

MEMORIA BLOQUE 1

PLATAFORMA ARDUINO Y RASPBERRY PI



José Gallardo Harillo

30/09/2022

Indice:

Objetivos.....	3
Introducción teórica	3
Plataforma Arduino.....	3
Componentes hardware/electrónicos básicos.....	6
Plataforma Raspberry pi.....	13
Ejercicios prácticos	14
1: Control de encendido-apagado mediante un relé desde el PC.	14
Conexión Hardware.....	14
Código.....	14
Resultado.....	16
2: Control de ángulo de un servo desde el PC.	16
Conexión Hardware.....	16
Código.....	17
Resultado.....	18
3: Imprimir la información que envía el PC vía serie al display LCD.....	18
Conexión Hardware.....	18
Código.....	19
Resultado.....	20
4: Termómetro usando tmp36GZ que imprime el valor por el display LCD.	21
Conexión Hardware.....	21
Código.....	22
Resultado.....	23
5: Mediante attachInterrupt(), hacer que mediante un switch/pulsador se imprima por el PC vía serie los datos recogidos por el LDR (valores del divisor de tensión y valor nominal).....	23
Conexión Hardware.....	23
Código.....	24
Resultado.....	25
6: Hacer que el arduino entre en power_down, y cuando reciba un byte se despierte y mande al PC vía serie los valores del LDR mencionados en el apartado anterior.	26
Conexión Hardware.....	26
Código.....	27
Resultado.....	29
7: Conectarse desde el PC a la Raspberry pi mediante SSH.....	30

Configuración previa	30
SSH.....	32
8: EXTRA: infrarrojos	34
Conexión Hardware.....	34
Código.....	35
Resultado.....	40
Conclusión.....	40

Objetivos

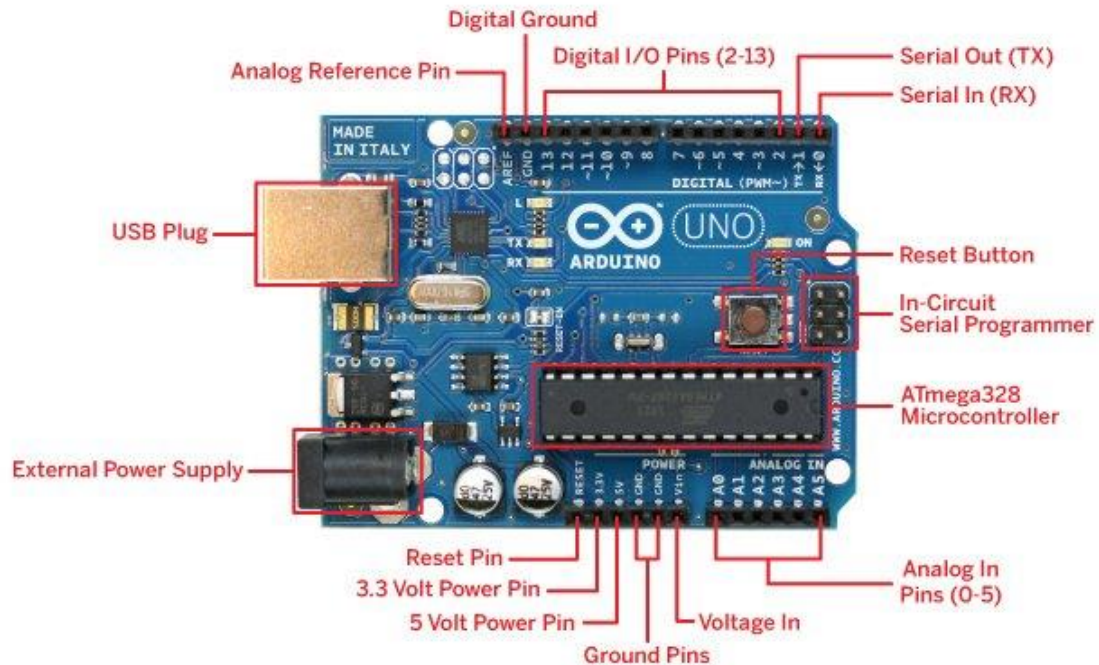
1. Conocer la plataforma Arduino (Características, variantes, modos de programación).
2. Conocer varios de los componentes básicos de hardware usados en los sistemas empujados.
3. Preparar el PC para programar en el entorno de Arduino.
4. Realizar ejemplos de funcionamiento sobre Arduino.
5. Preparar la plataforma Raspberry Pi /Orange Pi para que puedan cargarse diferentes versiones de Sistema Operativo.
6. Conocer, arrancar y comprobar el funcionamiento de la placa Raspberry Pi/Orange Pi.

