

LDH

ARDUINO

Fernando Ruiz-Gollury Herreros de Tejada

Índice

Objetivos	1
Introducción	1
Desarrollo de la Practica	2
1º ejercicio: coger el ejemplo del Blink	2
2º ejercicio: modificar el ejemplo del Blink con un pulsador	4
3º ejercicio: Controlar del Led mediante el puerto serie y un programa en Python	5
4º ejercicio: Control de encendido/Apagado de una bombilla por medio de relé	7
5º ejercicio: apagado de un led por interrupciones de Arduino.	9
6º ejercicio: Servo motor con control por puerto serie.	10
7º ejercicio: Control velocidad de giro y puente H	12
8º ejercicio: Pantalla LCD que imprima la información mandada desde el pc	14
9º ejercicio: Pantalla LCD que imprima la Temperatura del sensor tmp36gz	16
10º ejercicio: Contador 0 a 59 en display 7-segmento cada segundo	18

Plataforma Arduino

Objetivos:

Conocer la plataforma Arduino, con sus características y sus variantes. Aprender el modo de programación de la plataforma Arduino. Conocer los componentes hardware básico que se pueden usar en la plataforma Arduino.

Instalación de IDE de programación de la plataforma Arduino, aprendizaje y manejo del IDE de la plataforma de desarrollo. El siguiente paso realización de ejemplos básicos para ir cogiendo manejo con las herramientas de la práctica. Por último realización de material más avanzado.

Introducción:

Arduino es una plataforma (compañía o dos) de hardware libre y una comunidad tecnológica que diseña y manufactura placas de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programan las placas. Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, tanto para sus componentes de hardware como de software, son libres de licencia, de código abierto que permite libertad de acceso a los mismos³.

El hardware del que se componen son una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields), que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación serial con el computador.

Por otro lado, el software consiste en un entorno de desarrollo (IDE) basado en el entorno de Processing y lenguaje de programación basado en Wiring muy similar a C. El microcontrolador de la placa se programa a través de un computador, haciendo uso de comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

Arduino puede tomar información del entorno a través de sus pines de entrada/salida que son los puertos digitales y analógicos de entrada/salida que conforman la placa. Con la conexión de estos pines se toma la información de toda la placa de sensores y puede afectar aquellos que le rodea controlando luces, motores y otros tipos de actuaciones.

La plataforma de Arduino se inició en el año 2006 como proyecto para los alumnos del instituto italiano IVRE, para abaratar el coste de los microcontroladores que usaban. A lo largo de estos años la plataforma ha ido desarrollando distintos tipos de placas. En 2015 ha habido una división en el equipo de los fundadores creando dos plataformas muy similares pero distintas desde ese momento en el desarrollo hardware y del IDE. Al tener registrada la marca Arduino en Italia el fundador que se separó (www.arduino.org), el resto de fundadores ha creado una marca para su uso fuera de EEUU de nombre GENUINO (www.arduino.cc).

Desarrollo de la Práctica:

Durante el desarrollo de esta práctica con la placa Arduino nos vamos a encontrar con una serie de problemas o ejercicios a llevar a cabo, que se resolverán con el avance natural de la práctica a pesar del incremento en dificultad de los ejercicios.

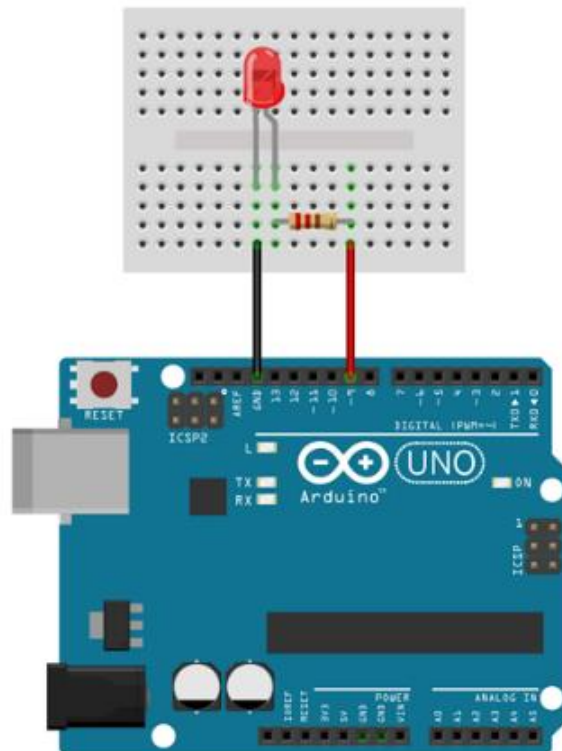
El primer escollo que nos vemos a superar es la instalación del IDE de desarrollo en el sistema operativo Ubuntu (distribución de Linux). Lo realizamos bajándonos de la web (www.arduino.cc) el IDE y con el terminal de Linux hacemos la instalación:

