

LHD: Memoria de Practica

ARDUINO

Alejandro González Sánchez

INDICE

-
- Objetivos pág. 3
 - Actividades Realizadas pág. 4
 - Conclusiones pág. 26

OBJETIVOS

- Conocer la plataforma arduino, sus características, sus variantes, sus modos de programación.
- Conocer una serie de componentes básicos de hardware típicos de aplicaciones de sistemas empuotrados.
- Preparar el PC para que funcione el entorno de desarrollo de arduino.
- Realizar ejemplos básicos de funcionamiento sobre arduino.
- Desarrollar otros ejemplos de uso de arduino manejando diversos componentes hardware.

ACTIVIDADES REALIZADAS

En la primera práctica nos hemos familiarizado con los diversos componentes que vamos a usar durante las distintas sesiones de prácticas con las diversas placas de desarrollo hardware.

En esta primera sesión hemos podido programar mediante python una aplicación mediante la cual hemos comunicado con el puerto serie y podido controlar y comunicar nuestro sistema enlazando Arduino y nuestro ordenador.

El profesor dio una introducción teórica para poder realizar la práctica, explicando el sistema Arduino brevemente.

Arduino es una plataforma de desarrollo de hardware abierta 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 y puede afectar aquello que le rodea controlando luces, motores y otros tipo de actuadores. El arduino que experimentamos en la práctica es un arduino ONE. Las ventajas que presenta arduino frente a otras plataformas son el entorno de desarrollo que es muy fácil de manejar, tiene un amplio conjunto de librerías de manejo de periféricos y que detrás hay una comunidad de desarrolladores muy grande.

Los componentes utilizados durante esta primera práctica

- Potenciómetro: resistencia variable, es un componente con tres terminales. El modo normal es conectar los extremos a VDD y GND, de manera que con la resistencia variable, el tercer terminal puede cambiarse de forma mecánica entre VDD y GND

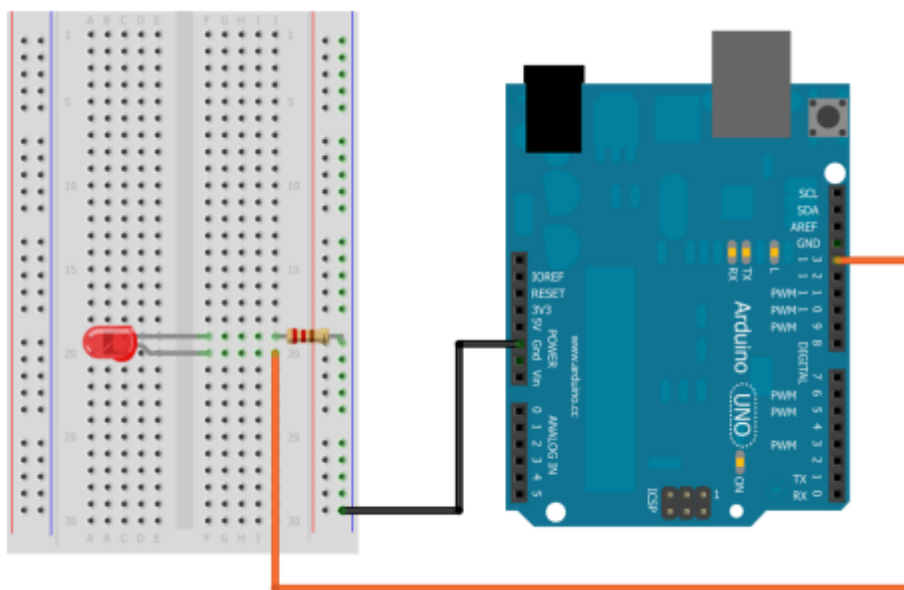
- Relé: Funciona como interruptor controlado por señal eléctrica (generalmente el control es de una señal electrónica de pocos voltios, mientras la salida puede ser señal de electricidad)

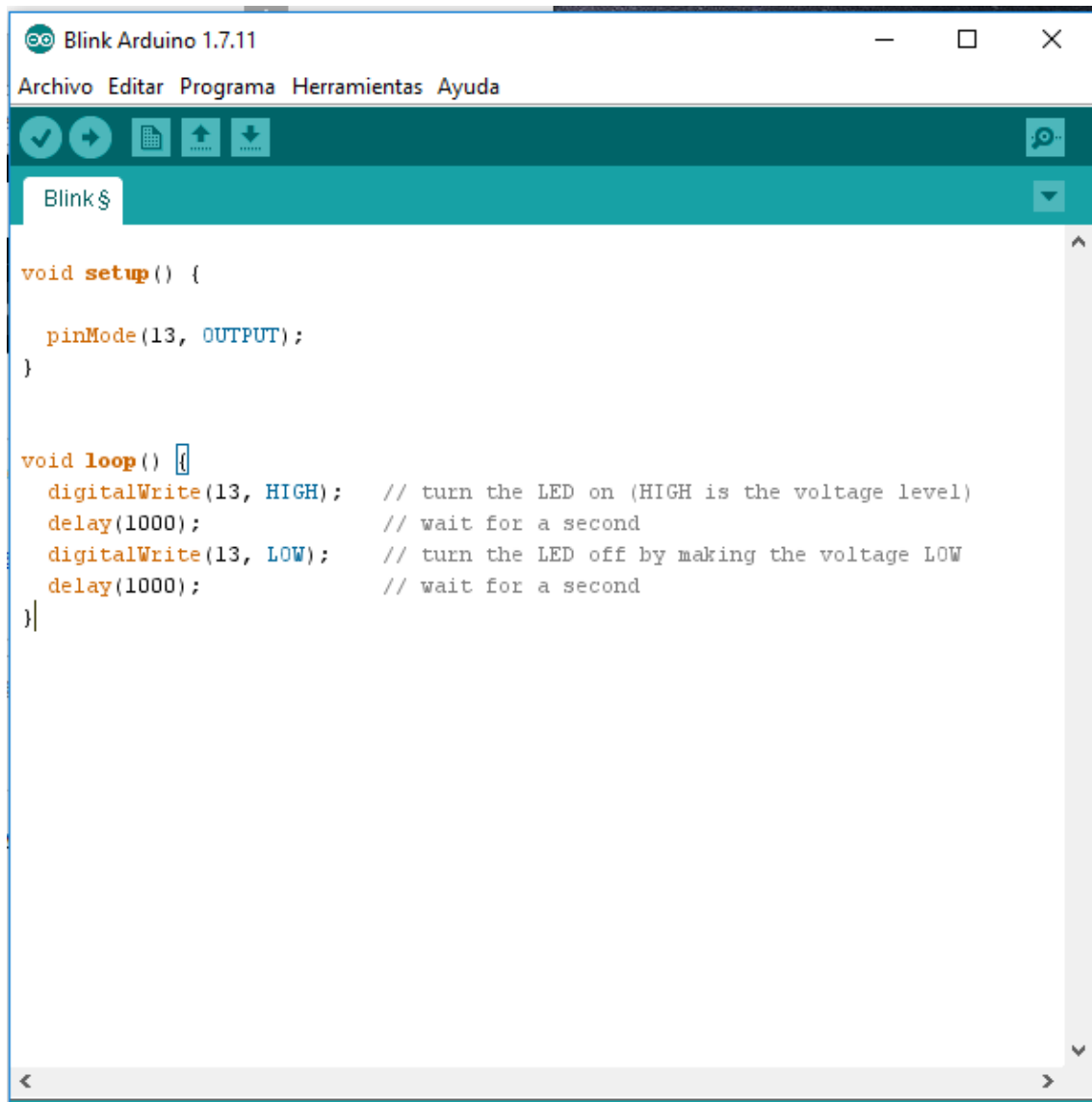
- Puente H: un circuito que permite intercambiar la polaridad en las señales de salida con señales de control de entrada.

