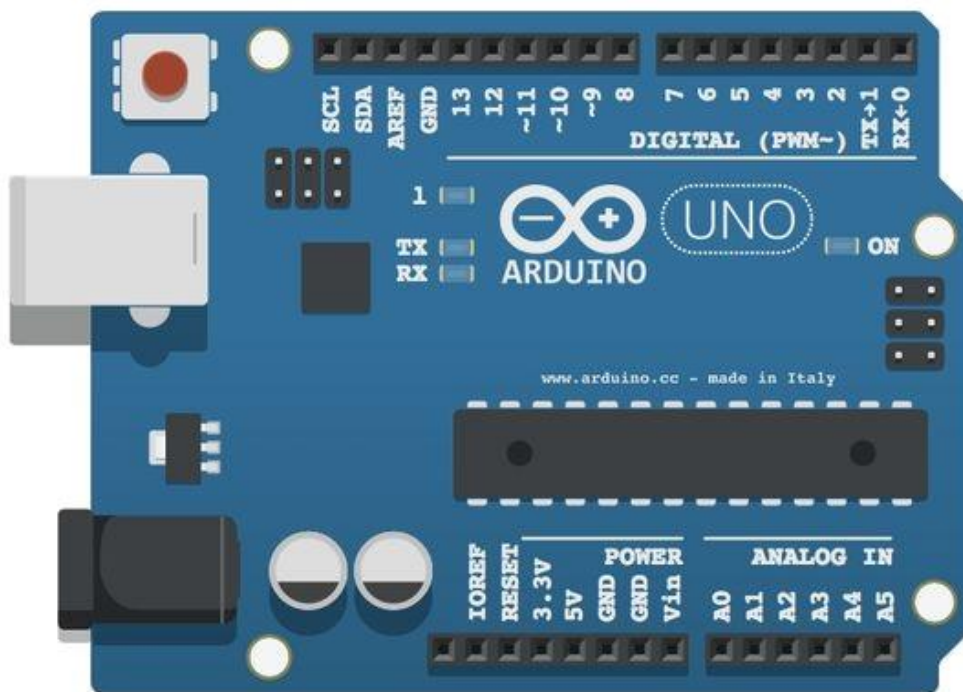


MEMORIA DE PRÁCTICA DE ARDUINO



Nombre: Francisco Núñez Cid

Correo: francisco.nunez.cid@gmail.com

Curso: 2016 – 2017

ÍNDICE

• Introducción	3
• Objetivos	3
• Fundamentos teóricos	
○ ¿Qué es arduino?	3
○ Arquitectura de la placa	3
○ Programación	4
○ Utilización del software	5
○ Hardware utilizado	5
• Ejercicios	6
• Conclusión	31
• Glosario	32
• Bibliografía	34

Introducción

Durante el desarrollo de todas las prácticas de laboratorio del primer bloque, de la parte de **Arduino**, he ido tomando una serie de notas de lo que he ido realizando, los problemas que me he ido encontrado a la hora de hacer los diferentes ejercicios, las soluciones de cada uno de ellos y los diferentes resultados que voy concluyendo.

Objetivos

Los principales propósitos de las prácticas de Arduino son los siguientes:

- Conocer las características de una placa de entrada/salida Arduino y utilizarla en comunicación con un PC.
- Conocer una serie de componentes básicos de hardware típico de aplicaciones de sistemas empotrados.
- Instalar y configurar el software suministrado por el fabricante, que posibilita la gestión de los distintos recursos de la tarjeta.
- Conocer la estructura de un programa en Arduino, su lenguaje de programación y el manejo del puerto serie a través de un programa en python.
- Conocer mecanismos de activación de señales externas de la placa. Obtener habilidades en el desarrollo, la carga y ejecución de programas, así como en la verificación del funcionamiento de algunos ejercicios.

Fundamentos teóricos

- **¿Qué es Arduino?**

Arduino es una plataforma de hardware libre, basada en un microcontrolador, principalmente Atmel AVR, montado en una PCB con los elementos esenciales para su funcionamiento, y pensada para proyectos multidisciplinarios.

- **Arquitectura de la placa**

La placa Arduino UNO se muestra en la siguiente imagen marcando los elementos más usados.

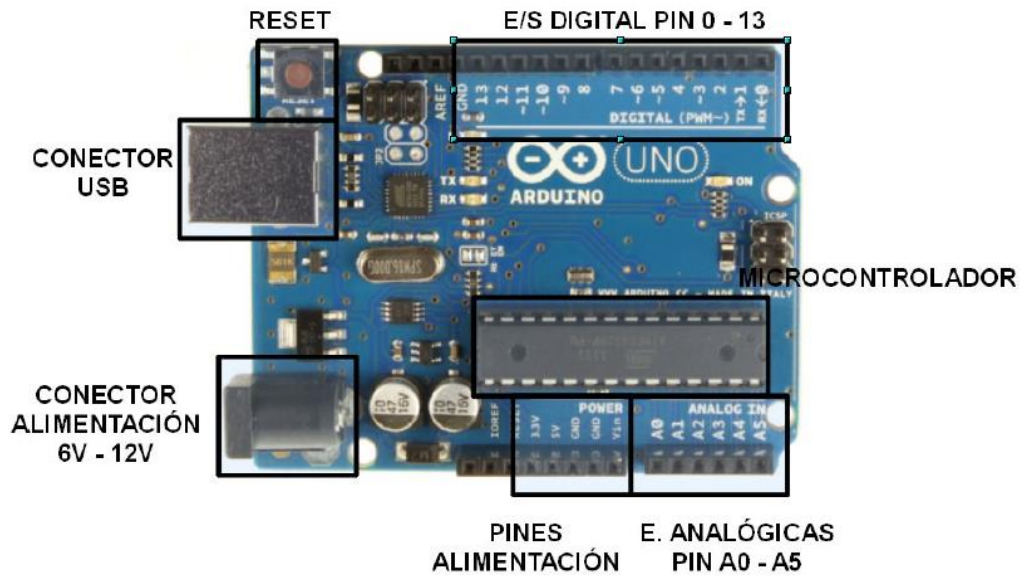


Imagen 1

Esta placa posee un microcontrolador de la marca Atmel y toda la circuitería de soporte que incluye, *reguladores de tensión*, un puerto USB conectado a un módulo adaptador USB-Serie, que permite programar el microcontrolador desde cualquier PC de manera cómoda y también hacer pruebas de comunicación con el propio chip.

Dispone de 14 pines que pueden configurarse como entrada o salida y a los que puede conectarse cualquier dispositivo que sea capaz de transmitir o recibir señales digitales de entre 0 y 5 V.

También dispone de entradas y salidas analógicas. Mediante las entradas analógicas podemos obtener datos de sensores en forma de variaciones continuas de un voltaje. Las salidas analógicas suelen utilizarse para enviar señales de control en forma de señales PWM.

- **Programación**

Un programa en Arduino consta de 3 partes:

- 1. Declaración de variables.**

- 2. Función de inicialización Setup ().**

Se ejecuta una única vez y es empleado para configurar un determinado pin como entrada o salida, configurar la comunicación serie, etc.

- 3. Función Loop ().**

Aquí se encuentra el programa que controlará la respuesta de la placa del Arduino, y esto lo hará infinitas veces hasta que se desconecte la alimentación de la placa.

Un ejemplo de la estructura sería:

```
// Declaración de Variables
int x ;

void setup () {
//Código inicialización microcontrolador,
//velocidad serial, pines de I/O....
}

void loop () {
// Código del programa
}
```

- **Utilización del Software**

1. Descarga e instalación de Arduino (<https://www.arduino.cc/en/Main/Software>)
2. Conexión de la placa Arduino al PC a través del cable USB.
3. Seleccionamos la placa que estamos utilizando. Herramientas → Placa.
4. Seleccionamos el puerto al que tenemos conectada la placa. Herramientas → Puerto.

Los pasos 2 – 4 se hacen siempre cada vez que se inicia el software de arduino.

- **Hardware utilizado**

1. Placa Arduino.
2. Cable USB.
3. Resistencias.
4. Diodos led.
5. Cables para realizar las conexiones.
6. Regleta o protoboard.
7. Potenciómetro.
8. Pulsadores.
9. Relé.
10. Servomotor.
11. Motor c. continua.
12. Sensor de temperatura.
13. Pantalla LCD.
14. Puente H.

Ejercicios

1) Ejecutar programas ejemplos de funcionamiento de arduino: Blink.

En este primer ejercicio hay que hacer que un led se encienda y se apague según unos intervalos de tiempo definidos en el código.

Se puede hacer de dos formas:

- Primera opción: la placa utilizada es la Arduino Uno, que tiene un led incorporado directamente en la placa en el PIN digital 13, por lo que no sería necesario utilizar un led para probar este ejemplo. Simplemente cargar el programa, ejecutar y debería de funcionar, encenderse (1 segundo) y apagarse (1 segundo), esto se realiza continuamente.

