

Universidad de Sevilla

Memorias Practicas

Laboratorio de Desarrollo de Hardware

Francisco Javier Solís Franco
30-11-2016

Índice

Memoria 1: Plataforma Arduino.....	2
Objetivos.....	2
Introducción.....	2
Desarrollo de la practica.....	4
Led's y bombillas.	4
Servos y motores.	6
LCD.	8
Contador.	12
Conclusión.....	14

Memoria 1: Plataforma Arduino.

Objetivos

Con estas prácticas pretendemos conocer la plataforma Arduino, sus características, sus variantes y como programar en ella, además de los componentes básicos y típicos que podemos usar para un sistema empotrado.

También prepararemos el PC para poder usar el entorno de desarrollo de Arduino, realizaremos ejemplos básicos y ejemplos con diferentes componentes hardware.

Realizaremos y tendremos como objetivo encender led's o luces, así como, mover un servo y un motor o la visualización de la temperatura en un display lcd que recogemos de un sensor de temperatura.

Introducción

¿Qué es Arduino?

Es una plataforma de desarrollo Open Hardware, es decir, de hardware abierto fácil de usar con la que podemos obtener información del entorno a través de sus pines de entrada o afectar a lo que le rodea.

Está basado inicialmente en microcontroladores AVR de 8 bits aunque algunos está basado en 32 bits.

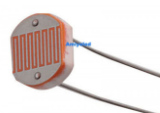
Los elementos hardware básicos que podemos conectar a esta placa pueden ir desde resistencias, condensadores y leds, hasta switches, display 7-segmentos, etc...

Aunque también podemos conectar potenciómetros, fotoresistencias, octoacopladores, sensores de temperaturas o relés como componentes habituales:

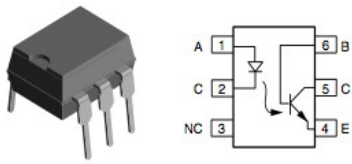
Potenciómetro: resistencia variable; componente con tres terminales. En modo normal los extremos se conectan a la alimentación (VDD, GND) de manera que, con la resistencia variable, el tercer terminal puede cambiarse de forma mecánica entre Vdd y GND.



Fotoresistencia (célula fotovoltaica): Componente biterminal que varía su resistencia con la luz.



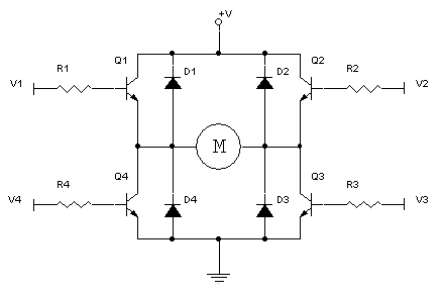
Optoacoplador: permite activar/desactivar un switch de un circuito desde otro circuito separado (fuentes de energía distintas).



Sensor de temperatura (LM36GZ): Tres terminales; Dos, de alimentación y el tercero de señal. Varía la tensión en función de la temperatura.



Puente H: un circuito que 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. Un microcontrolador puede cambiar el giro de rotación, para ello se emplea hardware añadido (típicamente un **puente H**).



Servo motor: básicamente 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. Típicamente se usa una señal PWM para colocar el servo en un determinado ángulo de giro.



Desarrollo de la practica

Para esta plataforma realizaremos un montaje y programación sencillas para la toma de contacto con la plataforma y el entorno de desarrollo.

Lo primero es instalar el IDE que lo hacemos descargándolo desde la web oficial y ejecutándolo. Una vez hecho esto procedemos a conectar la placa Arduino uno al puerto serie del PC y configuramos el entorno para comunicarnos con la placa.

Tras esto montaremos el primer sistema que consistirá en encender un led y cargaremos el programa de ejemplo Blink que nos proporciona el IDE.

Led's y bombillas.

Para seguir ampliando la toma de contacto mantendremos el mismo sistema, pero esta vez el programa encenderá o apagará el led según se le pase el comando "H" o "L" del PC a través de un programa en Python.