- Se descomprime el fichero.
- Con el terminal entramos en el directorio descomprimido anteriormente.
- Una vez en el directorio ponemos en el terminal: `./install.sh`

1º ejercicio: coger el ejemplo del Blink y hacerlo funcionar con un Led externo a la placa.

Montamos en la protoboard en circuito requerido y lo conectamos a la placa mediante un pin digital (pin 9) que a la hora de programarlo le decimos que es de salida. El programita se hace en el IDE con el que lo subimos a la placa para que se pueda ejecutar la funcionalidad que queremos.

Montaje:



Código Arduino:

```
const int ledPIN = 9;

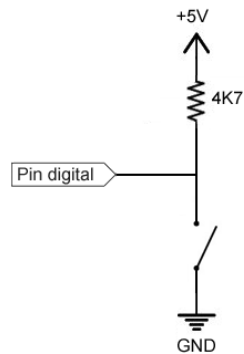
void setup() {
  Serial.begin(9600); //iniciar puerto serie
  pinMode(ledPIN , OUTPUT); //definir pin como salida
}

void loop(){
  digitalWrite(ledPIN , HIGH); // poner el Pin en HIGH
  delay(1000);                // esperar un segundo
  digitalWrite(ledPIN , LOW);  // poner el Pin en LOW
  delay(1000);                // esperar un segundo
}
```

2º ejercicio: Una pequeña modificación al ejemplo del Blink poniendo un pulsador al circuito para controlar el encendido y apagado.

Esta pequeña modificación la haremos con la ayuda de las resistencias de Pull Up, estas resistencias son un mecanismo básico, dentro de la electrónica. La resistencia Pull Up se coloca entre el Pin digital y la tensión de referencia de 5V; fuerza HIGH cuando el pulsador esta abierto y LOW cuando está cerrado.

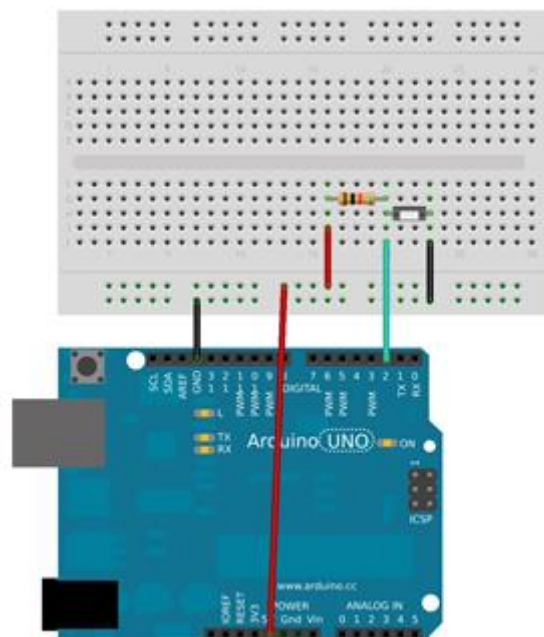
RESISTENCIA PULL UP



Pull Up más a fondo: <http://ovtoaster.com/resistencias-pulldown-y-pullup/>

Montaje:

RESISTENCIA PULL UP



Código Arduino:

```
const int inputPin = 2;

int value = 0;

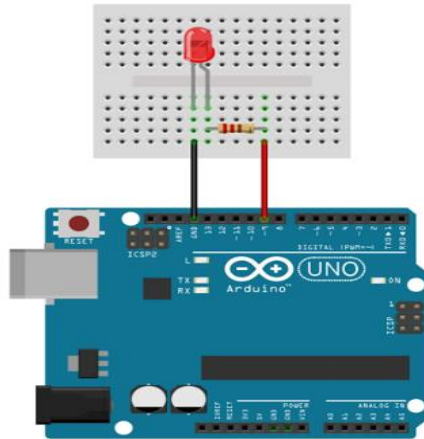
void setup() {
  Serial.begin(9600);
  pinMode(inputPin, INPUT);
}

void loop(){
  value = digitalRead(inputPin); //lectura digital de pin
  //mandar mensaje a puerto serie en función del valor leído
  if (value == HIGH) {
    Serial.println("Encendido");
  }
}
```

3º ejercicio: Controlar el encendido del Led mediante el puerto serie y un programa en Python

Daremos órdenes al Arduino por medio del puerto serie. Mediante un programa en Python le diremos cuando apagar y encender las luces de nuestro circuito.