Imagen del circuito de la primera opción (con el led encendido):

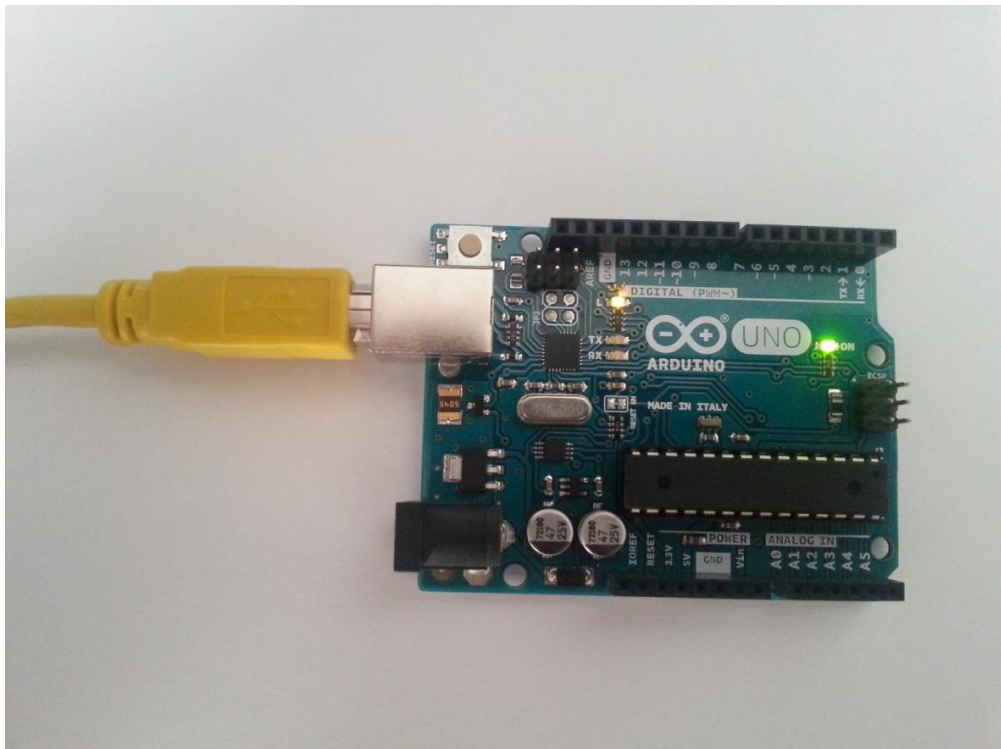


Imagen 2

- Segunda opción: utilizar un led (pata larga al PIN digital 13 y pata corta al GND) y una resistencia (para no fundir el led). Se carga el programa, se ejecuta y el led debería de encenderse (1 segundo) y apagarse (1 segundo), esto se realiza continuamente.

Imagen del circuito de la segunda opción:

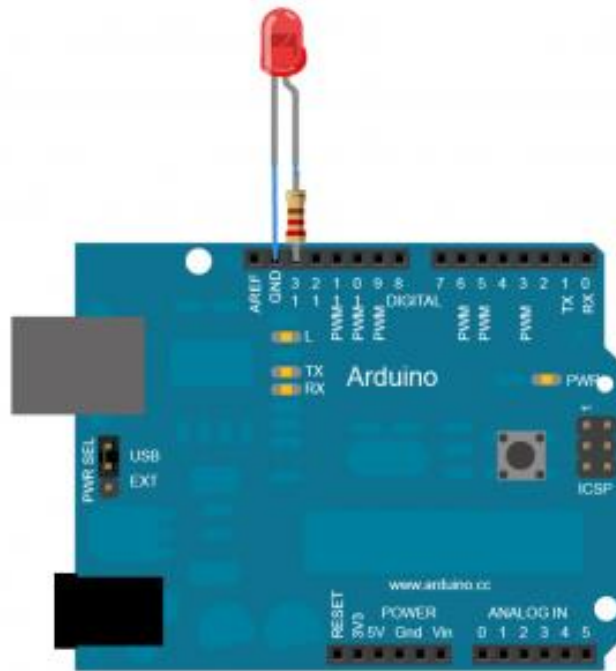


Imagen 3

El código a realizar en Arduino es el siguiente:

```
void setup() {  
  pinMode(13, OUTPUT); // inicializar el pin digital 13 como  
                        una salida.  
}  
  
// la function loop se ejecuta siempre.  
void loop() {  
  digitalWrite(13, HIGH); // Enciende el Led (HIGH)  
  delay(1000);           // Espera 1 segundo  
  digitalWrite(13, LOW);  // Apaga el Led (LOW)  
  delay(1000);           // Espera 1 segundo  
}
```

Nota: Este primer ejercicio ha sido una primera toma de contacto con el entorno de programación y con la placa. No he tenido ningún problema porque ya había utilizado Arduino antes en otra asignatura.

2) Controlar el encendido y apagado de un led con un pulsador conectado a VCC (5V) y GND.

En este segundo ejercicio hay que hacer que un led se encienda cuando se pulsa un pulsador y se apaga cuando se suelta.

Este ejercicio se podía haber realizado como el anterior de dos formas, pero yo lo he hecho de la siguiente manera:

- Primero, como la placa tiene integrado un led en el pin digital 13, pues vamos a usar ese y no tenemos que utilizar ningún led.
- Después, he colocado un pulsador que está conectado al PIN digital 11 y a GND.
- Por último, cargar el programa, ejecutar y debería de funcionar, al pulsar el pulsador se enciende el led y al soltarlo se debe de apagar.

Imagen del circuito realizado en prácticas.

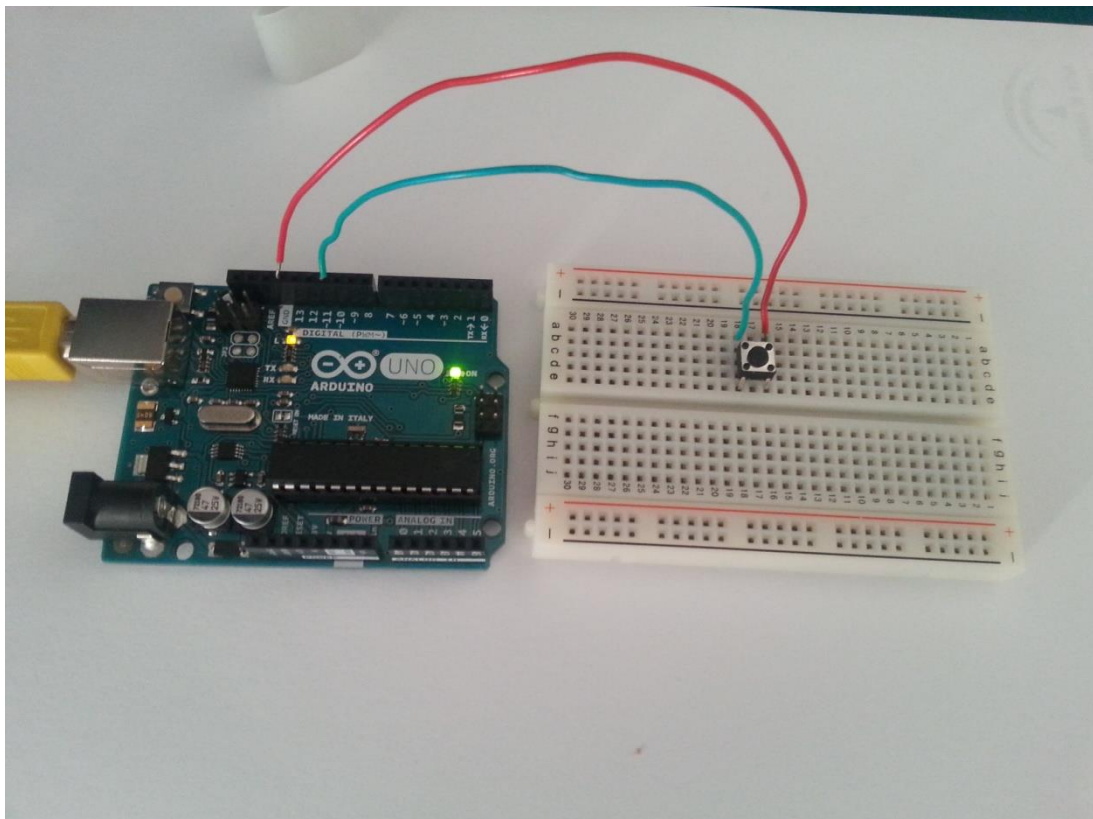


Imagen 4

El código a realizar es el siguiente:

```
int boton = 11; // número del pin para el botón
int ledPin = 13; // número del pin del LED
int estadoBoton = 0; // estado del botón (0 ó LOW es
                    //apagado y 1 ó HIGH es encendido)

void setup() {
  pinMode(ledPin, OUTPUT); //Se identifica el pin 13 como
                          //salida
  pinMode(boton, INPUT); //Se identifica el pin 11 como
                          //entrada
}

void loop(){
  // Leemos si el botón en el pin 11 está abierto o cerrado
  estadoBoton = digitalRead(boton);

  // Si está siendo pulsado es HIGH
  if (estadoBoton == HIGH) {
    digitalWrite(ledPin, HIGH); // Y el LED se enciende
  }
  else {
    digitalWrite(ledPin, LOW); // Si no es así, se apaga
  }
}
```

Nota: Este segundo ejercicio ha sido muy parecido al primero, con la diferencia, que ha visto que añadirle un pulsador para encender o apagar el led. No he tenido ningún problema en este ejercicio, porque ya lo he realizado anteriormente.

- 3) Controlar el encendido y apagado del LED desde el PC vía puerto serie. Hacer un programa en Python que envíe comandos de encendido y apagado a través del puerto serie.