- Sensor de temperatura: Tres terminales; Dos, de alimentación y el tercero de señal. Varía la tensión en función de la temperatura.
- Motor de corriente continua: Una tensión de continua activa el giro del motor. Gira en dos sentidos según la polaridad de la corriente. Un micro controlador 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 180 grados. 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.

ACTIVIDAD 1

En esta sesión de práctica hemos ejecutado de la librería de ejemplos la función blink, la cual hace que parpadee un led un tiempo determinado dependiendo de las esperas del código, la cual ha sido en este caso de 1000.





Evaluación después de la actividad

Tras la realización de la práctica, he aprendido a usar el entorno de desarrollo de arduino , como funciona su esquema principal y como es el entorno de desarrollo arduino.

Incidencias

- Al no tener conocimiento previo de la materia, coste adicional al poder ejecutarla, igualmente con Linux.

Valoración personal

Me ha parecido muy interesante puesto que hemos aprendido como es el entorno de Arduino, poder conocer cómo se trabaja con ella y sus procedimientos y poder tener base mínima para poder trastear en casa con la placa y aprender cosas nuevas partiendo de aquí.

ACTIVIDAD 2

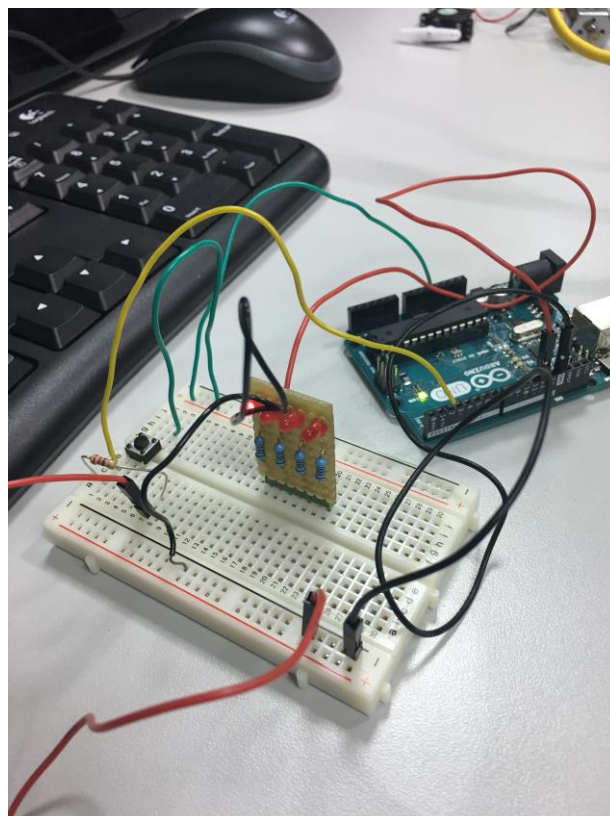
Mediante pulsador.

En esta segunda actividad debemos de controlar el encendido apagado del LED con un pulsador, el LED se encuentra en el pin 13, y el pulsador se encuentra en el pin 8. Entonces controlamos el led con el interruptor.

```
Blink §

int led = 13;
int pulsador1 = 2;
void setup() {
  pinMode(led, OUTPUT); // LED 13 como salida
  pinMode(pulsador1, INPUT); //pin 2 como entrada
  Serial.begin(9600); //Inicializo el puerto a 9600 baudios
}

void loop() {
  if(digitalRead(pulsador1) == HIGH){ //si el pulsador1 esta en alto
    digitalWrite(led, HIGH); //encender el LED
  }else{
    digitalWrite(led, LOW); //apagar el LED
  }
}
```



ACTIVIDAD 3

Controlar el encendido -apagado del LED desde el PC vía puerto serie.

En esta actividad tenemos que controlar el LED desde el ordenador. Para ello tenemos que hacer más variaciones en el código del Blink, además tenemos que tener instalado la herramienta de python en nuestro sistema Linux. Para ello usaremos el comando `sudo apt-get install python`.

Este es el código en python para realizar la comunicación.

```
/*Practica 2 TC 201617*/
serial.py
1  import serial
2  arduino=serial.Serial('/dev/ttyUSB1',9600)
3
4  print("Empezamos")
5  while True:
6      comando=raw_input('introduce comando:')
7      arduino.write(comando) #lo manda
8      if(comando == 'L'):
9          print ('Led ON')
10     elif comando=='H':
11         print('Led OFF')
12
13     arduino.close() #fin de la comunicacion
```

```
/*Practica 2 TC 201617*/
sublime.py x serial.py x arduinoLEDSEIE.py x
1  int led=13
2  void setup(){
3
4      pinMode(led,OUTPUT); //led 13 como salida
5      Serial.begin(9600); //inicializar puerto
6  }
7
8  void loop(){
9
10     if(Serial.available()){ //Si esta disponible
11         char c= Serial.read(); //guardamos la lectura en una variable
12         if(c=='H'){ //Si es una H
13             digitalWrite(led,HIGH);
14
15         } else if (c== 'L'){ //Si es una L
16             digitalWrite(led,LOW);
17         }
18     }
19 }
20 }
```


Una vez que lo ejecutamos nos dirá que introduzcamos un estado, si introducimos H se encenderá, y si introducimos L se apagará.

