3:
            write7seg(0x79);
            break;
        case 4:
            write7seg(0x33);
            break;
        case 5:
            write7seg(0x5b);
            break;
        case 6:
            write7seg(0x1f);
            break;
    }
}

```

```

        case 7:
            write7seg(0x70);
            break;
        case 8:
            write7seg(0x7f);
            break;
        case 9:
            write7seg(0x73);
            break;
        default:
    }
}

```

Función write7seg:

```

void write7seg(unsigned char arg){
    unsigned char segmen = 0x01;
    unsigned char display1;
    display1 = arg;
    for(int i=0; i<8;i++) {
        if((display1 & segmen) == 0x00)
            digitalWrite(i, LOW);
        else
            digitalWrite(i, HIGH);
        segmen <<=1;
    }
}

```

Conclusión

Hemos podido comprobar que la plataforma arduino es una plataforma fácil de programar y muy versátil.