En este tercer ejercicio hay que hacer que a través del puerto serie encendamos un led (pulsando H) o se apague (pulsando L) con un programa Python.

Yo he resuelto este ejercicio de la siguiente manera:

- El montaje del circuito valdría el del ejercicio 1, así que no voy a volver a explicarlo (ver la imagen 2 o 3).

- Primero, voy a escribir el código que se va a cargar en la placa Arduino. Declaro la variable “led” como entero (int), a la que le asignaré el valor 13, que es el pin donde está conectado el led. Tras esto, inicializaré la comunicación y, en el “loop()”, que es la función principal, se irá leyendo constantemente los valores que nos estén llegando desde Python.

El código sería:

```
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(); //Guardo la lectura
                                //en una variable c
        if (c == 'H') { //Si es 'H' enciendo el LED
            digitalWrite(led, HIGH);
        } else if (c == 'L') { //Si es 'L' apago el
                                //LED
            digitalWrite(led, LOW);
        }
    }
}
```

- Después, voy a programar en Python el código que permitirá mandarle comandos a la placa Arduino, que lo he llamado **“Prueba.py”**, pero antes de escribir el código hay que instalar la librería **“python-serial”** en una ventana de símbolos para que no nos de error a la hora de ejecutarlo.

sudo apt-get install python-serial

Este es el código de la clase **Prueba.py**:

```
import serial
arduino = serial.Serial('/dev/ttyACM0', 9600)
print("Starting!")

while True:
    comando = raw_input('Introduce un comando: ')
    arduino.write(comando)
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')
    arduino.close()
```

El código es muy fácil, primero se conecta al puerto de la placa Arduino, y después mientras sea True, va leyendo los valores que se le van introduciendo al puerto serie, si es H (se enciende) y si es L (se apaga).

- Luego, en la ventana de comandos ejecutamos la instrucción: **cd Escritorio** que es donde tengo el archivo **Prueba.py**.
- Por último, ejecuto la instrucción **python Prueba.py** para ejecutar el programa y debería de funcionar. Al escribir "H" en la ventana de comandos el led se debería de encender y si pulsamos "L" el led se debería de apagar.

Nota: En este tercer ejercicio la parte de hardware no he tenido ningún problema porque era como la del ejercicio 1, pero donde he tenido algunos problemas ha sido en la parte de software, porque nunca había programado en Python y al principio no he entendido algunas cosas que había que hacer en este ejercicio, pero al final me ha salido buscando información en internet.

- 4) Controlar el encendido y apagado de una bombilla mediante un relé. Hacer un programa en Python que envíe comandos de encendido y apagado a través del puerto serie.

En este cuarto ejercicio hay que hacer lo mismo que en el ejercicio anterior, pero en vez de un led que se encienda una bombilla y que sea mediante un relé.

Yo he resuelto este ejercicio de la siguiente manera:

- Primero he montado el circuito. He comenzado por el relé, conectando la patilla Vcc (5V) a positivo de la regleta, la patilla GND a negativo de la regleta y la patilla IN1 (hace seleccionar el relé de la izquierda) al pin digital 13.

Imagen de un relé para distinguir las conexiones.

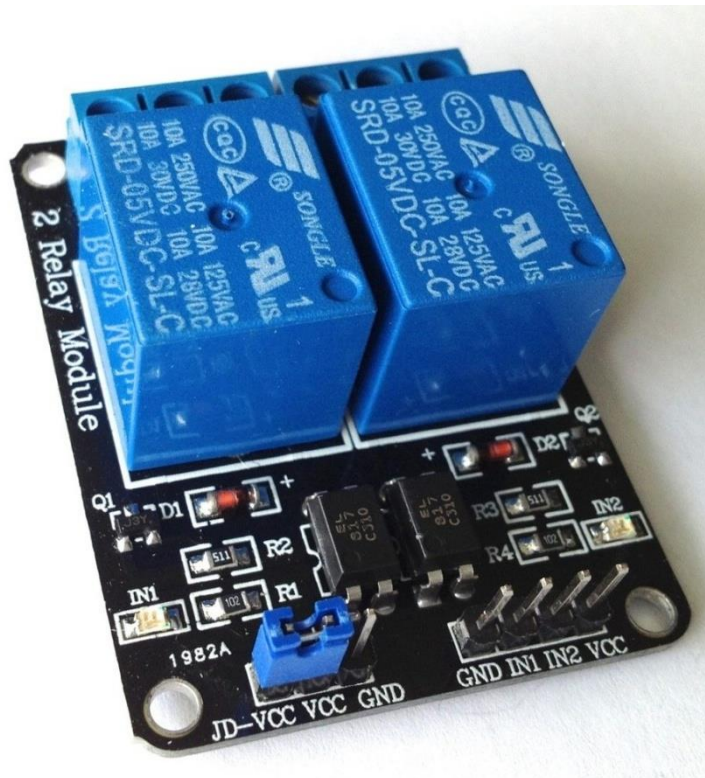


Imagen 5

- Después, de la placa arduino he sacado dos cables, el de GND a negativo de la regleta y Vcc a positivo de la regleta. También he conectado la bombilla al relé por la parte de atrás que tiene 3 orificios. El cable de la bombilla tiene dos partes una de ella siempre va al orificio central del relé y el otro cable al orificio de la derecha o izquierda.
- Por último, la parte de software se hace igual que el ejercicio anterior, hay que seguir los mismos pasos (los códigos de Arduino y de Python valen los mismos que el ejercicio anterior, lo único que se ha cambiado ha sido el montaje del circuito). Al pulsar “H” se debe de encender la bombilla y el relé tiene que hacer un click y al pulsar luego “L” se debe de apagar la bombilla y debe hacer otro click el relé.

Imagen del circuito completo realizado en clase.

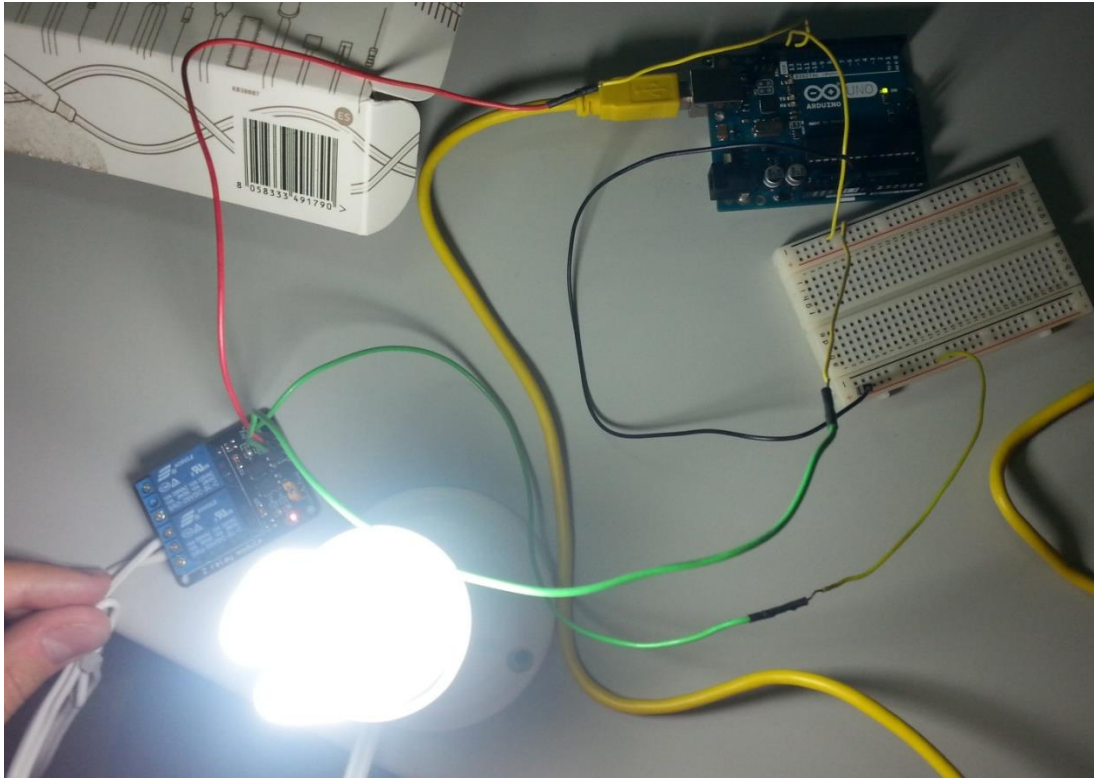


Imagen 6

Nota: En este cuarto ejercicio no he tenido ningún problema a la hora de hacer la parte de hardware y la parte de software ha sido la misma que en el anterior ejercicio, que ahí es donde me dio problemas (la programación en Python), pero ya no ha sido problema para este ejercicio.

- 5) Encendido y apagado de un led mediante un pulsador empleando las interrupciones en arduino. Utilizar un pulsador en configuración de pull-up o pull-down para activar una interrupción en alguno de los pines de interrupciones.

En este quinto ejercicio hay que hacer como el ejercicio 2, pero empleando interrupciones en arduino.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, voy a hacer la parte de hardware. Como la placa tiene integrado un led en el pin digital 13, pues vamos a usar ese y no tenemos que utilizar ningún otro led.
- Después, he colocado un pulsador que está conectado al PIN digital 11 y a GND.