Montaje:



Código Arduino:

```
const int pinled1= 9;

void setup() {
  Serial.begin(9600);
  pinMode(pinled1, OUTPUT);
}

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(pinled1, HIGH);
    } else if (c == 'L') { //Si es una 'L', apago el LED
      digitalWrite(pinled1, LOW);
    }
  }
}
```


Código Python:

```
import serial

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

print("Starting!")

comando = raw_input('Introduce palabra: ') #Input

arduino.write(comando) #Mandar un comando hacia Arduino

if comando == 'H':
```

4ºejercicio: Control de encendido/Apagado de una bombilla por medio de relé

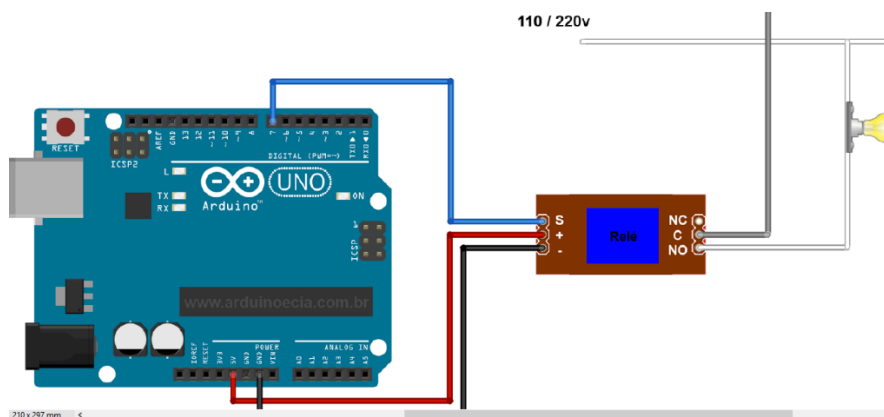
Para poder hacer que Arduino manipule el encendido cosas con voltaje superior a 5v voltios hay que usar un relé que es lo que nos sirve como interruptor.

¿Qué es un relé?

<https://es.wikipedia.org/wiki/Rel%C3%A9>

Ejemplo del funcionamiento del relé: <https://www.youtube.com/watch?v=QjszJEncw8>

Montaje:



Código Arduino

```
const int pin = 7;

void setup() {
  Serial.begin(9600); //iniciar puerto serie
  pinMode(pin, OUTPUT); //definir pin como salida
}

void loop(){
  if (Serial.available()){
    char c= Serial.read();
    if (c == 'H') { //Si es una 'H', enciendo el LED
      digitalWrite(pin, HIGH);
    } else if (c == 'L') { //Si es una 'L', apago el LED
```

El control del puerto serie se puede hacer desde el IDE de Arduino, o como en el ejemplo anterior desde el pc mediante el código Python del ejercicio anterior.

5º ejercicio: apagado de un led por interrupciones de Arduino.

Vamos a utilizar las interrupciones de Arduino para manipular un led con la ayuda de un pulsador. Veremos que el pulsador presenta un problema de rebote en funcionamiento con las interrupciones, por el tiempo de respuesta del pulsador.