Evaluación después de la actividad

He conseguido conectar mediante puerto serie la placa de arduino con una app en python que hemos desarrollado

Incidencias

- Desconocimiento del puerto serie y su uso
- Búsqueda en internet de habilitar el puerto serie y usarlo entre los dos sistemas
- Mejor conocimiento y uso de los distintos tipos de puertos de Arduino

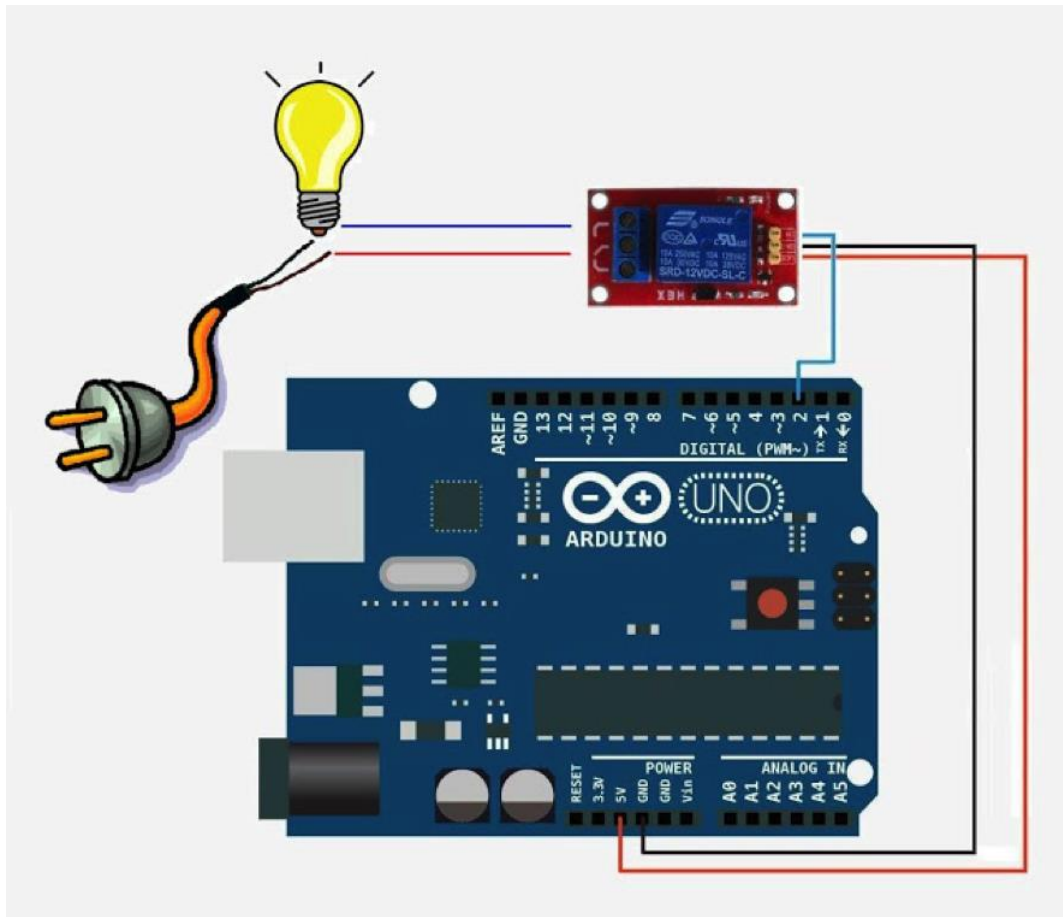
Valoración personal

Practica que me ha servido para conocer el envío de comandos a través del puerto serie utilizando el lenguaje Python desde un pc a una placa arduino así como para conocer el funcionamiento de dicho puerto

ACTIVIDAD 4

Una vez que hemos realizado la actividad anterior vamos a conectar un relé en el puerto dos, donde antes conectábamos el LED .

El relé: Funciona como un interruptor controlado por señal eléctrica (generalmente el control es de una señal electrónica de pocos voltios, mientras la salida puede ser señal de electricidad) el relé usado no soporta más de 10A y puede funcionar hasta 240V



Evaluación después de la actividad

Aprendizaje y conocimiento de que es un relé y su uso

Incidencias

- Se ha roto una bombilla al liar los cables y caerse al suelo y error con el relé

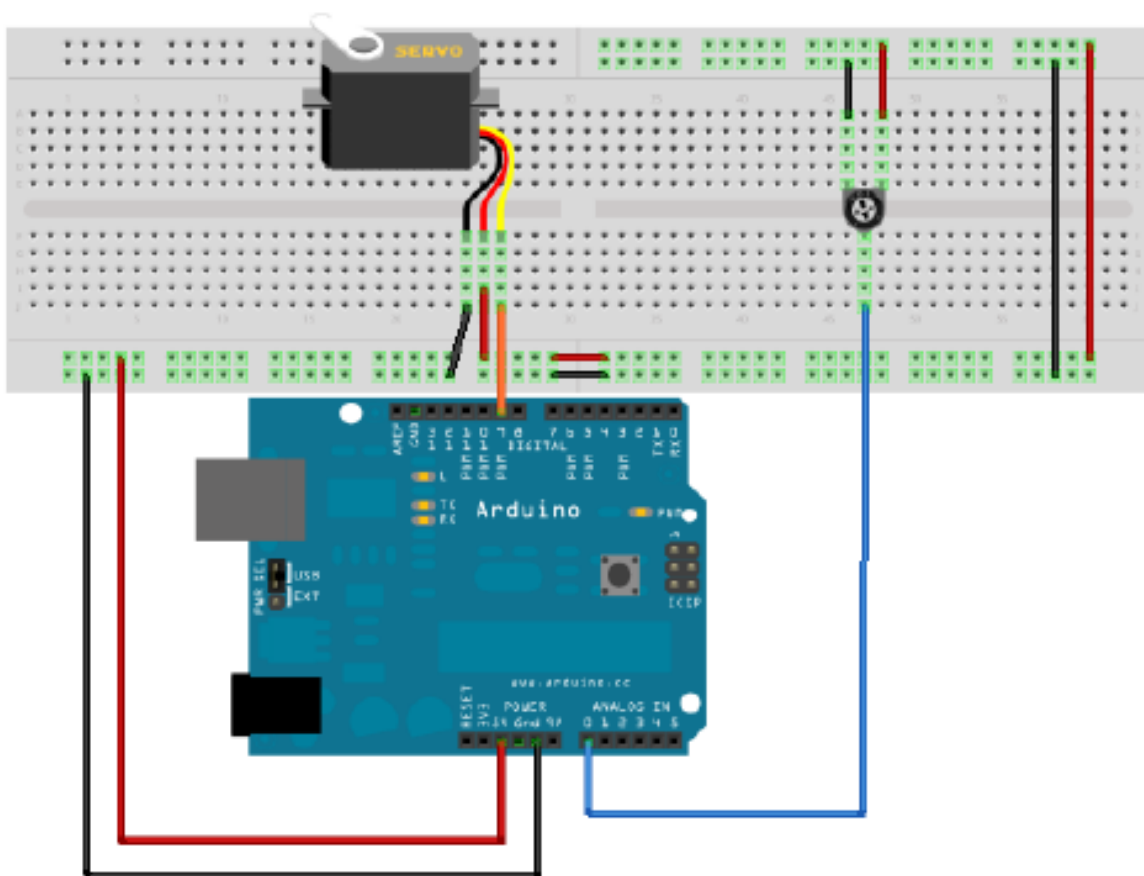
Valoración personal

Me ha parecido de especial interés puesto q a través de Arduino podemos controlar cualquier sistema que tengamos en casa

ACTIVIDAD 5

Control de la posición de un servo motor con un potenciómetro. Añadir librería servo en arduino. Servo motor: motor de DC.

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 unos 180°. El ángulo de giro depende del tamaño de 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.