Código de Arduino:

```
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);
    }
}
}

```

Código Python:

```

import serial

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

print("Starting!")

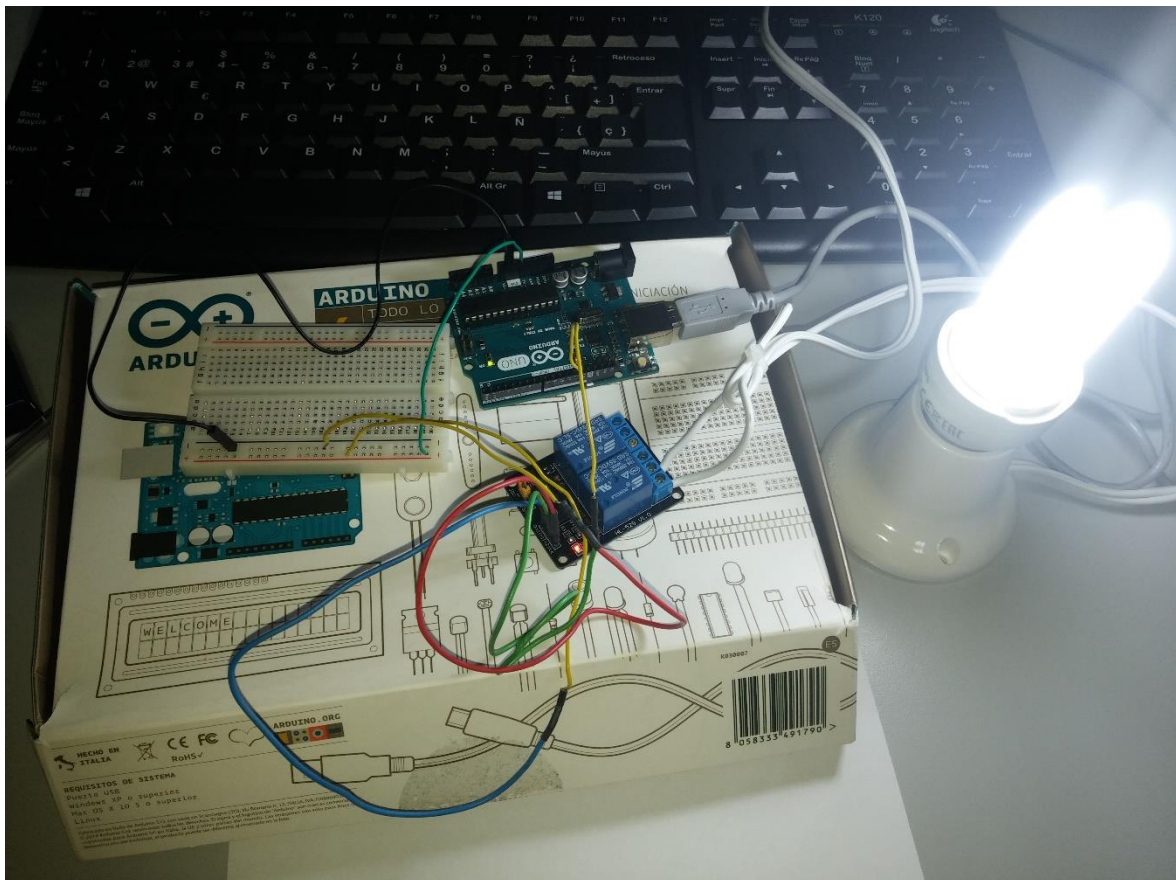
while True:
    comando = raw_input('Introduce un comando: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    if comando == 'H':
        print('LED ENCENDIDO')
    elif comando == 'L':
        print('LED APAGADO')

arduino.close() #Finalizamos la comunicacion

```

Continuamos procediendo a realizar la misma función, pero en vez de encender o apagar un led, lo realizaremos sobre una bombilla, por lo que tendremos que añadir un relé.