Código Arduino

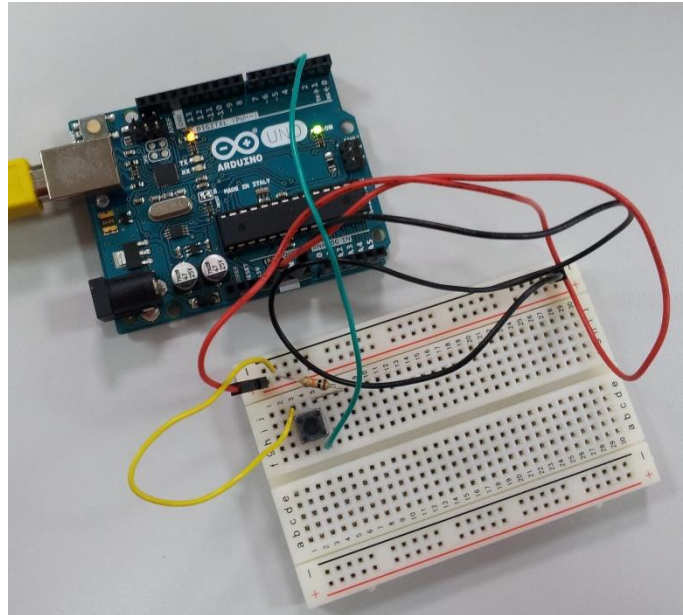
```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

Montaje:

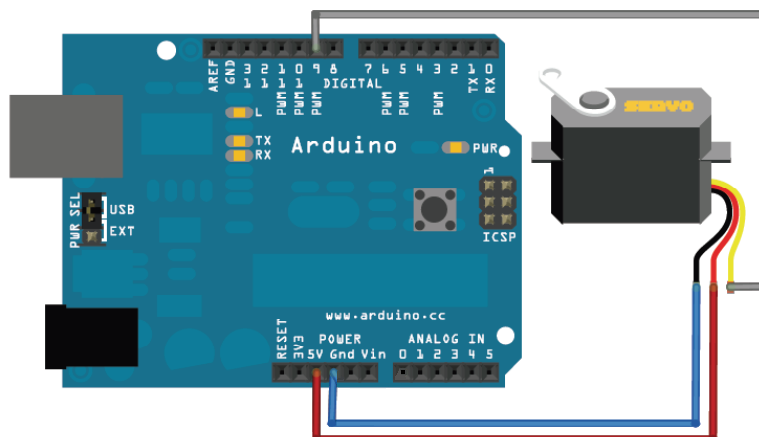


6º ejercicio: Servo motor con control por puerto serie.

Para cambiar las posiciones del servo lo haremos mediante el puerto serie desde el pc con un programa en python.

¿Qué es un servo ? <https://es.wikipedia.org/wiki/Servomotor>

Montaje:



Código Arduino

```
#include <Servo.h>

Servo servo;

int potenciometro = 0;

int aux;

void setup() {
    Serial.begin(9600);
    servo.attach(9);
}

void loop() {
    if (Serial.available()) {
        aux = Serial.parseInt();
        Serial.print (aux);
        servo.write(aux);
    }
}
```

Código Python

```
import serial

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

print('Comenzando')

while True:
    comando = raw_input('Introduce un comando: ')
    arduino.write(comando)
    print "GIRADO: ", comando

arduino.close()
```

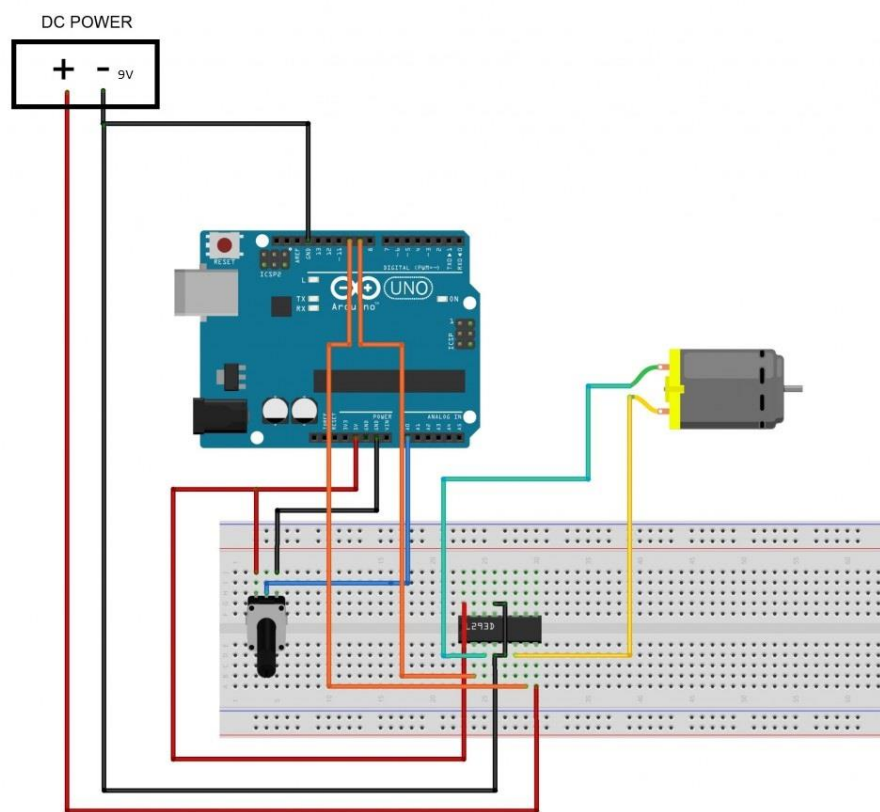
7º ejercicio: Control velocidad de giro y puente H

Al motor de continua por medio del puente H vamos hacer que gire en ambos sentidos y dejarlo parado, todo esto ayudándonos del potenciómetro. Según la posición del potenciómetro estará parado o girando en un sentido u otro, aparte de modular la velocidad según dicha posición.

¿Qué es un potenciómetro? <https://es.wikipedia.org/wiki/Potenci%C3%B3metro>

¿Qué es un puente H? [https://es.wikipedia.org/wiki/Puente_H_\(electr%C3%B3nica\)](https://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica))

Montaje:



Código Arduino