En esta práctica hemos usado la librería de Servo.h

```
1 import serial
2
3 arduino = serial.Serial('/dev/ttyACM0', 9600)
4
5 print("Starting!")
6
7 while True:
8     val = raw_input('Introduce el angulo: ') #Input
9     arduino.write(val) #Mandar un comando hacia Arduino
10
11 arduino.close() #Finalizamos la comunicacion
```

Evaluación después de la actividad

Aprendizaje y conocimiento del uso de las salidas PWM de Arduino para poder simular una señal analógica entre 0-1024 que es la anchura del pulso

Incidencias

- Ninguna en particular

Valoración personal

Me ha gustado poder conocer cómo se podría controlar la dirección de un coche teledirigido mediante este sistema.

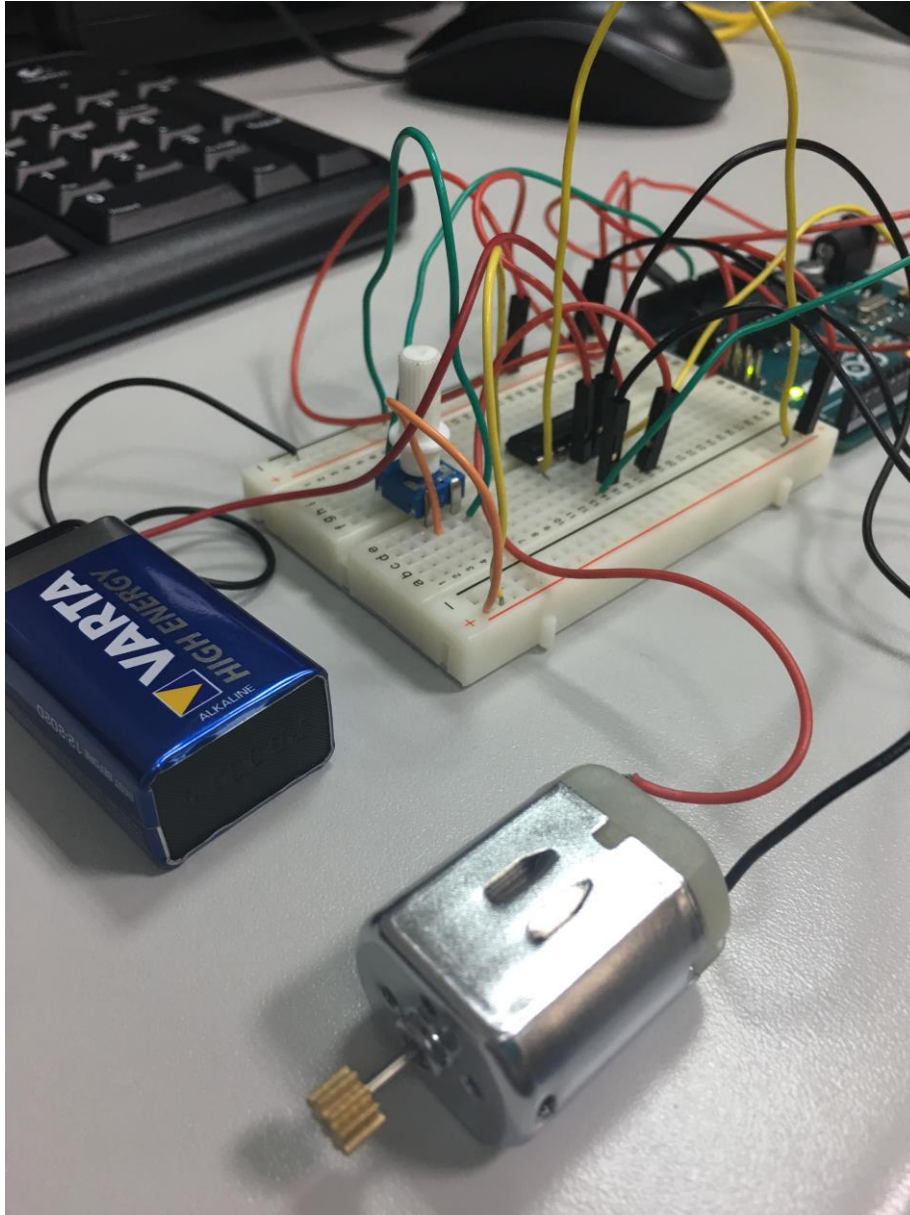
ACTIVIDAD 6

En esta actividad vamos a controlar el ángulo del servo desde el pc, mediante un programa realizado en python, le enviaremos un ángulo entre 0 y 180 y el servo adoptara esa posición

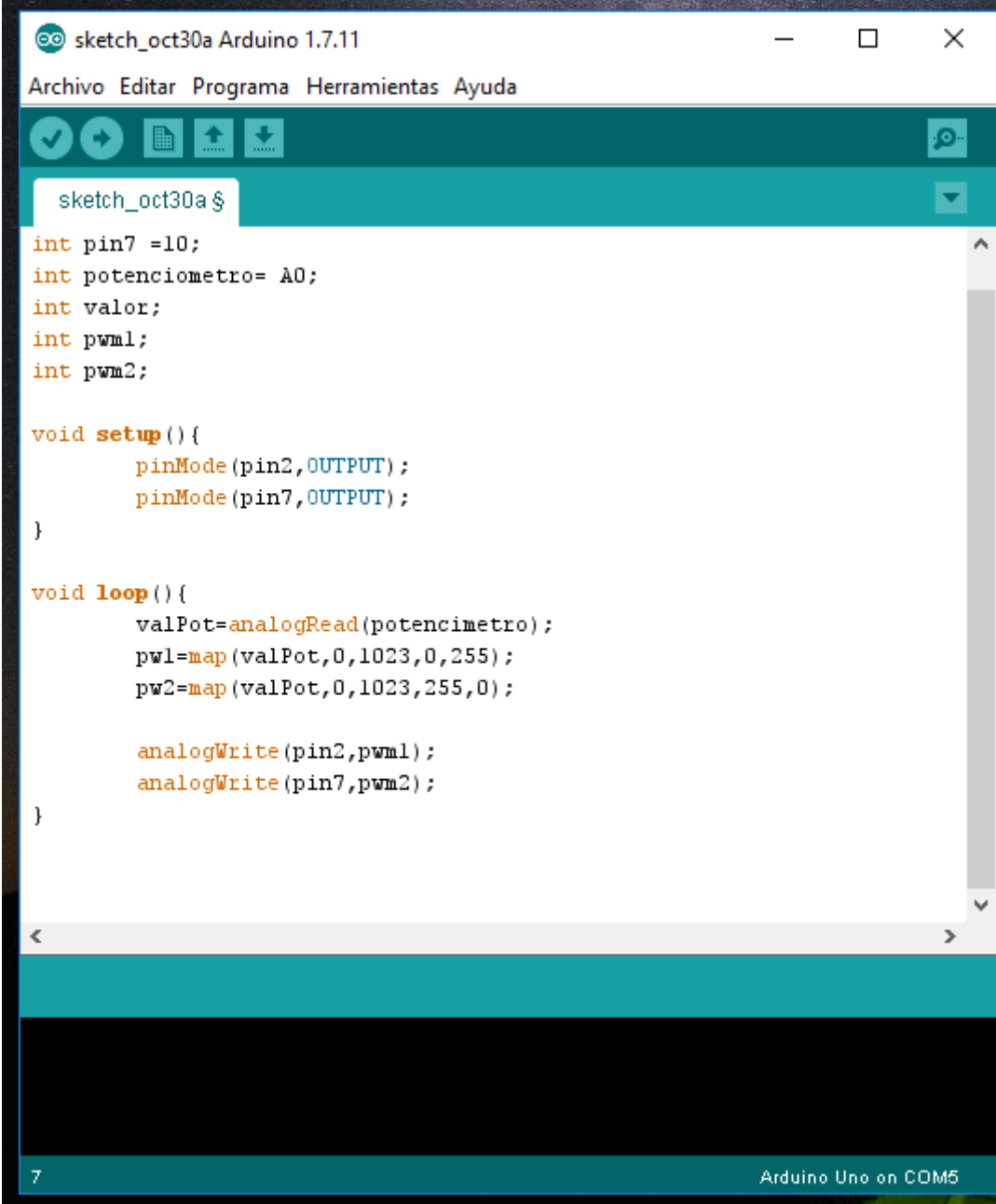
```
1  import serial
2
3  arduino = serial.Serial('/dev/ttyACM0', 9600)
4
5  print("Starting!")
6
7  while True:
8      val = raw_input('Introduce el angulo: ') #Input
9      arduino.write(val) #Mandar un comando hacia Arduino
10
11  arduino.close() #Finalizamos la comunicacion
12
13
14 #include <Servo.h>
15
16 int val = 0; //Variable de entrada del Serial
17
18 Servo servo; //Creamos un objeto Servo de nombre... servo
19
20 void setup()
21 {
22     Serial.begin(9600); //Iniciamos el serial
23     servo.attach(9); //Conectamos el servo al pin digital 3
24 }
25
26 void loop()
27 {
28     if(Serial.available() > 0) //Detecta si hay alguna entrada por serial
29     {
30         val = Serial.parseInt();
31         servo.write(val); //Mueve el servo a la posición entrada (excepto si es 0)
32     }
33     delay(20);
34 }
```