El código es el mismo que en el ejemplo anterior



Vídeo del funcionamiento: https://youtu.be/RpHb2n_JHU

Servos y motores.

Ahora realizaremos montajes usando servos y motores.

Para ello, comenzaremos controlando un servo con un potenciómetro.

(código, foto y video)

Para ampliar este conocimiento realizaremos el mismo control del servo sustituyendo el potenciómetro por un programa en Python al que pasaremos los grados de giro del servo.

Código Arduino:

```
#include <Servo.h>
Servo servol;
int posicion;

void setup() {
    Serial.begin(9600);
    servol.attach(9); //seleccionamos el pin 9 para el control
}

void loop() {
    if(Serial.available()) {
        posicion = Serial.parseInt();
        servol.write(posicion);
    }
}
```

Código Python:

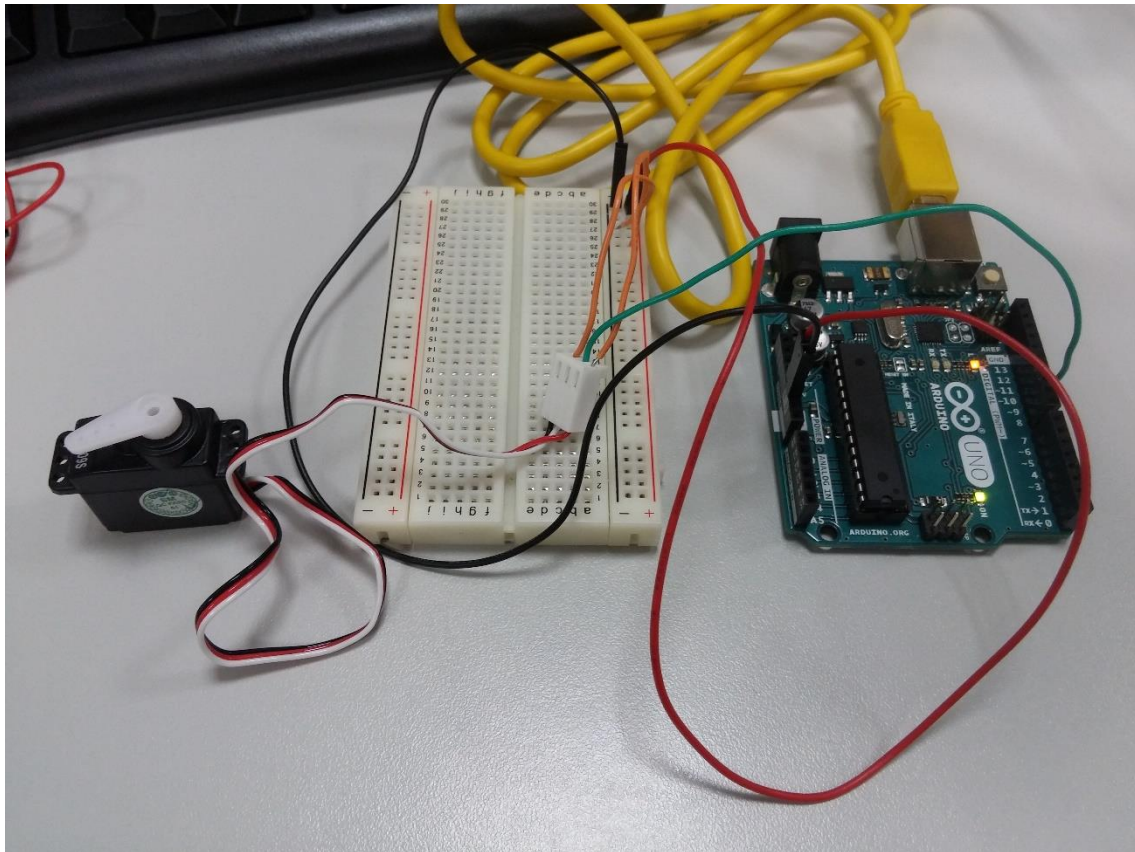
```
import serial

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

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



Vídeo de funcionamiento: <https://youtu.be/-1osAsn7jxU>

Y para terminar de afianzar el conocimiento sobre servos y motores, controlaremos un motor de continua en doble sentido, por lo que usaremos un motor DC y el chip L293DNE.

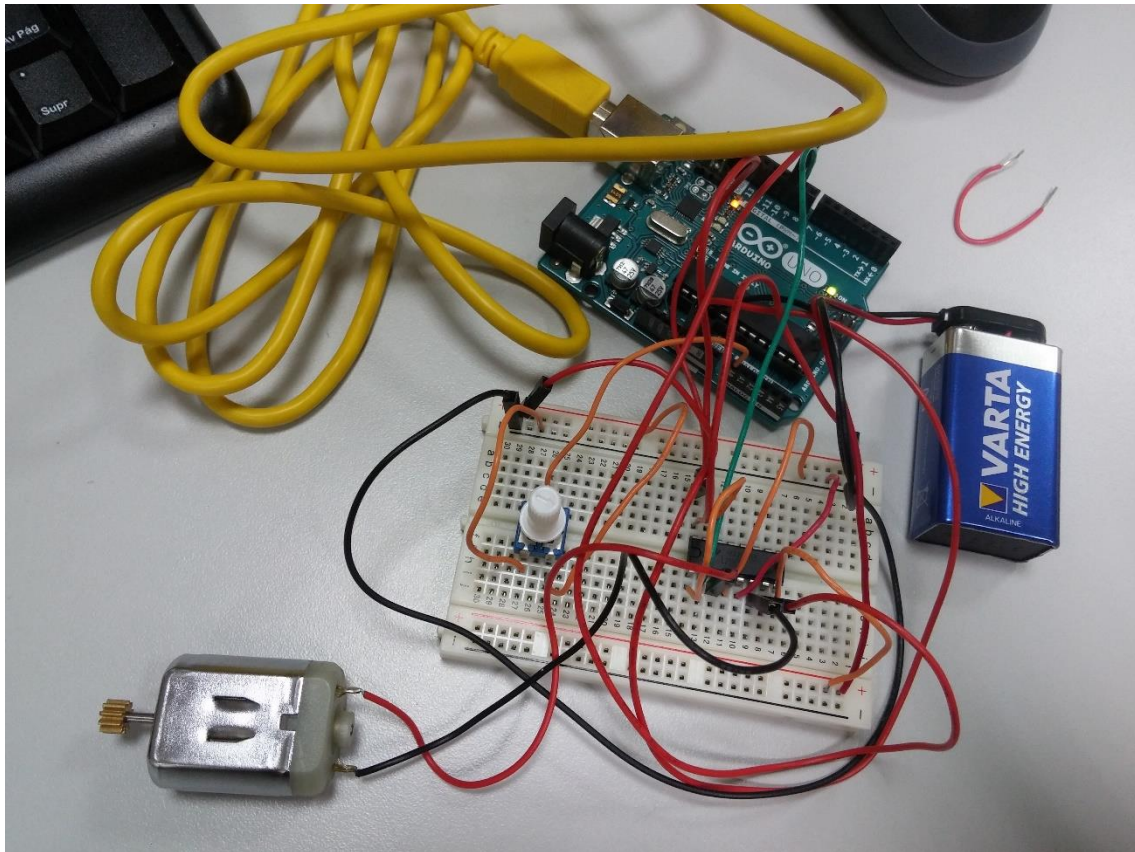
Código Arduino:

```
int pin2=9; //entrada 2 del L293D
int pin7=10; //entrada 7 del L293D
int pote=A0; //entrada del potenciómetro

int valorpote; //variable que recoge el valor del potenciómetro
int pwm1; //variable del pwm1
int pwm2; //variable del pwm2

void setup() {
    //inicializamos los pins de salida
    pinMode(pin2, OUTPUT);
    pinMode(pin7, OUTPUT);
}

void loop() {
    //almacenamos el valor del potenciómetro en la variable
    valorpote = analogRead(pote);
    //mapeamos los pwm
    pwm1 = map(valorpote, 0, 1023, 0, 255);
    pwm2 = map(valorpote, 0, 1023, 255, 0); //pwm2 está invertido
    //sacamos el pwm de las salidas
    analogWrite(pin2, pwm1);
    analogWrite(pin7, pwm2);
}
```

Vídeo del funcionamiento: https://youtu.be/_qsshKG6BC4

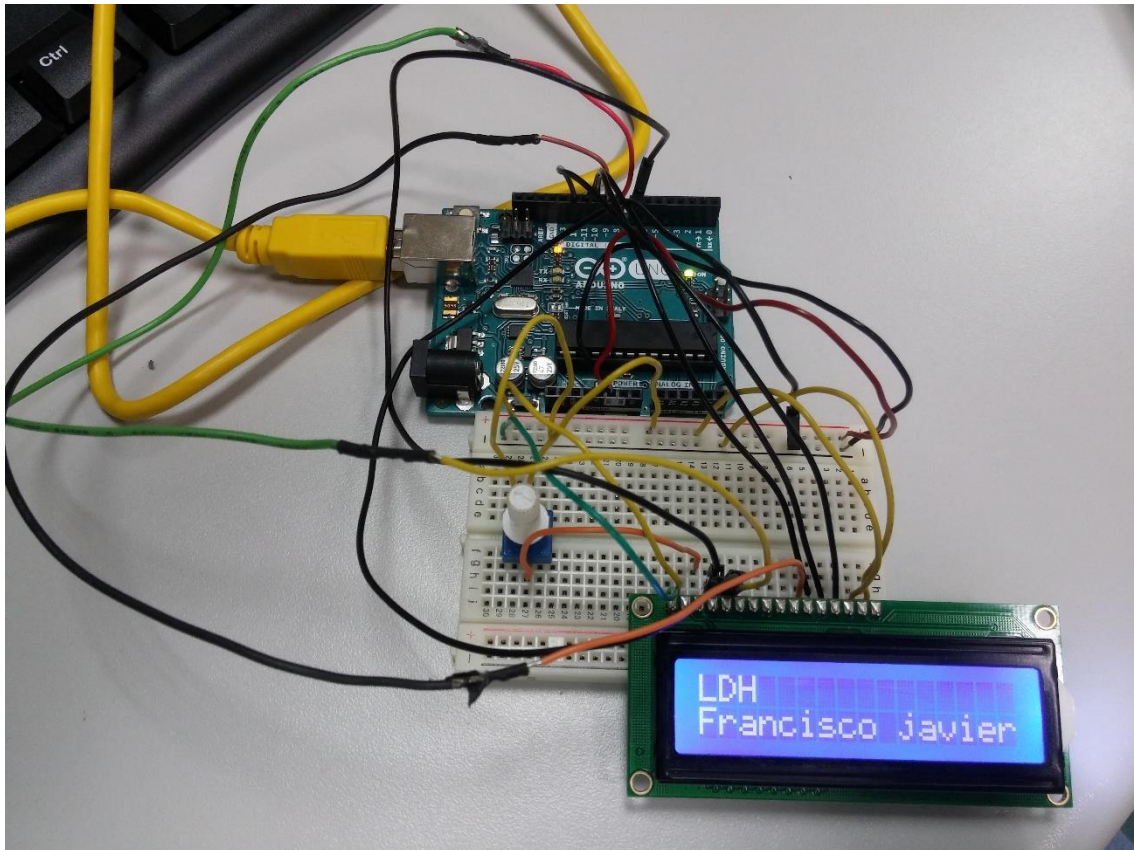
LCD.

Para trabajar con los lcd's emplearemos la pantalla LCM1602C que se basa en un controlador Hitachi HD44780 e importaremos la librería de Arduino "LiquidCrystal".

Comenzaremos por realizar un ejemplo básico en el que mostraremos en dos filas el nombre de la asignatura y el nombre del alumno.

Para lo cual conectaremos el LCD de la siguiente forma:

- Pin 4(LCD) -> Pin 7(Arduino)
- Pin 5(LCD) -> Pin GND(Arduino)
- Pin 6(LCD) -> Pin 8(Arduino)
- Pin 11(LCD)-> Pin 9(Arduino)
- Pin 12(LCD)-> Pin 10(Arduino)
- Pin 13(LCD)-> Pin 11(Arduino)
- Pin 14(LCD)-> Pin 12(Arduino)



Código Arduino:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7,8,9,10,11,12);

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

void loop() {
    lcd.setCursor(0,0);
    lcd.write("LDH");
    lcd.setCursor(0,1);
    lcd.write("Francisco Javier");
}
```