Introducción teórica

Plataforma Arduino

Arduino es una plataforma de desarrollo de hardware abierta (open Hardware) basada en software y hardware flexibles y fáciles de usar. Arduino puede tomar información del entorno a través de sus pines de

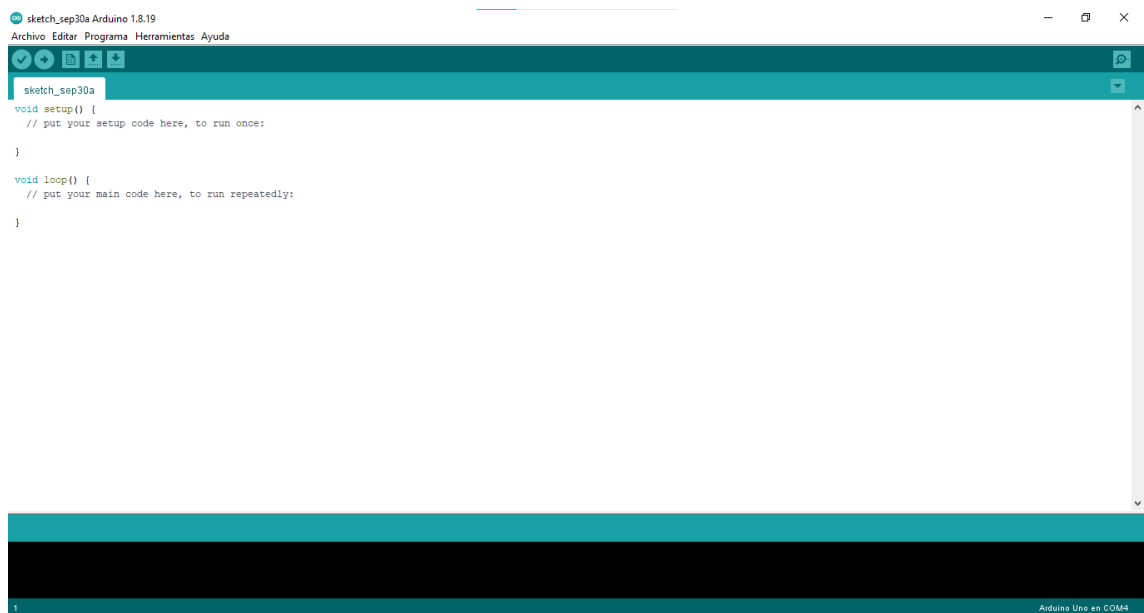
entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros tipos de actuadores.



Entre sus partes más destacables para los ejercicios a realizar tenemos:

1. 14 pines digitales, donde el 0 y el 1 son pines para el envío-recepción serie.
2. 6 pines analógicos, donde A4 también es entrada SDA, y A5 entrada SCL.
3. 3 pines a tierra.
4. 2 pines de alimentación (3.3 y 5 V)
5. Conexión USB tipo B

Su entorno de desarrollo puede ser descargado fácilmente a través de la página web oficial de Arduino, lista para ser utilizada quitando la actualización/instalación de drivers de la placa. Principalmente se programa por c/c++, pero puede ser programable con otros lenguajes.



Estructura general de un programa:

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

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

Componentes hardware/electrónicos básicos

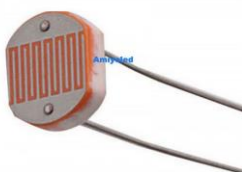
1. Resistencia: Componente usado para reducir el voltaje en caso de estar alimentando un componente que necesite una cantidad menor de corriente que la que proporciona la fuente.