Imagen del circuito realizado en prácticas.

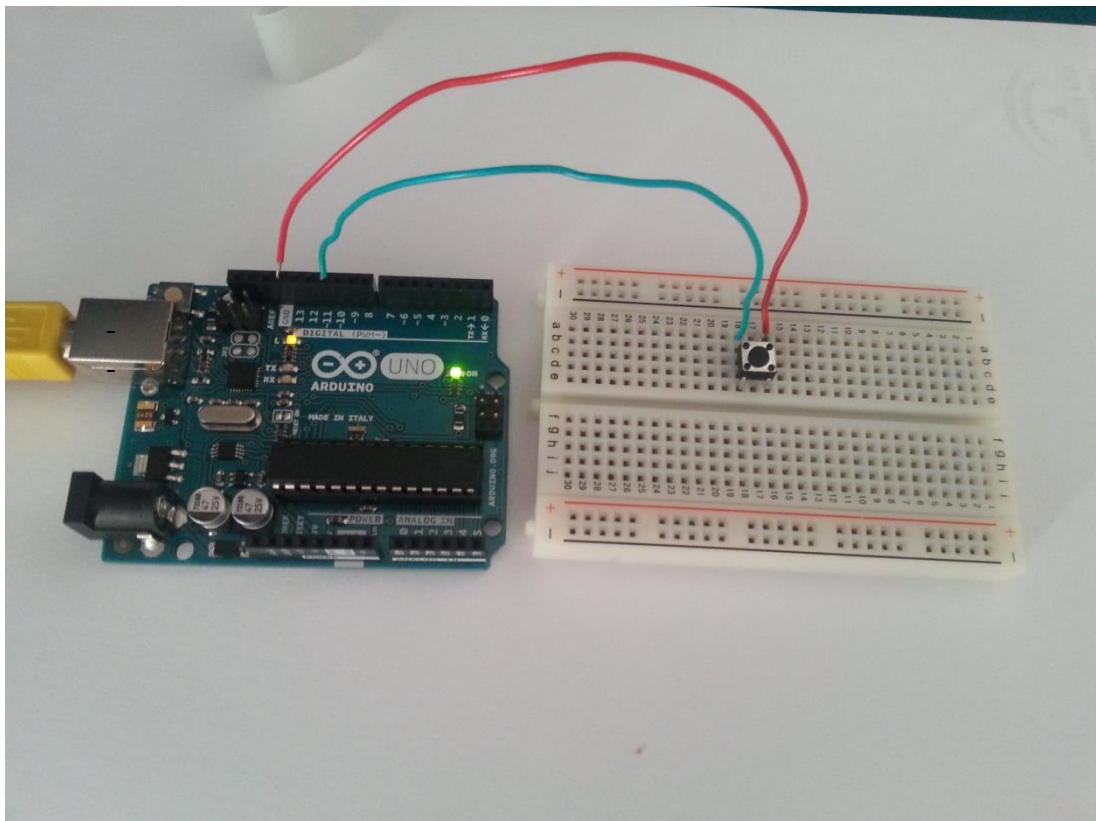


Imagen 7

- Ahora, hago la parte de software. Primero, creo unas constantes de tipo byte que son para el led (pin digital 13), para la interrupción (pin digital 11) y el estado se pone a bajo (0).
- Tras esto, en la función “loop()”, que es la función principal, se pone el “ledPin” como una salida y el “interruptPin” como una entrada de pull-up (que lo que hace es elevar la tensión de salida de un circuito lógico, a la tensión que, por lo general mediante un divisor de tensión, se pone a la entrada de un amplificador con el fin de desplazar su punto de trabajo, ver imagen 8) y después la interrupción se pone a **FALLING**, para que no se produzcan rebotes.

Estado Pull-Up:

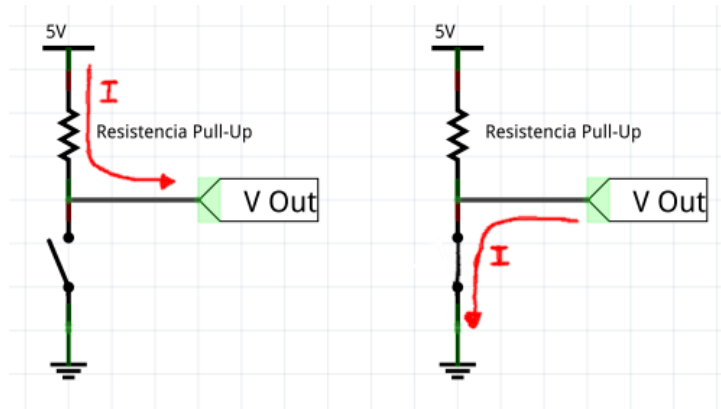


Imagen 8

El código es el siguiente:

```
const byte ledPin = 13;
const byte interruptPin = 11;
volatile byte state = LOW;

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

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

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

Nota: En este quinto ejercicio no he tenido problemas a la hora de hacer la parte hardware, porque era como el ejercicio 2. Donde he tenido un poco de problemas ha sido en la parte de software, en el código del ejercicio, porque antes tenía puesto **CHANGE** y esto hacía que me provocara rebotes el pulsador y al ponerle **FALLING** se arregló, aunque alguna vez se pulsaba y seguía sin apagarse el led. He estado buscando información por internet para poder hacer el código de arduino, pero al final he entendido lo que hace el programa.

- 6) Control de la posición de un servo motor con un potenciómetro. Controlar el ángulo de posicionamiento del servo desde el PC, enviando el valor de ángulo (desde 0 a 180 grados) a través del puerto serie.

En este sexto ejercicio lo que tenemos que hacer es que a través del puerto serie mandarle un ángulo a la placa para que el servo gire dependiendo del valor indicado.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, he realizado la parte de hardware. He colocado el potenciómetro, en la que tiene 3 patillas (ver imagen 9, para ver las diferentes patillas y los números indicados). La patilla 1 la he conectado a 5V, la patilla 2 a negativo y la patilla 3 al pin analógico A0.

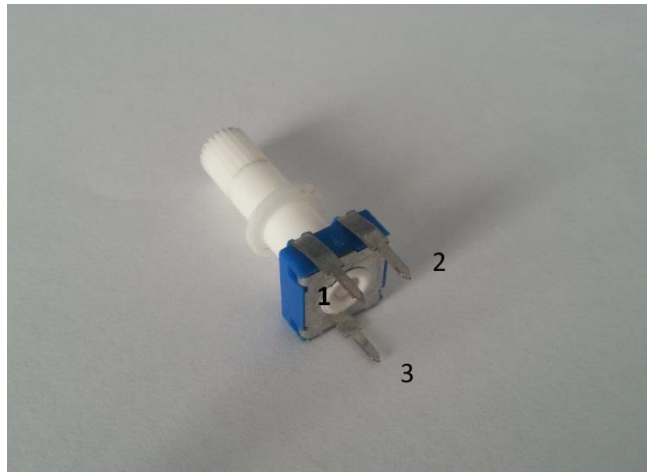


Imagen 9

- Después, he colocado el servo motor, que tiene 3 cables: el de color rojo (va a positivo), el de color negro (va a GND) y el de color blanco (va al pin digital 9).

Este es el circuito realizado en prácticas.

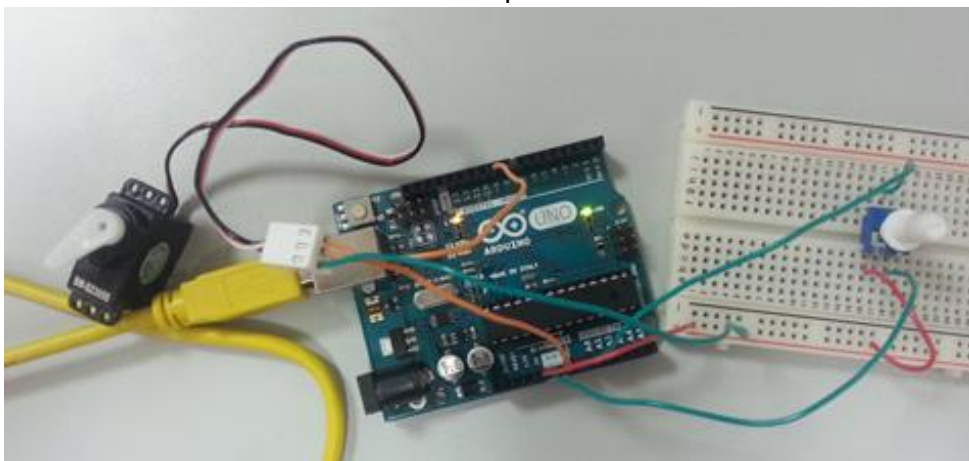


Imagen 10

- Una vez realizada la parte de hardware, me he pasado a la parte de Software. Primero, voy a escribir el código que se va a cargar en la placa Arduino. Hay que pasarle la librería de servo (**#include <Servo.h>**), para que permita a la placa arduino controlar servomotores. Ahora, creo una variable “myservo” como tipo Servo, a la que asignaré el pin digital 9, que es donde está conectado el servo. Tras esto, inicializaré la comunicación, y en el “loop” se irán leyendo los valores que nos estén llegando desde el programa Python.

El código sería:

```
#include <Servo.h>

Servo myservo;