Continuamos modificando el ejemplo anterior para que nos muestre un mensaje enviado desde el PC, con lo cual usaremos el código Python de ejemplos anteriores para realizar la comunicación por puerto serie.

Código Arduino:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7,8,9,10,11,12);

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

```

}
void loop(){
    if(Serial.available()){
        lcd.setCursor(0,0);
        lcd.print(Serial.readString());
        lcd.setCursor(0,1);
        lcd.print("");
    }
}

```

Código Python:

```

import serial

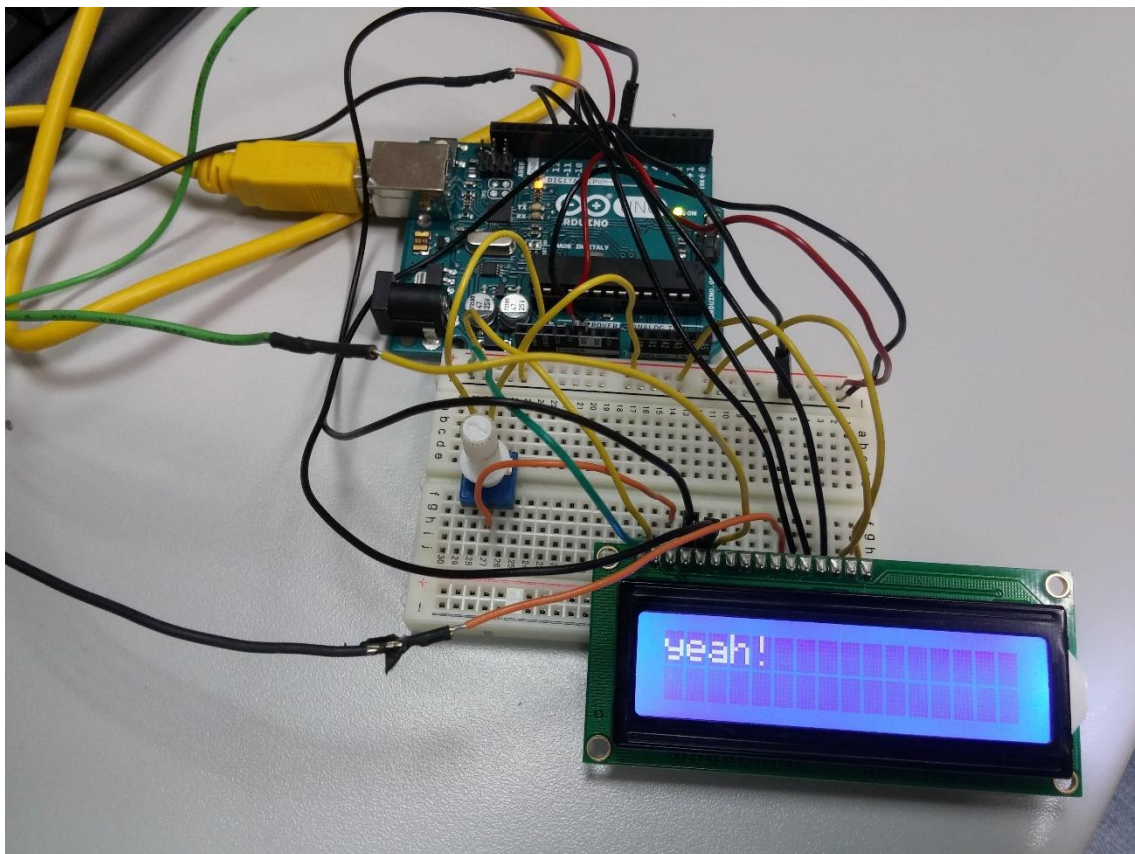
arduino = serial.Serial('/dev/ttyACM0', 9600)

print("Starting!")

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

arduino.close() #Finalizamos la comunicacion