```
int pin2 = 9;
int pin7 = 10;
int pot = A0;
int valorpot;
int pwm1;
int pwm2;

void setup() {
  // put your setup code here, to run once:
  pinMode (pin2, OUTPUT);
  pinMode (pin7, OUTPUT);
}

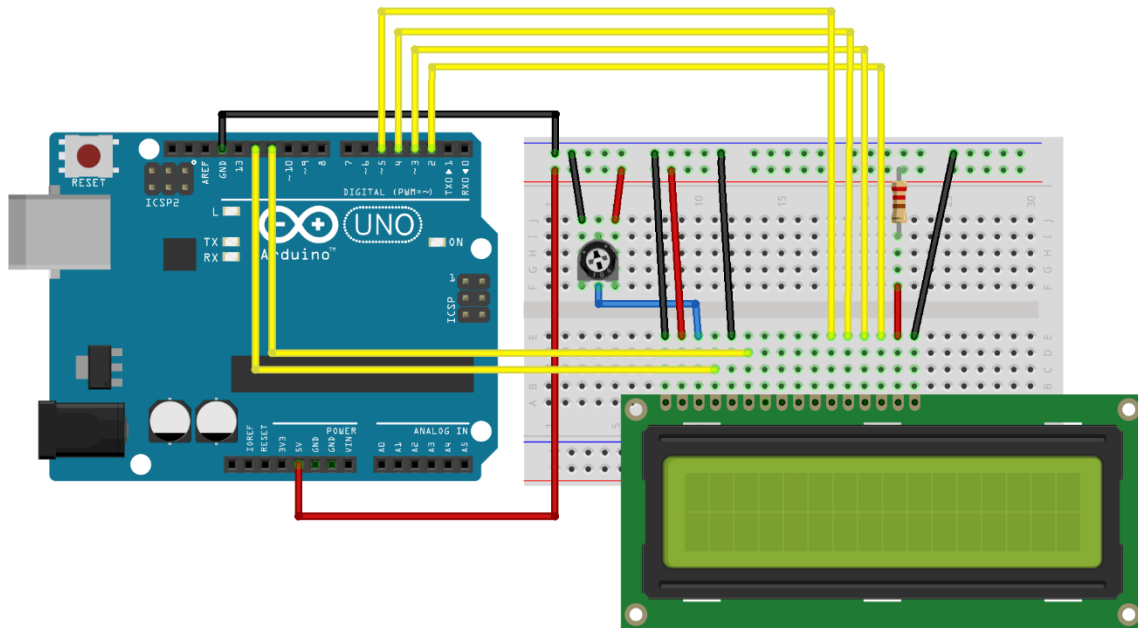
void loop() {
  // put your main code here, to run repeatedly:
  valorpot = analogRead (pot);
  pwm1 = map (valorpot, 0, 1023, 0, 255);
  pwm2 = map (valorpot, 0, 1023, 255, 0);

  analogWrite (pin2,pwm1);
  analogWrite (pin7, pwm2);
}
```

8º ejercicio: Pantalla LCD que imprima la información mandada desde el pc

Por medio del puerto serie de Arduino y de un programa en Python lo que lo que haremos es que nos muestre en la pantalla LCD lo que nosotros queramos.

Montaje:



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()
```