2. Potenciómetro: Resistencia variable, 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.



3. Fotorresistencia (LDR): Componente biterminal que varía su resistencia con la luz.



4. Diodo: Componente electrónico que hace pasar la corriente si se supera un valor umbral de corriente.



5. Led: Diodo emisor de luz hecho para iluminarse cuando pasa la corriente, variando la luminosidad con respecto la corriente que pase.



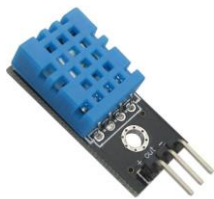
6. Transistor: Dispositivo electrónico semiconductor. Permite el paso de una señal en respuesta a otra. Se puede configurar o "comportar" como amplificador, oscilador, conmutador o rectificador.



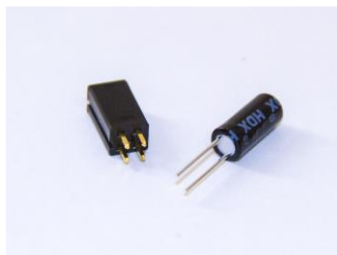
7. Sensor de temperatura (LM36GZ): Posee tres terminales; Dos, de alimentación y el tercero de señal. Varía la tensión en función de la temperatura (Funcionamiento similar al transistor).



8. Sensor de temperatura y humedad (DHT 11, 22): Posee tres terminales; Dos, de alimentación y el tercero de señal. Señal digital con un protocolo de comunicación (existe librería DHT en arduino), donde se puede sacar la temperatura y humedad captada.



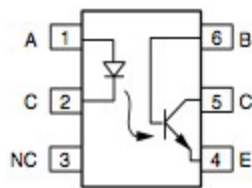
9. Sensor de inclinación: Se trata de una bola de metal interconecta a dos terminales en función de la inclinación del componente.



10. Switch/pulsador: Componentes que cambian de estado según el estado físico que tengan (encendido/apagado).

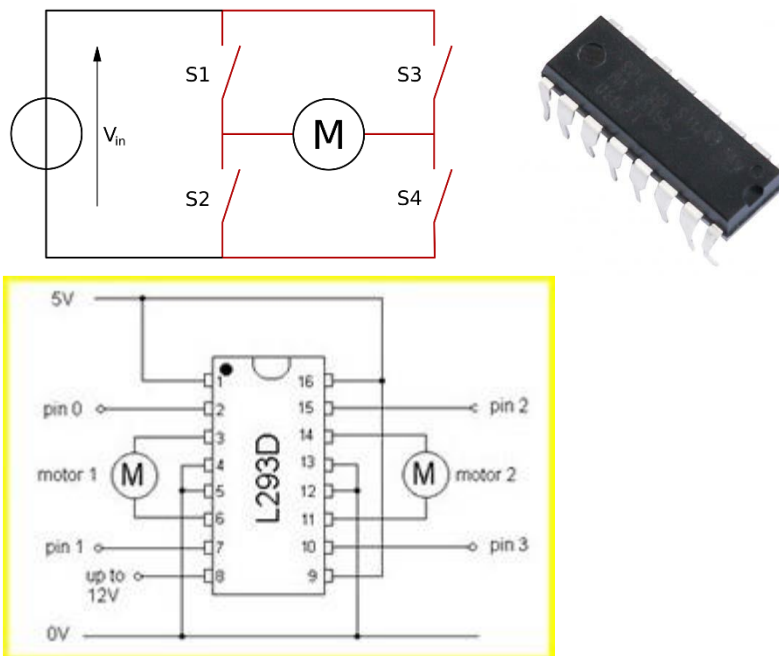


11. Optoacoplador: permite activar/desactivar un switch de un circuito desde otro circuito separado (fuentes de energía distintas)



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

Ej: L293D: Circuito integrado que incorpora dos puentes H



13. 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)



14. Motores (desde el punto de vista del microcontrolador):

- 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).



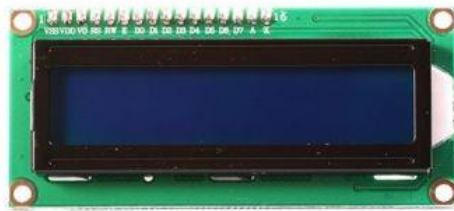
- Servo motor: básicamente un motor DC con circuitería de control ya incluida. La función del servomotor es girar hasta colocarse en un ángulo determinado y mantenerse en esa posición mientras se desee. Suelen tener un ángulo de giro de 0 a 180 grados. El ángulo de giro en el que se posicionan depende 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.