```

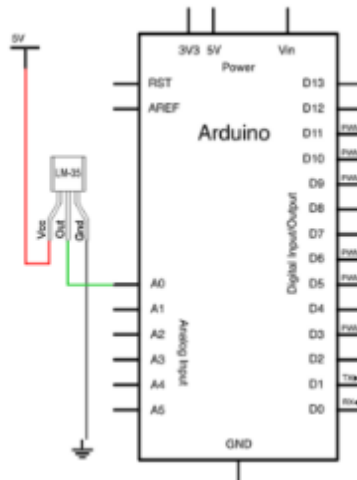


Vídeo del funcionamiento: <https://youtu.be/DxXCvxJn248>

Para finalizar realizaremos un termómetro con el sensor de temperatura TMP36GZ y el LCD.

Para ello recogemos la información suministrada por el sensor y lo mostraremos en el LCD, el esquema de conexión es el siguiente:

(esquema, código, foto y video)



Código Arduino:

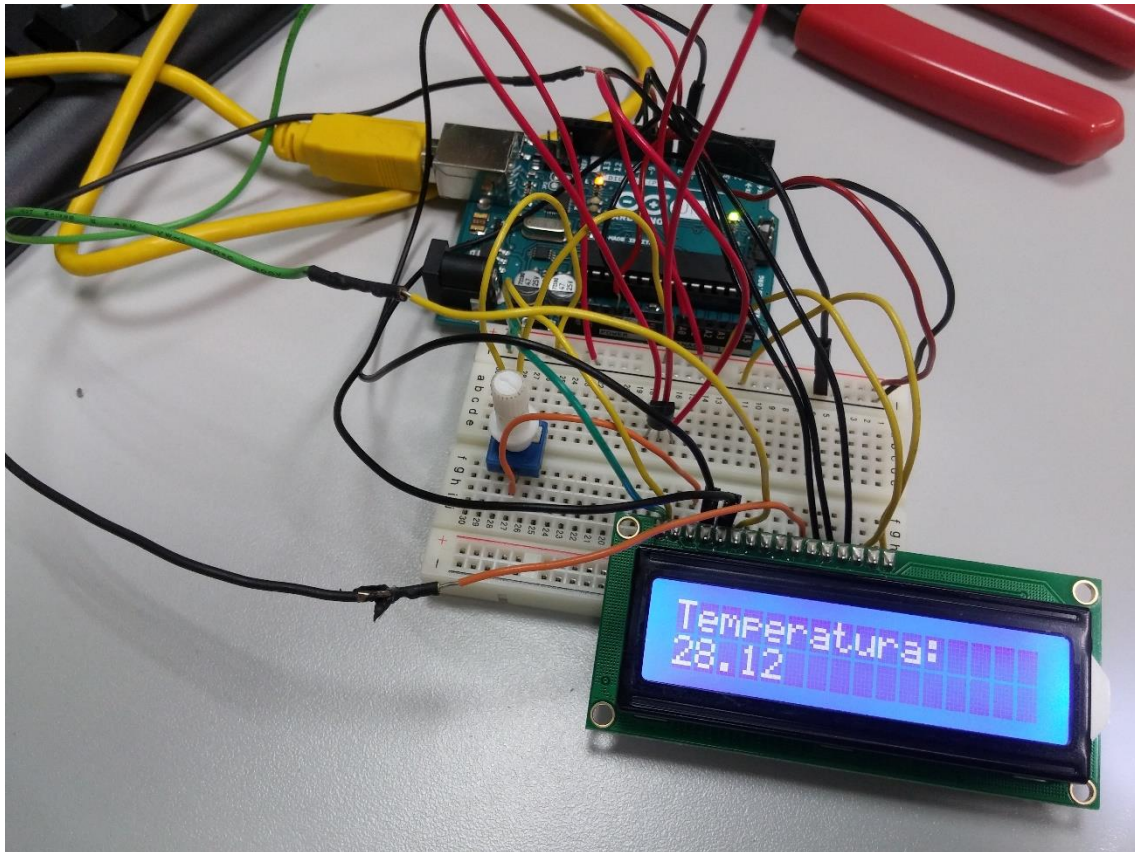
```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7,8,9,10,11,12);
const int sensorPin = A0;
int sensorVal;
float voltage;
float temperature;

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

void loop() {
  sensorVal = analogRead(sensorPin);
  voltage = (sensorVal/1024.0) * 0.5 ;
  temperature = (voltage - .5)* 100 ;

  lcd.setCursor(0,0);
  lcd.print("Temperatura:");
  lcd.setCursor(0,1);
  lcd.print(temperature);
  delay(1000);
}
```