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

void loop() {
    if(Serial.available){
        int a = Serial.parseInt();
        myservo.write(a);
    }
}
```

- Después, voy a programar en Python el código que permitirá mandarle comandos a la placa Arduino, que lo he llamado **“Servo.py”**, pero antes de escribir el código hay que instalar la librería **“python-serial”** en una ventana de símbolos para que no nos de error a la hora de ejecutarlo.

sudo apt-get install python-serial

Este es el código de la clase **Servo.py**:

```
import serial

arduino = serial.Serial('/dev/ttyACM0', 9600)
print("Starting !!")

while True:
    comando = raw_input('Introduce un comando')
    arduino.write(comando)

arduino.close() #Finaliza la comunicación
```

- Luego, en la ventana de comandos ejecutamos la instrucción: **cd Escritorio** que es donde tengo el archivo **Servo.py**.
- Por último, ejecuto la instrucción **python Servo.py** para ejecutar el programa y debería de funcionar. Al escribir, por ejemplo, 50 se moverá el servo 50°, y así con otros ejemplos (de 0° hasta 180°).

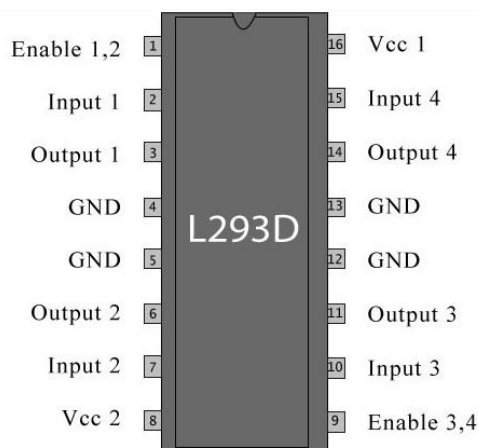
Nota: En este sexto ejercicio la parte de hardware no he tenido ningún problema porque solo era conectar un potenciómetro y las diferentes conexiones del servo, y la parte de software tampoco ha sido ningún problema. Para el código arduino he tenido que buscar información por internet porque no sabía que había que pasarle una librería para los servomotores, y para el código en python ha sido una pequeña modificación de uno de los ejercicios anteriores.

- 7) Control de la velocidad de giro de un motor de continua en doble sentido con un potenciómetro. (mitad señal potenciómetro, giro positivo, otra mitad señal potenciómetro, giro negativo).

En este séptimo ejercicio lo que hay que hacer es que a través de un potenciómetro haga girar un motor de continua en un sentido u otro, dependiendo de la posición del potenciómetro.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, he comenzado con la parte hardware. En la que he tenido que utilizar: un motor de continua, una pila de 9V (para poder alimentar el motor), un potenciómetro (que cuando lo giras a la derecha va en sentido horario y cuando lo giras a la izquierda va en sentido anti horario) y un puente H (para que me cambie las polaridades en las señales de salida con señales de control de entrada).



Esquema Puente H L293D (Imagen 11)

Esquema del circuito a realizar:

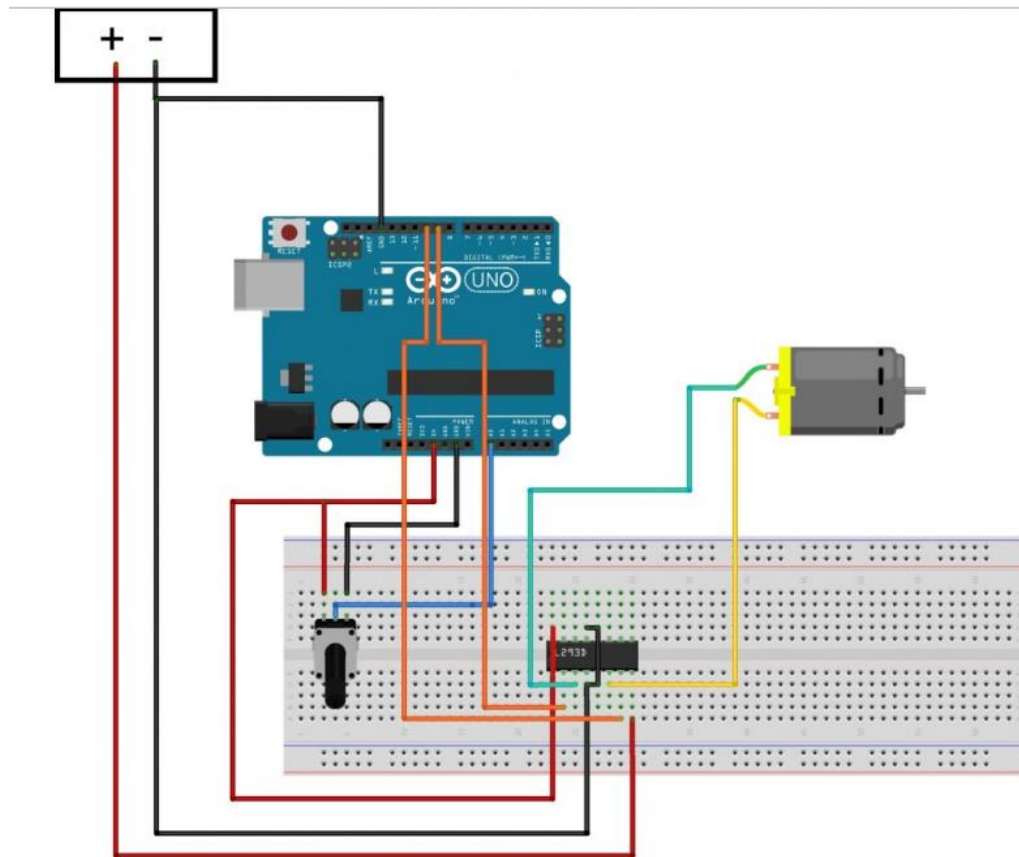


Imagen 12

He añadido este esquema por si no se ven bien las diferentes conexiones en el circuito que he realizado en prácticas.

Este es el circuito realizado en prácticas.

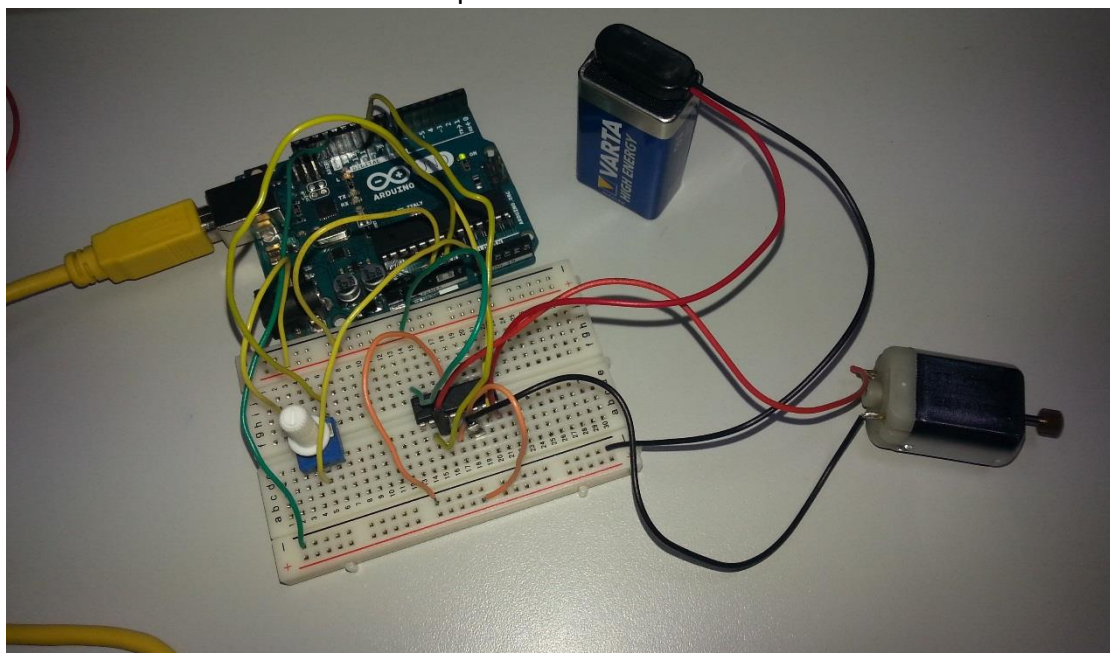


Imagen 13

- Una vez realizado el montaje del circuito me paso a la parte de software. En el código de arduino explico que es cada variable y qué hago.

El código sería:

```
int pin2=9;    //Entrada 2 del L293D al pin 9
int pin7=10;   //Entrada 7 del L293D al pin 10
int pot=A0;    //Potenciómetro al pin A0

int vpot;      //Variable que recoge el valor potenc.
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 potenc. en la variable
    vpot=analogRead(pot);