- Motor paso a paso: Son motores que giran un determinado ángulo por cada impulso eléctrico. El ángulo puede ser grande (90 grados) o pequeño (1,8 grados). Típicamente se controlan con puentes H.

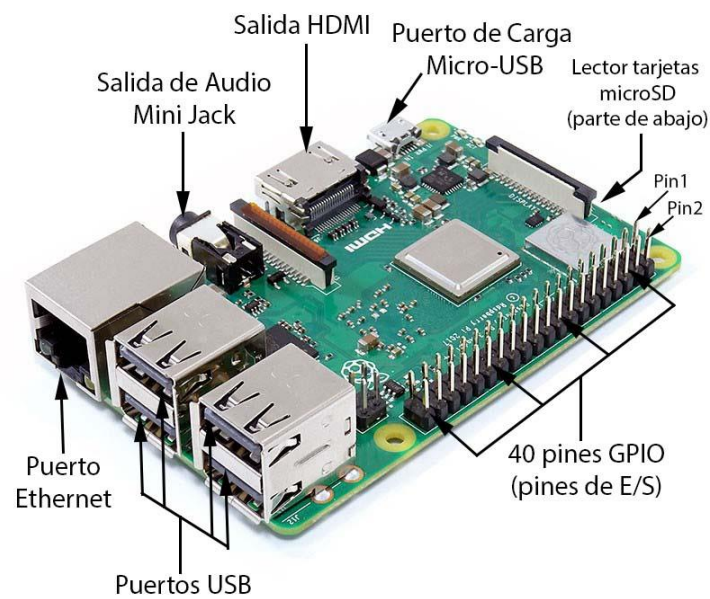


- 15.LCD: Pantalla de cristal líquido donde pueden imprimirse mensajes. Esta puede utilizar un adaptador I2C para facilitar su implementación.



Plataforma Raspberry pi

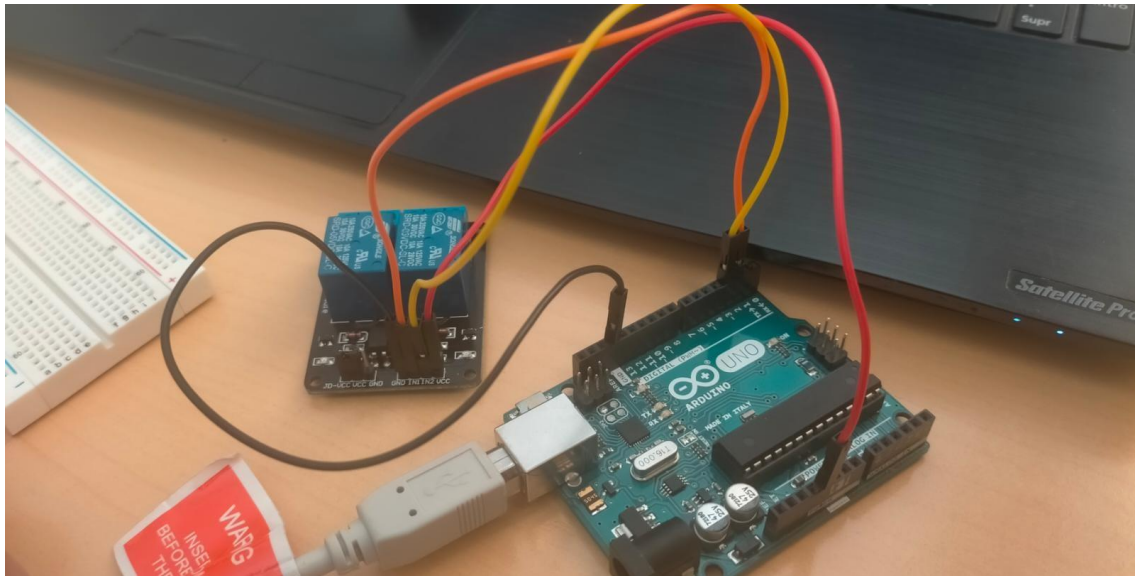
La Raspberry pi es un ordenador de placa simple y bajo consumo que, al igual que Arduino, tiene como objetivo principal estimular la motivación de los niños a adentrarse en el mundo de la programación y las ciencias de la computación. En el siguiente bloque, utilizaremos esta como elemento principal del Nodo Gateway, siendo esta la que reciba toda la información del exterior y de las ordenes en respuesta a esa información proporcionada (dicho a grosso modo, se verá con más detalle en el bloque 2).