ACTIVIDAD 7

En esta actividad vamos a trabajar con un motor de continua, donde controlaremos el giro y la velocidad del mismo.



- Puente H: es un circuito que permite intercambiar la polaridad en las señales de salida con señales de control de entrada.
- Potenciómetro: es una resistencia variable; componente con tres terminales. La tercera patilla es la resistencia variable, se encuentra de forma mecánica entre VDD y GND.



```
sketch_oct30a Arduino 1.7.11
Archivo Editar Programa Herramientas Ayuda

sketch_oct30a $

int pin7 =10;
int potencimetro= A0;
int valor;
int pwm1;
int pwm2;

void setup(){
    pinMode(pin2,OUTPUT);
    pinMode(pin7,OUTPUT);
}

void loop(){
    valPot=analogRead(potencimetro);
    pwm1=map(valPot,0,1023,0,255);
    pwm2=map(valPot,0,1023,255,0);

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

7 Arduino Uno on COM5

Se aprovechará la capacidad del puente H para usarlo en un motor de corriente continua el cual dependiendo de la polaridad recibida girará en un sentido u en otro.

Evaluación después de la actividad

Esta ha sido más complicada de hacer puesto que desconocida del puente H y cómo funcionaba internamente. También me ha costado más poder hacer el buen mapeo para mover el servo como queríamos.

Incidencias

- Un buen retraso por mala conexión del puente H.

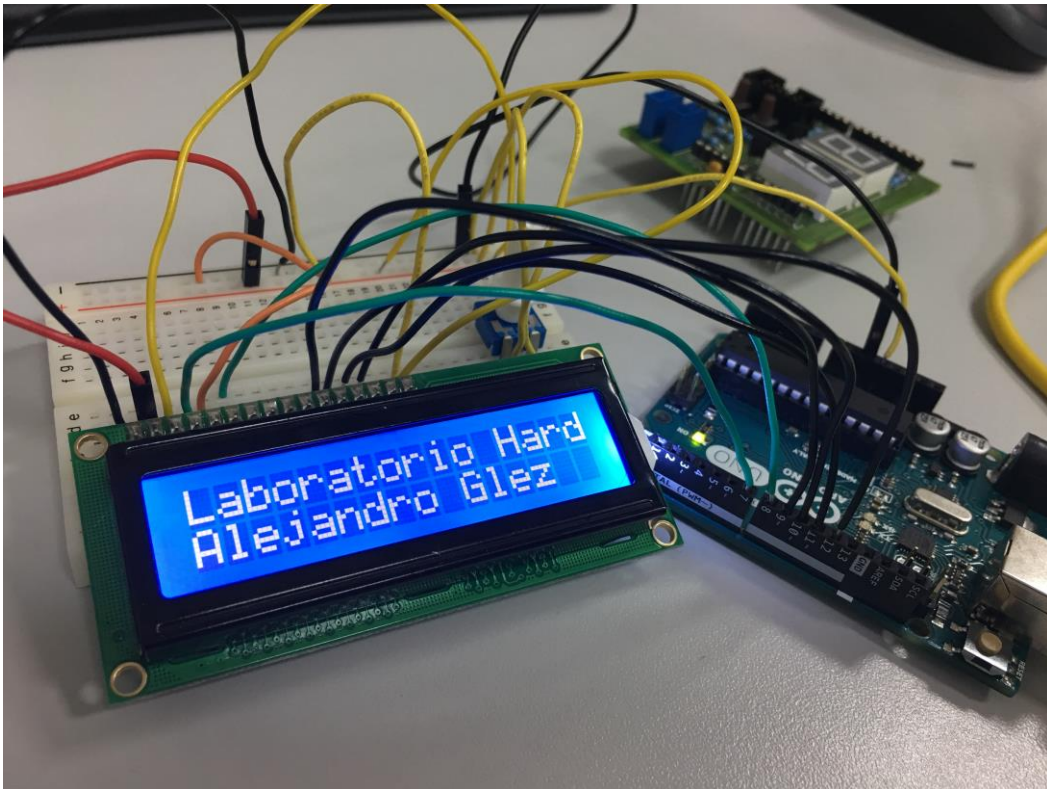
Valoración personal

Me ha gustado poder conocer cómo se podría controlar la dirección de un coche teledirigido mediante este sistema.

ACTIVIDAD 8

En esta nueva actividad vamos a trabajar con la pantalla lcd. En esta primera actividad con la pantalla vamos a controlar la luminosidad de la pantalla que tiene de fondo. Luego pasaremos a:

- Variación 1: Imprimir en dos filas con desplazamiento a la izquierda:
 - - Fila 0: nombre de la asignatura
 - - Fila 1: nombre alumno



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);

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

void loop() {
  lcd.setCursor(0,0);
  lcd.print("Laborario Hard");
  lcd.setCursor(0,1);
  lcd.print("Alejandro Glez");
}
}
```