    //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(vpot, 0, 1023, 0, 255);
    pwm2 = map(vpot, 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);
}
```

- Por último, al ejecutar el programa, si se gira el potenciómetro a la derecha el motor se mueve en sentido horario y si se gira a la izquierda el motor se mueve en sentido anti horario.

Nota: En este séptimo ejercicio con la parte de hardware no he tenido dificultad a la hora de hacer el montaje del circuito, el único problema que he tenido ha sido a la hora de probarlo, que el motor de continua no funcionaba bien, porque uno de los cables tenía un mal contacto y ahí he perdido un poco de tiempo. En la parte de software he estado buscando información por internet para realizarlo, pero he entendido lo que hace el código de arduino.

8) Imprimir información en una pantalla LCD que envía el pc vía serie.

En este octavo ejercicio hay que hacer que a través del pc vía serie se mande un mensaje y que se muestre en una pantalla LCD.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, he comenzado por el montaje del circuito. He utilizado un potenciómetro para regular la iluminación de la pantalla LCD (ver imagen 9, para ver las diferentes patillas y los números indicados). La patilla 1 la he conectado a 5V, la patilla 2 a negativo y la patilla 3 al pin 3 (por la parte izquierda) de la pantalla LCD. El resto de conexiones lo muestro en el esquema (ver imagen 14) y montaje de mi circuito (ver imagen 15).

Esquema del circuito a realizar:

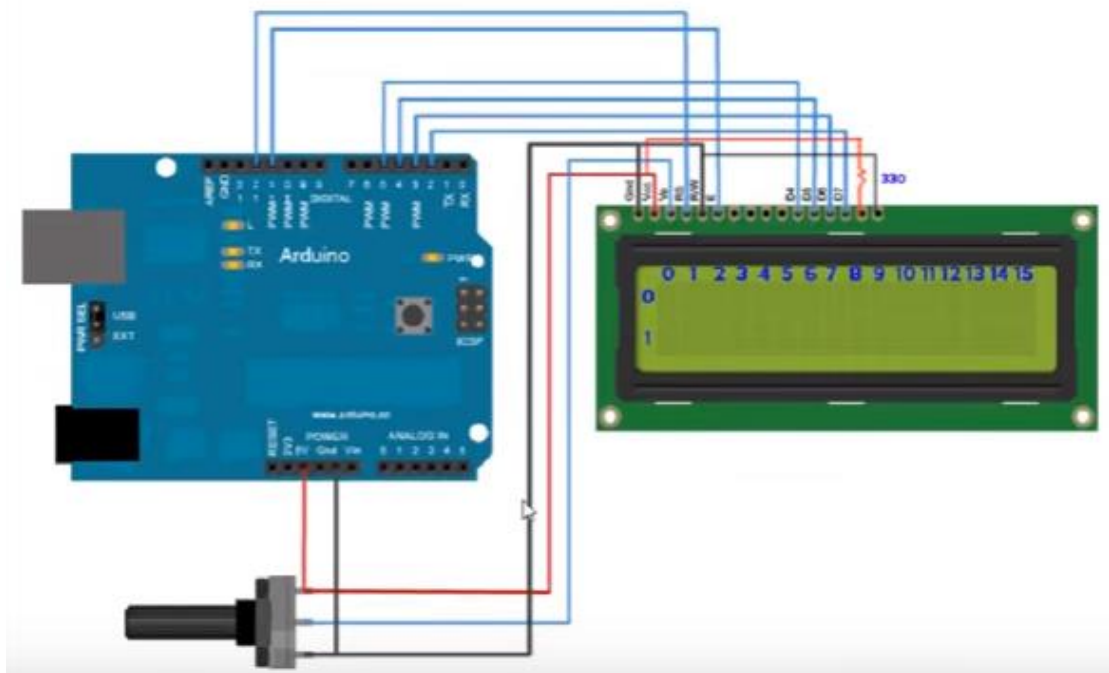


Imagen 14

Este es el circuito realizado en prácticas:

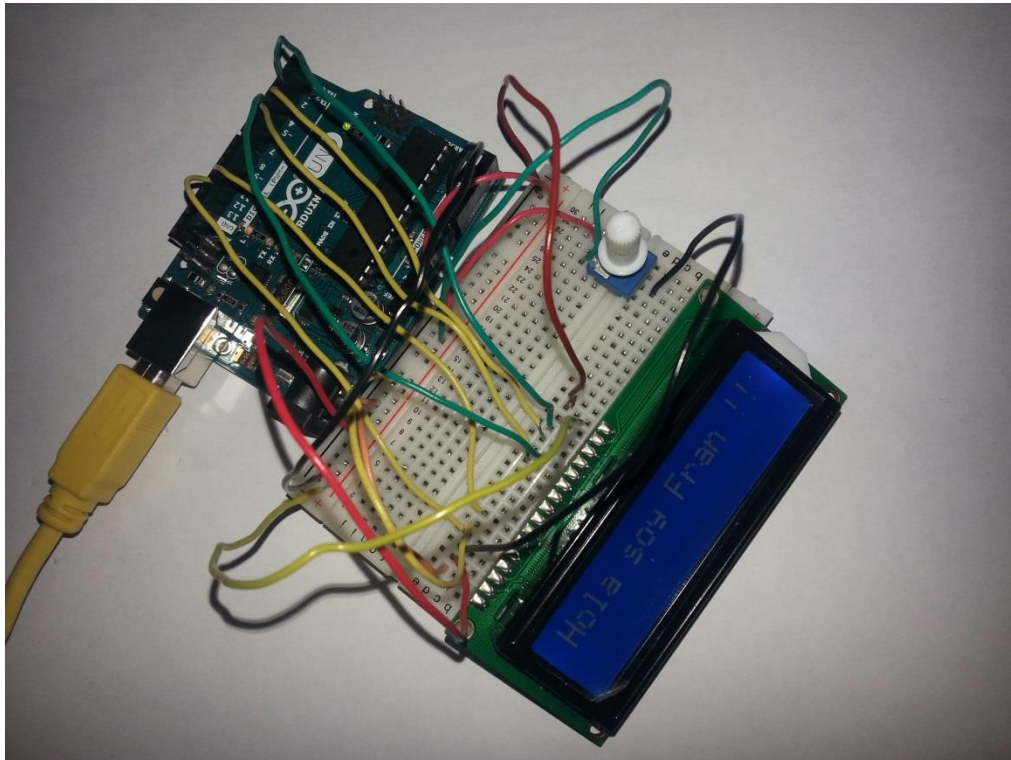


Imagen 15

- Una vez realizada la parte de hardware, me he pasado a la parte de Software. Primero, voy a escribir el código que se va a cargar en la placa Arduino. Hay que pasarle la librería de Lcd (**#include <LiquidCrystal.h>**), para que permita a la placa arduino controlar las pantallas lcd. Se le pasan las conexiones de la pantalla que van en los pines digitales (12, 11, 5, 4, 3, 2) y se crean dos variables: "caracter" de tipo char y "cadena" de tipo String. Tras esto, inicializaré la comunicación, y en el "loop" se irán leyendo los caracteres de la frase que nos está llegando desde el programa Python (mientras Serial.available() sea mayor que 0).

El código sería:

```
#include <LiquidCrystal.h>

LiquidCrystal Pantalla(12, 11, 5, 4, 3, 2);
char caracter;
String cadena;

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

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



```
delay(100);  
Pantalla.clear();  
while(Serial.available() > 0){  
  
    character = Serial.read();  
    cadena += character;  
}  
Pantalla.print(cadena);  
character = '\0';  
cadena = "\0";  
}  
}
```

- Después, voy a programar en Python el código que permitirá mandarle comandos a la placa Arduino, que lo he llamado **"LCD.py"**, pero antes de escribir el código hay que instalar la librería **"python-serial"** en una ventana de símbolos para que no nos de error a la hora de ejecutarlo.

```
sudo apt-get install python-serial
```

Este es el código de la clase **LCD.py**:

```
import serial  
  
arduino = serial.Serial('/dev/ttyACM0', 9600)  
print('Starting !!')  
  
while True:  
    comando = raw_input('Introduce una  
                        frase')  
    arduino.write(comando)  
  
arduino.close() #Finaliza la comunicación
```

- Luego, en la ventana de comandos ejecutamos la instrucción: **cd Escritorio** que es donde tengo el archivo **LCD.py**.
- Por último, ejecuto la instrucción **python LCD.py** para ejecutar el programa y debería de funcionar. Al escribir una frase debería de mostrarla en la pantalla LCD que está en el montaje del circuito, en la imagen 15, se observa: **Hola soy Fran !!** .

Nota: En este octavo ejercicio con la parte de hardware el único problema que he tenido ha sido que como hay muchas conexiones, pues me he equivocado en algunas y no me funcionaba bien, pero al final lo he conseguido. En la parte de software no he tenido ningún problema en realizarlo, porque es la comunicación pc vía serie que es como he hecho en otros ejercicios anteriores, por lo que no me ha sido difícil de hacerlo y también he buscado información para hacer el programa de arduino.

- 9) Diseñar un termómetro con la temperatura en el display LCD, empleando el sensor de temperatura tmp36GZ.

En este noveno ejercicio lo que hay que hacer es que a través de un sensor de temperatura, con el circuito anterior, se muestre la temperatura en la pantalla LCD.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, he realizado la parte de hardware. He reutilizado el montaje del circuito anterior incluyendo un sensor de temperatura que es el tmp36GZ (ver imagen 16, para diferenciar las distintas patas del sensor). La pata 1 va a 5V, la 2 va al pin analógico A0 y la 3 va a GND.