Ejercicios prácticos

1: Control de encendido-apagado mediante un relé desde el PC.

Conexión Hardware



IN1 es relé 1 e IN2 es relé 2

Código

Phyton

```
import serial #Importamos serial para realizar la comunicación serie con Arduino desde visual studio
Arduino = serial.Serial('COM4', 9600) #Puerto serie

while True:
    val = input("Introduce tecla por favor: ")
    if len(val) >= 1: #impedir problemas con los inputs vacíos
        Arduino.write(val.encode()) #El valor introducido en val (en nuestro contexto una tecla),
                                   #es transmitido por el puerto serie Arduino
        print(Arduino.readline().decode()) #Imprime lo que recibe el puerto serie del PC
```

Arduino

```
//En mi caso poseo una placa con 2 relés,
//para usar cada uno asigno los siguientes pines
int rele = 2;
int rele2 = 3;

//Variable que almacenará la información
//recibida mediante puerto serie
int tecla;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(rele, OUTPUT);
  pinMode(rele2, OUTPUT);

  //Ponemos el estado de los relés en alto para que se
  //inicialicen apagados en caso de que en el uso anterior
  //se haya dejado algún relé encendido
  digitalWrite(rele, HIGH);
  digitalWrite(rele2, HIGH);
}

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

  if (Serial.available()>0){ //Si recibe algún byte desde puerto serie
    tecla=Serial.read(); //Lectura del puerto serie

    if(tecla=='1'){ //Si se ha pulsado la tecla 1
      digitalWrite(rele, LOW); //Enciende relé 1
      Serial.println("rele 1 encendido");
    }

    if(tecla=='0'){ //Si se ha pulsado la tecla 0
      digitalWrite(rele, HIGH); //Apaga relé 1
      Serial.println("rele 1 apagado");
    }

    if(tecla=='3'){ //Si se ha pulsado la tecla 3
      digitalWrite(rele2, LOW); //Enciende relé 2
      Serial.println("rele 2 encendido");
    }

    if(tecla=='2'){ //Si se ha pulsado la tecla 2
      digitalWrite(rele2, HIGH); //Apaga relé 2
      Serial.println("rele 2 apagado");
    }
  }
}
```