El

código en Python que hemos usado en esta práctica es el mismo que hemos usado en las anteriores para poder comunicarnos vía serie y poder recolectar los datos.

```
import serial

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

print("Starting!")

while True:
    comando = raw_input('Introduce una frase: ') #Input
    arduino.write(comando) #Mandar un comando hacia Arduino
    print('Comando mandado')

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

Evaluación después de la actividad

Tras un tutorial paso a paso proporcionado por el profesor, hemos conseguido conectar todos los pines de la pantalla lcd y poder usarla.

También hemos aprendido a usar la librería liquidCristal.

Incidencias

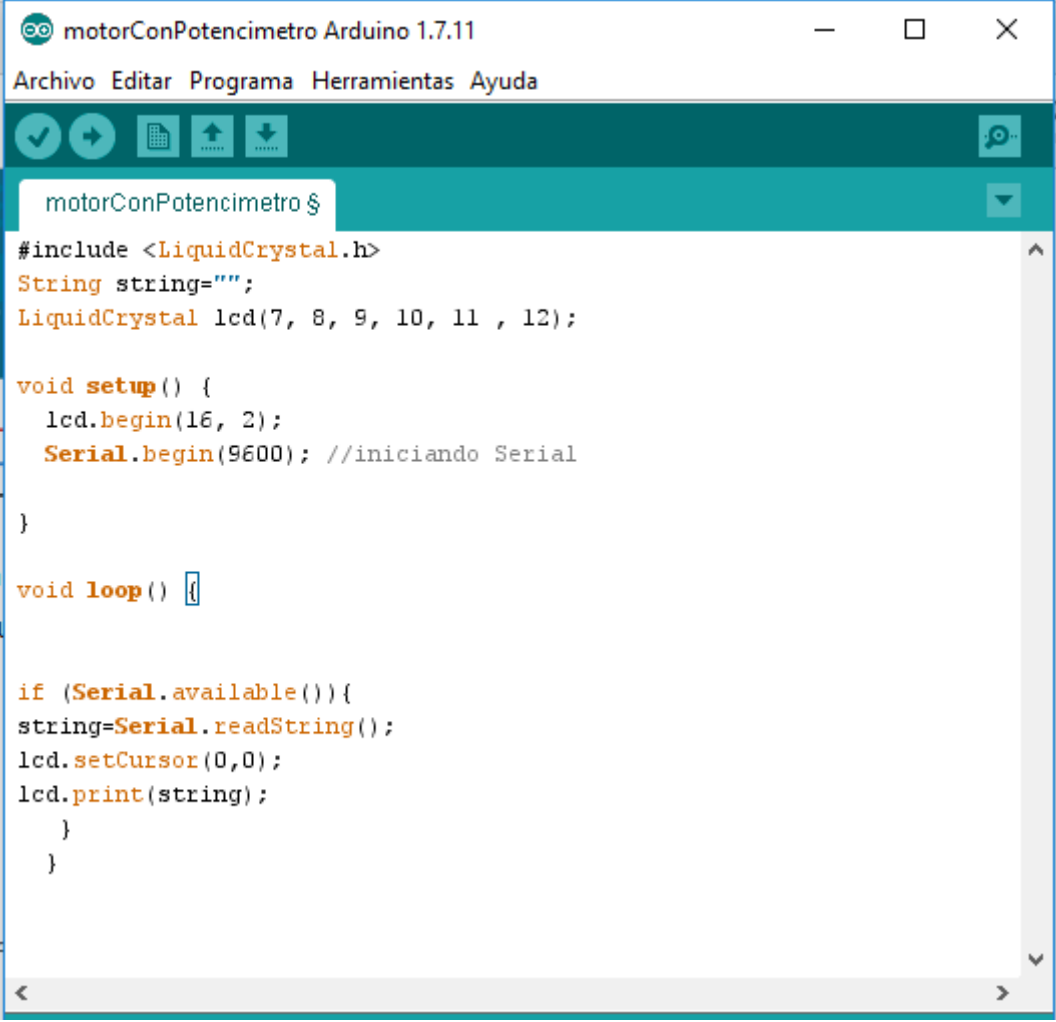
- No poder visualizar bien la pantalla por mal conexión del potenciómetro.
- Mal ajuste de los punteros de la pantalla y mala visualización

Valoración personal

Muy contento y satisfecho con los resultados. Y más cuando llegue a casa y usando esta práctica pude ampliarla un poco más, haciendo un diseño propio basado en lo hecho en clases.

ACTIVIDAD 9

En esta actividad vamos a mostrar por la LCD que está conectada al arduino lo que le introduzcamos mediante un programa creado en python similar a los que antes hemos mostrado.

The image shows the Arduino IDE interface. The title bar reads "motorConPotencimetro Arduino 1.7.11". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". The toolbar contains icons for checking, running, uploading, and downloading. The main text area shows the following C++ code:

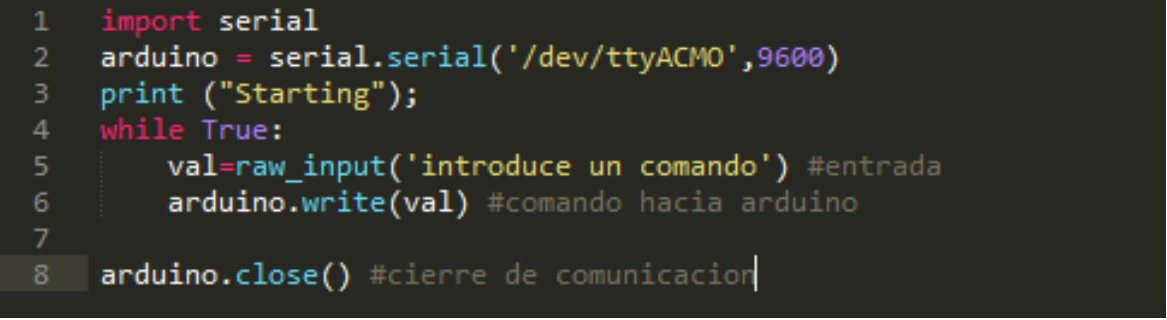
```
motorConPotencimetro $
#include <LiquidCrystal.h>
String string="";
LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);