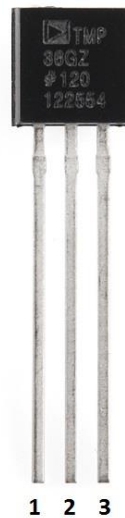
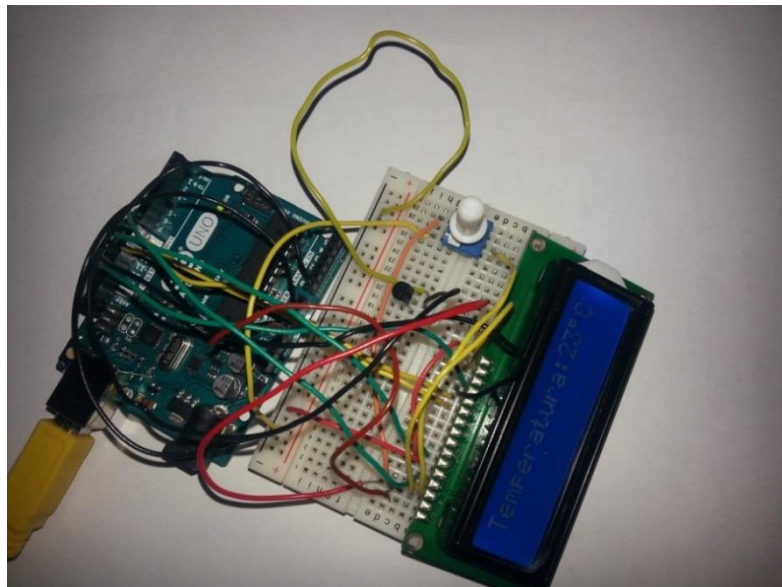


Imagen 16

Este es el circuito realizado en prácticas:



- Después, he realizado la parte de software. Hay que pasarle la librería de lcd (**#include <LiquidCrystal.h>**), para que permita a la placa arduino controlar las pantallas lcd. Se le pasan las conexiones de la pantalla que van en los pines digitales (12, 11, 5, 4, 3, 2) y se crea una variable: "sensorPin" de tipo int para la pata 2 del sensor, que va a ser el pin analógico A0. Tras esto, inicializaré la comunicación, y en el "loop", crearé una variable: "voltaje" de tipo float, que me calcula la tensión del circuito y otra variable: "temperature" de tipo int, que sigue una ecuación para obtener los grados (°C) dentro de un rango.
- Por último, se ejecuta el programa en arduino, y debería de mostrar la temperatura que hace en la pantalla del LCD. Al probarlo en clase me salió 23 °C.

El código sería:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5 , 4, 3, 2);
const int sensorPin = A0;

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

void loop(){
    int sensorVal = analogRead(sensorPin);

    Serial.print("Sensor Value: ");
    Serial.print(sensorVal);

    float voltage = (sensorVal/1024.0) * 5.0;

    Serial.print(", Voltios: ");
    Serial.print(voltage);
    Serial.print("; Temp. Grados: ");
    int temperature = (voltage - .5) * 100;
    Serial.println(temperature);

    lcd.begin(16, 2);
    lcd.setCursor(0,0);
    lcd.write("Temperatura: ");
    lcd.setCursor(12,0);
    lcd.print(temperature);
    lcd.setCursor(14,0);
    lcd.write((char)223);
    lcd.setCursor(15,0);
    lcd.write("C");
    delay(2000);
}
```

Nota: En este noveno ejercicio con la parte de hardware no he tenido ningún problema porque he reutilizado el montaje del circuito del ejercicio 8, lo único que he tenido que hacer es añadirle el sensor de temperatura. Con la parte de software, el único problema que he tenido ha sido que no entendía muy bien la ecuación para calcular la temperatura, pero al final lo he entendido. He buscado información por Internet para poder realizar la parte de software.

10) Diseñar un contador 00 a 59 en display 7-seg cada segundo.

En este décimo ejercicio lo que hay que hacer es que a través de una placa con un display integrado hacer un contador que vaya de 00 hasta 59 cada segundo, y que vuelva a comenzar una vez que llegue a 59.

Este ejercicio lo he realizado de la siguiente manera:

- Primero, he comenzado por la parte de hardware. Hay que utilizar una placa proporcionada por el profesor, que va a hacer de contador a través de unos displays 7-segmentos que tiene. Simplemente, lo que hay que hacer es colocar la parte con más pines a la parte digital de la placa arduino, y la parte con menos pines va a la parte analógica de arduino.

Esta es la placa (circuito) que hace como contador:

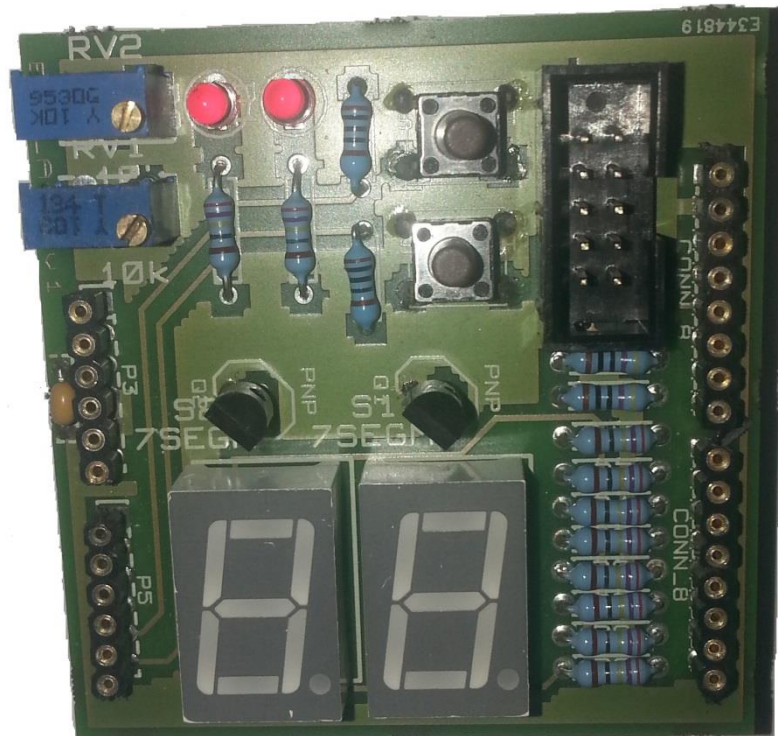


Imagen 18

- En la siguiente imagen, se observa el circuito realizado en prácticas, en el que se ve el display encendido con el número 55. El display tiene que contar desde 0 hasta 59.

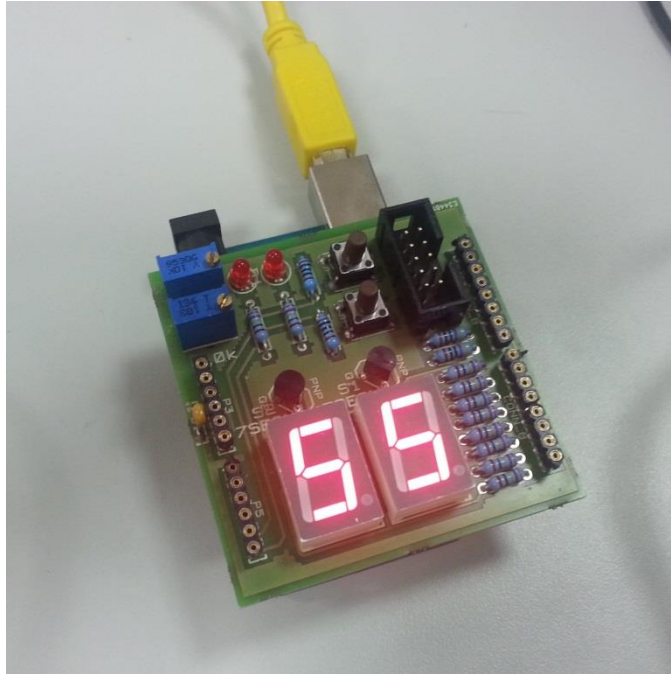


Imagen 19

- Después, he realizado la parte de software. El código ha sido proporcionado por el profesor, pero ha visto que completar algunas cosas. Primero, declara una variable de tipo Array **"segPins[]"**, en el que le pasa los pines utilizados en la placa arduino, una variable **"tiempototal"** de tipo int, que lo que hace es que vaya a la velocidad del segundo con la fórmula que se encuentra en el "refresh". También, hay una variable **"j"**, de tipo int, que lo que hace es que parpadee más rápido o más lento el display (si se pone muy bajo, por ejemplo, un valor de 10, se irá incrementando el valor del contador pero los displays parpadearán, sin embargo, si la j es más alta ese parpadeo no se notará a simple vista) y unas variables donde se guarden el valor de los displays: **"dat0"** y **"dat1"**, de tipo int, que tienen que estar a 0 los dos si se quiere que comience el contador desde 0. Hay una serie de funciones que son: **"refresh"**, **"write_data"**, **"write7seg"** y se explica lo que hace cada función en el código más abajo.

El código sería:

```
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;

void setup() {
    for (int thisseg = 0; thisseg < 8; thisseg++) {
        pinMode(segPins[thisseg], OUTPUT);
    }
    pinMode(disp1, OUTPUT);
    pinMode(disp2, OUTPUT);
}

void loop() {
    //Llamada a función refresh pasando los datos
    //correctos

    refresh(dat1,dat0);