Resultado

```
C:\Users\PC\AppData\Local\Programs\Python\Python310\python.exe
Introduce tecla por favor: 1
rele 1 encendido

Introduce tecla por favor: 0
rele 1 apagado

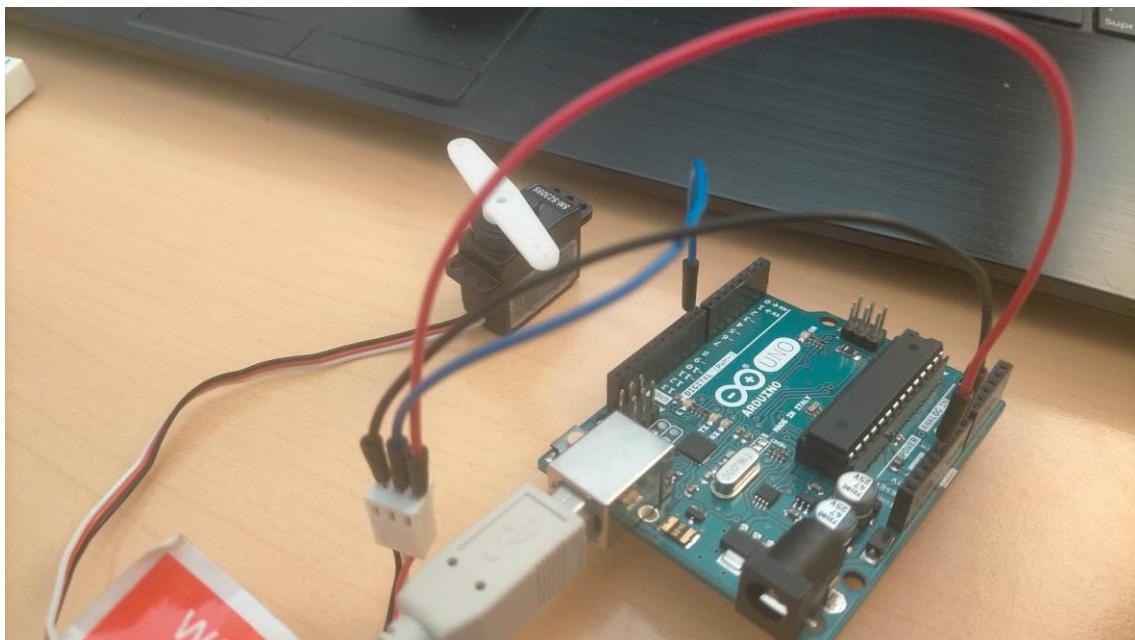
Introduce tecla por favor: 2
rele 2 apagado

Introduce tecla por favor: 3
rele 2 encendido
```

A nivel físico realizan la acción indicada.

2: Control de ángulo de un servo desde el PC.

Conexión Hardware



Código

Phyton

(El código es exactamente el mismo que en el caso anterior)

Arduino

```
#include <Servo.h>
|
Servo myservo; // objeto servo

//Variable que almacenará la información
//recibida mediante puerto serie
int tecla;

void setup() {
  Serial.begin(9600);
  myservo.attach(9); // Pin 9 asignado al servo
}

void loop() {

  if (Serial.available()>0){//Si recibe algún byte desde puerto serie

    tecla = Serial.parseInt(); //Leemos y parseamos a entero el valor recibido para usarlo en el servo
    myservo.write(tecla); //Indicamos al servo a qué grados se tiene que poner

    Serial.println(tecla);
  }

}
```

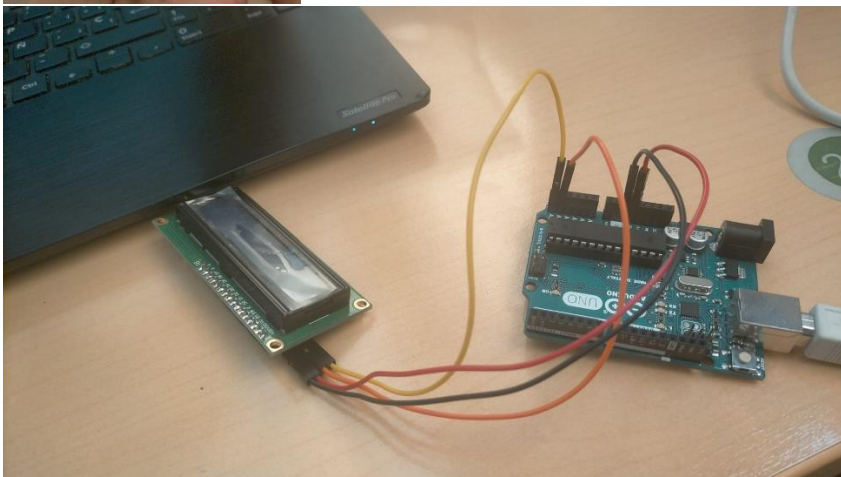
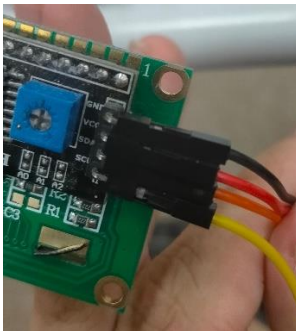
Resultado

```
C:\Users\PC\AppData\Local\Programs\Python\Python310\python.exe
Introduce tecla por favor: 89
89
Introduce tecla por favor: 180
180
Introduce tecla por favor: 0
0
Introduce tecla por favor: 90
90
Introduce tecla por favor: 180
180
Introduce tecla por favor: 0
0
Introduce tecla por favor: r
0
Introduce tecla por favor: k
0
```

Si metemos una tecla que no sea un número sencillamente se pone a 0º.