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

void loop() {

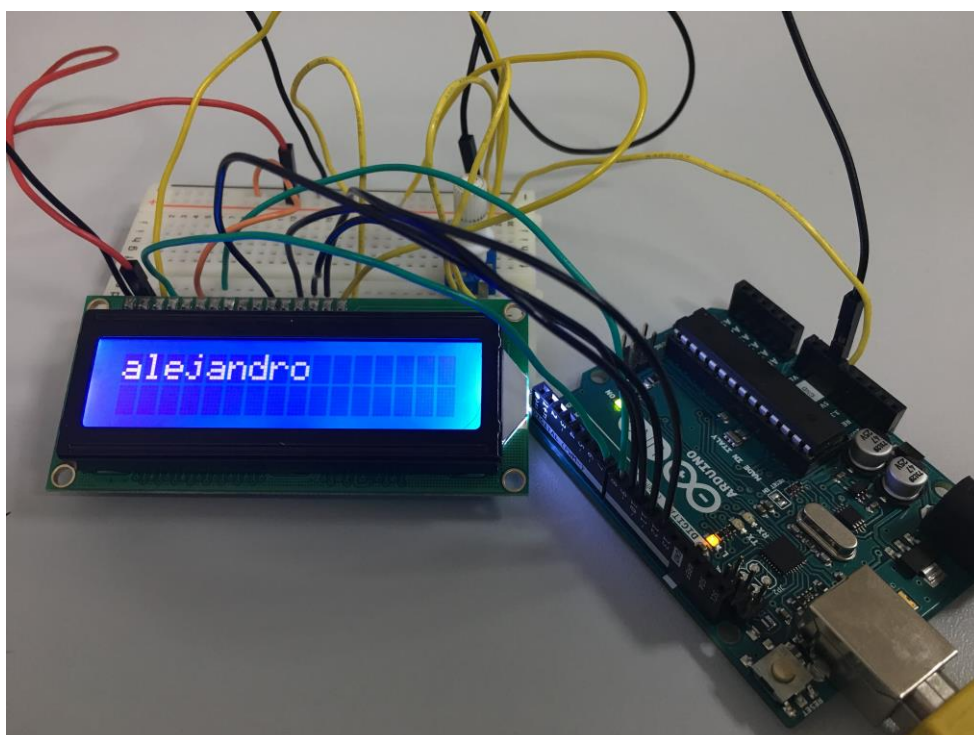
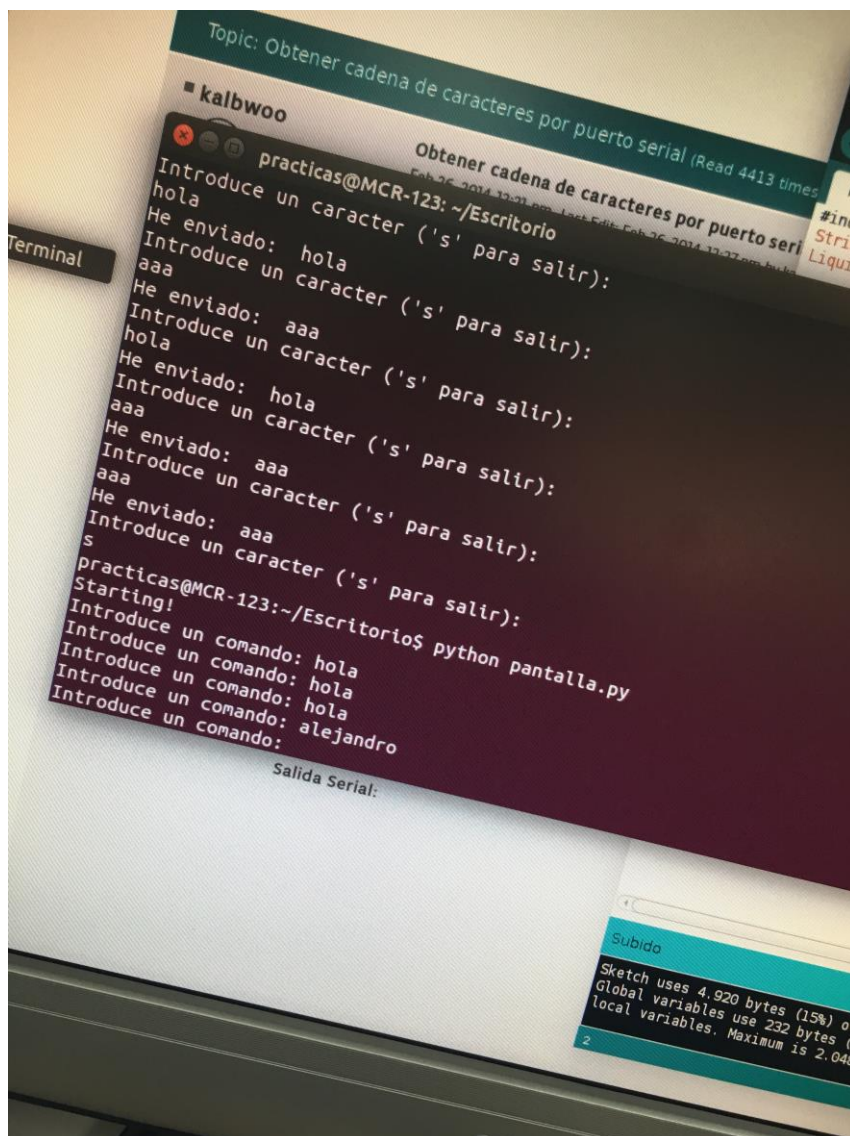
  if (Serial.available()){
    string=Serial.readString();
    lcd.setCursor(0,0);
    lcd.print(string);
  }
}
```

Este es el código en Arduino.

The image shows a Python script in a dark-themed editor. The code is as follows:

```
1 import serial
2 arduino = serial.serial('/dev/ttyACMO',9600)
3 print ("Starting");
4 while True:
5     val=raw_input('introduce un comando') #entrada
6     arduino.write(val) #comando hacia arduino
7
8 arduino.close() #cierre de comunicacion
```

Código en Python para introducir texto.



ACTIVIDAD 10

En esta actividad vamos a realizar un termómetro utilizando la pantalla LCD para mostrar la temperatura y un sensor de temperatura.

-Sensor de temperatura: tiene tres terminales, dos para polarizar y el tercero es para la señal, el funcionamiento que tiene es que varía la tensión según la temperatura.

```
String string="";
LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);
int Sensor = 0 ; // Prog_15_1
int umbral = 25 ;
void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600); //iniciando Serial
  lcd.setCursor(0,1);
  lcd.print("La temperatura:");
}