    // Cálculo correcto de los datos dat1, dat0 --> desde
    //0 0 hasta 5 9

    dat1 = dat1 + 1;
    if(dat1 == 10){
        dat0 = dat0 + 1;
        dat1 = 0;
    }
    if(dat0 == 6){
        dat0 = 0;
        dat1 = 0;
    }
}

// Función refresh: Duración total de ejecución de
refresh: tiempototal.
//Se van intercambiando los displays (disp1 con dat0
y disp2 con dat1) a una frecuencia que evite el
parpadeo (j--> numero de veces que se activan ambos
displays)

void refresh( int data1, int data0) {
    int tiempo_refresco = tiempototal/(2*j);
```

```
// Bucle de activación de los displays. El bucle se
ejecuta j veces. Para escribir en los displays se
llama a la función write_data (dato)
```

```
for(int i = 0; i <= j;i++){
    digitalWrite(displ1,0);
    digitalWrite(displ2,1);
    write_data(dat1);
    delay(tiempo_refresco);
    digitalWrite(displ1,1);
    digitalWrite(displ2,0);
    write_data(dat0);
    delay(tiempo_refresco);
}
}
```

```
// Función write_data(dato): Transforma dato en su
código siete segmentos para escribirlo en el display
```

```
void write_data (int arg) {

    switch (arg) {
        case 0:
            //do something when var equals 1
            write7seg(0x7e);
            break;
        case 1:
            //do something when var equals 2
            write7seg(0x30);
            break;
        case 2:
            //do something when var equals 1
            write7seg(0x6d);
            break;
        case 3:
            //do something when var equals 2
            write7seg(0x79);
            break;
        case 4:
            //do something when var equals 1
            write7seg(0x33);
            break;
        case 5:
            //do something when var equals 2
            write7seg(0x5b);
            break;
        case 6:
            //do something when var equals 1
            write7seg(0x1f);
            break;
        case 7:
            //do something when var equals 2
            write7seg(0x70);
            break;
    }
}
```

```
        case 8:
            //do something when var equals 1
            write7seg(0x7f);
            break;
        case 9:
            //do something when var equals 1
            write7seg(0x73);
            break;
    }
}

// Función write7seg(dato_7seg): escribe el valor
//dato_7seg en el display

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; }
}
```

- Luego, el profesor me ha dicho que haga un cambio en el programa, para que el contador empiece desde 40 y llegue hasta 80. Lo único que habría que cambiar sería:

```
//Para que empiece el display desde 40
int dat1 = 0;
int dat0 = 4;
```

```
//En la función "loop", para que llegue hasta 80 y
una vez que llegue empiece desde 40 y no desde 0.
```

```
dat1 = dat1 + 1;
    if(dat1 == 10){
        dat0 = dat0 + 1;
        dat1 = 0;
    }
    if(dat0 == 8){
        dat0 = 4;
        dat1 = 0;
    }
```

Circuito realizado en prácticas, el display va desde 40 hasta 80:

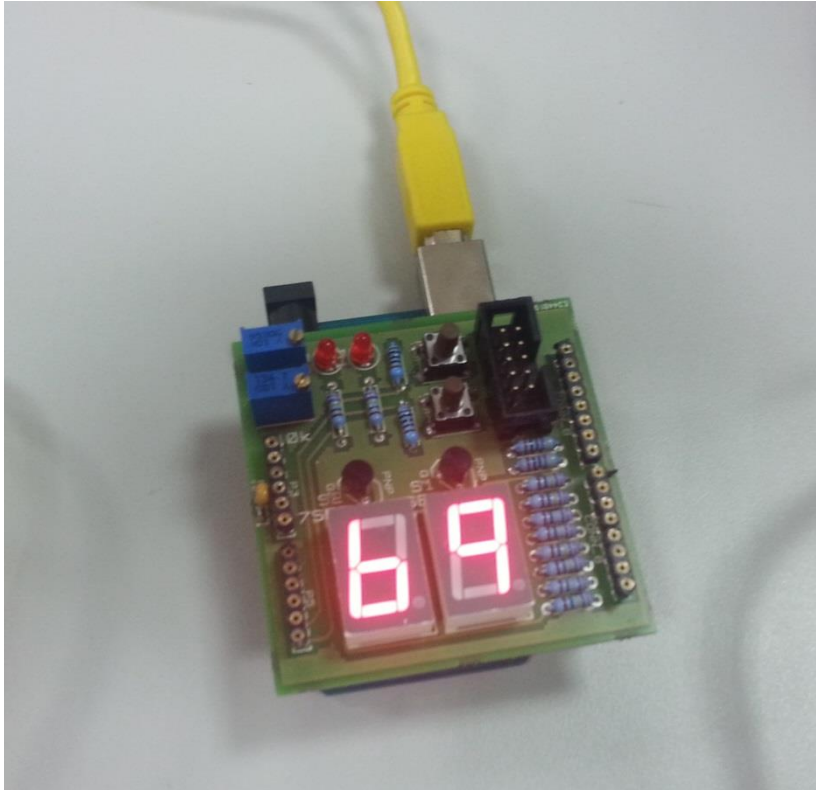


Imagen 20

Nota: En este décimo ejercicio no he tenido problemas con la parte de hardware, porque solo ha sido conectar la placa con los displays en la placa de arduino. En la parte de software tampoco he tenido problemas, porque el código lo ha proporcionado el profesor y ha habido que añadirle un par de cosas. También ha visto que hacerle unas modificaciones que nos ha dicho el profesor, pero han sido fáciles.

Conclusión

Con la realización de estas tres prácticas de Arduino, he aprendido a desenvolverme mejor con el uso de la placa y con el entorno de programación, aunque ya tenía una idea sobre Arduino, porque ya había hecho algunas prácticas en otra asignatura, pero eran ejercicios más básicos.

Algunas cosas que me han parecido interesantes en las prácticas son: la comunicación vía puerto serie, que ha habido que hacerlo a través de un programa en Python, que al principio no entendía muy bien, porque nunca he programado en Python, pero al final lo he entendido. También, me ha parecido interesante el ejercicio del relé con la bombilla y el ejercicio de la pantalla LCD que me muestra la temperatura.

Glosario

- **Atmel AVR:** es una familia de microcontroladores RISC del fabricante estadounidense Atmel.
- **Display:** dispositivo de ciertos aparatos electrónicos que permite mostrar información al usuario de manera visual.
- **GND (toma de tierra):** se emplea en las instalaciones eléctricas para llevar a tierra cualquier derivación indebida de la corriente eléctrica a los elementos que puedan estar en contacto, ya sea directa o indirectamente, con los usuarios de aparatos de uso normal, por un fallo del aislamiento de los conductores activos, evitando el paso de corriente al posible usuario.
- **Interrupción:** es una señal recibida por el procesador de una computadora, para indicarle que debe interrumpir el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.
- **Led:** es un diodo que emite luz.
- **Microcontrolador:** es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica.
- **Motor c.continua:** es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio, gracias a la acción que se genera del campo magnético.
- **Pantalla LCD:** es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.
- **PCB:** es la superficie constituida por caminos, pistas o buses de material conductor laminadas sobre una base no conductora. El circuito impreso se utiliza para conectar eléctricamente a través de las pistas conductoras, y sostener mecánicamente, por medio de la base, un conjunto de componentes electrónicos.
- **Potenciómetro:** resistencia variables; componente con tres terminales. En modo normal los extremos se conecta a la alimentación (VDD, GND) de manera que, con la resistencia variable, el tercer terminal puede cambiarse de forma mecánica entre VDD y GND.

- **Puente H:** circuito que permite intercambiar la polaridad en las señales de salida con señales de control de entrada.
- **Pull-down:** funciona igual que Pull-up, pero está conectado a tierra. Se mantiene la señal lógica a un nivel lógico bajo cuando no está conectado ningún otro dispositivo activo.
- **Pull-up:** es la acción de elevar la tensión de salida de un circuito lógico, a la tensión que, por lo general mediante un divisor de tensión, se pone a la entrada de un amplificador con el fin de desplazar su punto de trabajo.
- **Pulsador:** es un dispositivo utilizado para realizar cierta función. Los botones son por lo general activados, al ser pulsados con un dedo. Permiten el flujo de corriente mientras son accionados. Cuando ya no se presiona sobre él vuelve a su posición de reposo.
- **Regleta:** es un tablero con orificios que se encuentran conectados eléctricamente entre sí de manera interna, habitualmente siguiendo patrones de líneas, en el cual se pueden insertar componentes electrónicos y cables para el armado y prototipado de circuitos electrónicos y sistemas similares.
- **Reguladores de tensión:** es un dispositivo electrónico diseñado para mantener un nivel de tensión constante.
- **Relé:** es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.
- **Resistencia:** oposición que tienen los electrones al moverse a través de un conductor.
- **Sensor de temperatura:** componente hardware que tiene tres terminales. Dos de alimentación y la tercera de señal. Varía la tensión en función de la temperatura.
- **Servomotor:** es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición. 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 180 grados.

Bibliografía

- <https://www.arduino.cc/>
- <https://es.wikipedia.org/wiki/Arduino>
- <http://www.arduino.org/>
- <https://www.dte.us.es/docencia/etsii/gii-ic/laboratorio-de-desarrollo-hardware/lab/labs1/view>
- <https://geekytheory.com/arduino-raspberry-pi-raspduino/>
- <http://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino>
- <https://www.arduino.cc/en/Reference/AttachInterrupt>