Código Arduino

```
#include <LiquidCrystal.h>

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

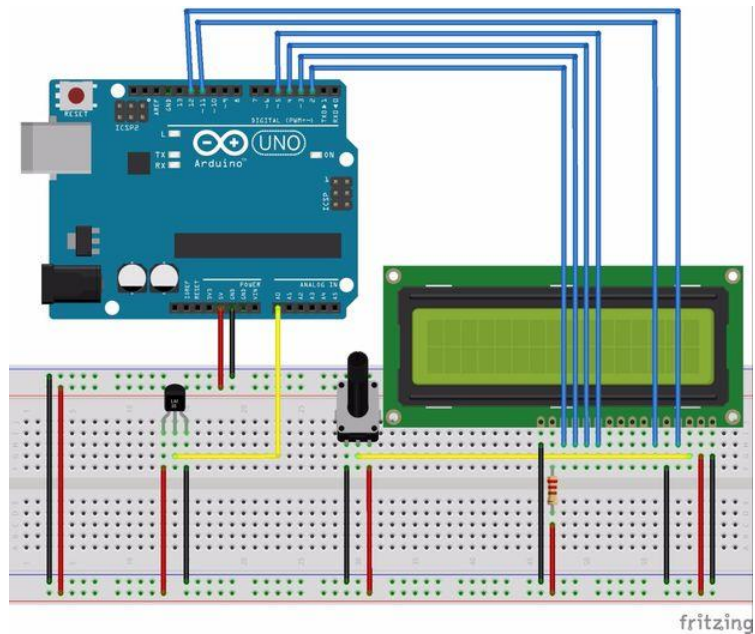
void setup() {
  lcd.begin(16,2);
  Serial.begin(9600);
}

void loop() {
  if(Serial.available()) {
    delay(100);
    lcd.clear();
    while(Serial.available() > 0) {
      lcd.write(Serial.read());
    }
  }
}
```

9º ejercicio: Pantalla LCD que imprima la Temperatura del sensor tmp36gz

En este ejercicio lo que queremos hacer es un termómetro que nos diga la temperatura ambiental allí donde se coloque.

Montaje:



Código Arduino

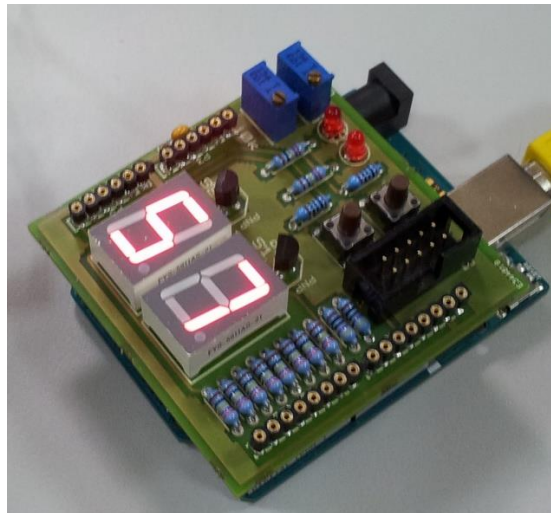
```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
int sensorPin = 0;
float valorGradosCent() {
  int dato;
  float resultado;
  dato = analogRead(sensorPin);
  resultado = (5000.0 * dato)/1024;
  resultado = (resultado - 500)/10;
  return resultado;
}
```

```
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(0,0);  
  lcd.write("Temperatura: ");  
  float temperatura = (voltaje - .5) *100;  
  lcd.setCursor(0,1);  
  lcd.print(temperatura);  
  lcd.setCursor(5,1);  
  lcd.write((char)223);  
  lcd.setCursor(6,1);  
  lcd.write("C");  
  delay(2000);  
}
```

10º ejercicio: Contador 0 a 59 en display 7-segmento cada segundo

Con un Shield específico del laboratorio facilitado por el profesor con dos display 7-segmentos es como vamos a hacer nuestro contador de 00 a 59. Una vez Hecho el contador el profesor me pido que empezara en 50 en vez de en 0, así como manipular su velocidad; estos cambios supuso tocar levemente el código.

Montaje:



Código Arduino

```
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 = 80;
```

```
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;  
    }  
}
```

```

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

void setup() {
    for (int thisseg = 0; thisseg < 8; thisseg++) {
        pinMode(segPins[thisseg], OUTPUT);
    }
    pinMode(dis1, OUTPUT);
    pinMode(dis2, OUTPUT);
}

void loop() {
    int valor;
    while(1){

        for (valor = 50; valor < tiempoMax; valor ++) {
            dat1 = valor / 10;
            dat0 = valor % 10;
            refresh(dat1, dat0);
        }
    }
}

```