void loop() {

  int lectura = analogRead(Sensor);
  float voltaje = 5.0 /1024 * lectura ; // Atencion aqui
  float temp = voltaje * 100 -50 ;
  Serial.println(temp) ; delay(1000);
  lcd.setCursor(0,0);
  lcd.print(temp);
}
```

Evaluación después de la actividad

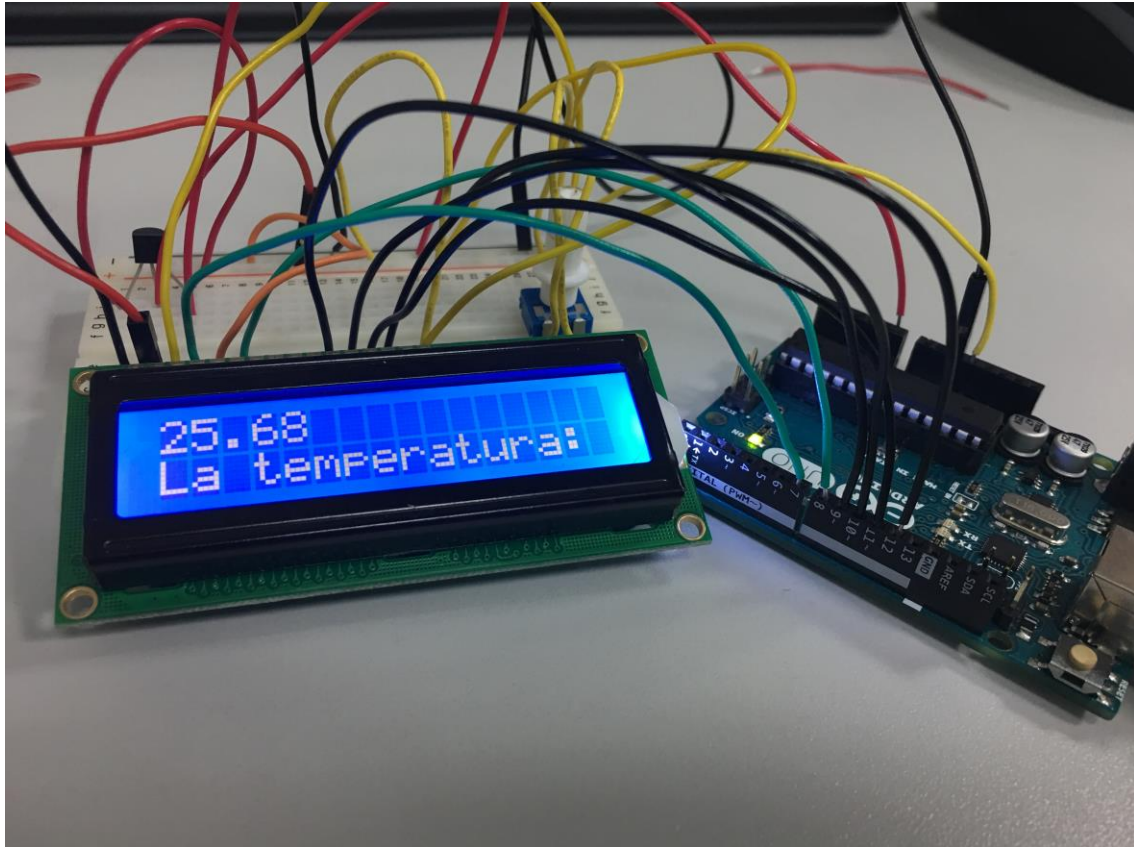
Esta práctica ha sido un poco más compleja puesto que mezclaba varios sistemas que hemos hecho por separado anteriormente. Se trataba de unir el sensor de temperatura, el código python para el puerto serie y el Display. Al final pudimos comprobar el funcionamiento y como mostraba en pantalla la temperatura recogida del sensor que variaba simplemente tocándolo con los dedos y transmitirle calor.

Incidencias

- El mal muestreo del valor del sensor. Al buscar en internet había varios modelos y código según modelo exacto.

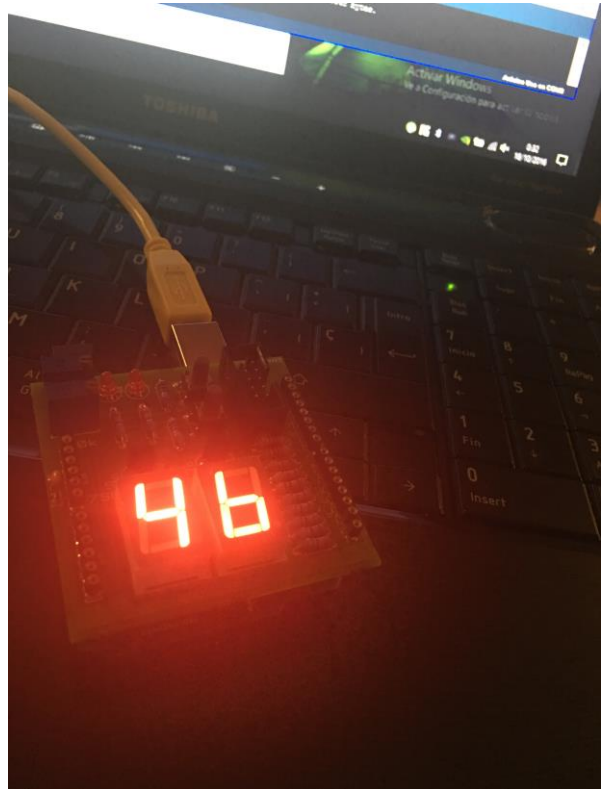
Valoración personal

Práctica muy interesante puesto que combinamos varios sistemas y podría ser útil en algún momento



ACTIVIDAD 11

Contador 00 a 59 en display 7-seg cada segundo



//CODIGO

```
int segPins[] = { 0, 1, 2, 3, 4, 5, 6, 7};

int tiempototal= 1000;
int j = 40;
int displ =8;
int disp2= 9;

int dat1 = 0;
int dat0 = 0;

void setup() {
  for (int thisseg = 0; thisseg < 8; thisseg++) {
    pinMode(segPins[thisseg], OUTPUT);
  }
  pinMode(displ, OUTPUT);
  pinMode(disp2, OUTPUT);
  for(int i=0;i<=9;i++){
    dat1=i;
    for(int j=0;j<=9;j++){
      dat0=j;
    }
  }
  delay(1000);
}

// Función refresh: Duración total de ejecución de refresh: tiempototal.
//Se van intercambiando los displays (displ con dat0 y disp2 con dat1) a una frecuencia que evite el par
void refresh( int datal, 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 ll

}

// 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

void write7seg (unsigned char arg) {
  unsigned char segmen = 0x01;
  unsigned char displayl;
  displayl = arg;

  for (int i = 0; i < 8; i++) {
    if ((displayl & segmen) == 0x00)
      digitalWrite(i, LOW);
    else
      digitalWrite(i, HIGH);

    segmen <<= 1;
  }
}

```


Evaluación después de la actividad

Esta práctica además de larga tenía una complejidad extra puesto que teníamos que usar un Shield de Arduino aparte, aprender cómo funcionaba y hacer una buena conexión. También ha sido más difícil el buen manejo del bus a velocidades altas. Pero lo más complejo ha sido el entendimiento del código dado a medias para terminarlo nosotros.

Incidencias

- Mala visualización
- Visualización de secuencias erróneas.
- Mal uso de los delays

Valoración personal

Compleja pero muy gratificante cuando consigues ver los resultados en pantalla funcionando de forma correcta

CONCLUSIONES

Personalmente pienso que este tipo de práctica es muy buena, porque vemos sistemas que podrían ser reales y resultados en tiempo real. Programamos pensando y vemos a su vez como por ejemplo una bombilla se enciende, o un motor gira 30 grados. Lo cual es una forma atractiva de programar y lo hace más dinámico. Este tipo de prácticas se debería de ver algo en cursos anteriores, porque la gente que entra en esta titulación solo ven asignaturas de matemáticas y algoritmos. Cuando cosas así, prácticas y visuales son las que motivan al alumno. Por otra parte son prácticas muy completas, algunas un poco más difíciles pero en general están muy bien. Me hubiese gustado ver algunas shield mas de arduino, pero entiendo que debido al tiempo que tenemos de práctica no podamos avanzar más en materia de arduino y tengamos que pasar a otras placas.