Vídeo del funcionamiento: <https://youtu.be/5-ZPU0OqqtM>

Contador.

Ahora trataremos de realizar un contador de 60 segundos con displays 7 segmentos mediante una shield que colocaremos directamente sobre la Arduino Uno.

Tendremos que hacer que el display se refresque cada segundo con el nuevo valor de la cuenta y que cuando llegue a 59 se vuelva a poner a 00, por lo que tendremos que controlar dos datos que se mostraran en dos 7 segmentos de la siguiente manera:

Ambos displays empezaran en 0, el display que lleva las unidades irá de 0 a 9 y cuando llegue a 9 el display de las decenas se incrementará, el cual al llegar a 5 y el otro 9 ambos vuelven a 0, es decir, el funcionamiento básico de un contador de 60 segundos.

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 cont = 0 ;
int dat1 = 0 ;
int dat0 = 0 ;

void setup() {
  // put your setup code here, to run once:
  // loop over the pin array and set them all to output:
  for (int thisseg = 0; thisseg < 8; thisseg++) {
    pinMode(segPins[thisseg], OUTPUT);
  }
}
```

```

    pinMode(displ1, OUTPUT);
    pinMode(displ2, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    // dat1 = 0;
    // dat0 = 0;
    // while (1) {
    //Llamada a función refresh pasando los datos correctos
        dat0 = cont / 10 ;
        dat1 = cont % 10 ;
        refresh(dat1, dat0);
        if(cont < 59){
            cont = cont + 1 ;
        }else{
            cont = 0 ;
        }
    }
    // Cálculo correcto de los datos dat1, dat0 --> desde 0 0 hasta 5 9
}
// Función refresh: Duración total de ejecución de refresh:
tiempototal.
//Se van intercambiando los displays (displ1 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; }
}

```

Vídeo del funcionamiento: <https://youtu.be/jiPyWegOzoI>

Conclusión

Hemos conocido la plataforma Arduino, su entorno de desarrollo y su programación básica, realizando ejemplos básicos y comprobando su funcionamiento con componentes hardware básicos y típicos, concretamente con leds, relés, pantalla LCD, sensor de temperatura, servo y un motor de continua, alcanzando los objetivos propuesto.

De esta forma comprobamos como esta plataforma nos ofrece una gran cantidad de opciones para el aprendizaje de microcontroladores y nos da la oportunidad de poder probar nuestro conocimientos y estudios.

Además de darnos la libertad de poder probar prototipos y mezclar infinidad de sensores, aprender a usarlos correctamente.