3: Imprimir la información que envía el PC vía serie al display LCD.

Conexión Hardware



Código

Phyton

```
import serial #Importamos serial para realizar la comunicación serie con Arduino desde visual studio
Arduino = serial.Serial('COM4', 9600) #Puerto serie

while True:
    val = input("Introduce tecla por favor: ")
    if len(val) >= 1: #impedir problemas con los inputs vacíos
        Arduino.write(val.encode()) #El valor introducido en val (en nuestro contexto una tecla),
                                   #es transmitido por el puerto serie Arduino
```

El mismo código que los anteriores, pero en este caso no se recibe nada por el puerto serie del PC.

Arduino

```
#include <LiquidCrystal_I2C.h>

//Crear el objeto lcd dirección 0x3F y 16 columnas x 2 filas
LiquidCrystal_I2C lcd(0x3F,16,2);

String mensaje; //Mensaje que se deberá imprimir por el LCD

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

    Serial.begin(9600);

    // Inicializar el LCD
    lcd.init();

    //Encender la luz de fondo.
    lcd.backlight();
}
```

```

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

    if (Serial.available()>0){ //Si recibe algún byte desde puerto serie
        lcd.clear(); //Recoloca el LCD de paso a la columna 0

        mensaje = Serial.readStringUntil('\n'); //Leemos el mensaje escrito vía serie

        String fila = mensaje.substring(0,16); //fila 1

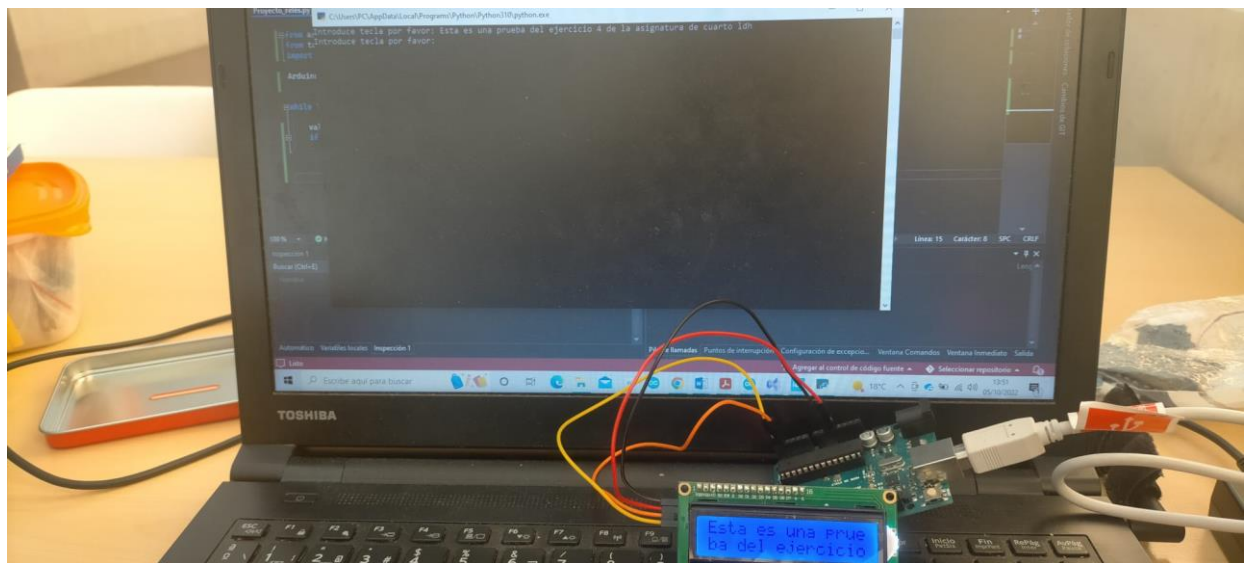
        lcd.print(letra);

        lcd.setCursor(0,1);

        fila = mensaje.substring(16,32); //fila 2
        lcd.print(letra);
    }
}

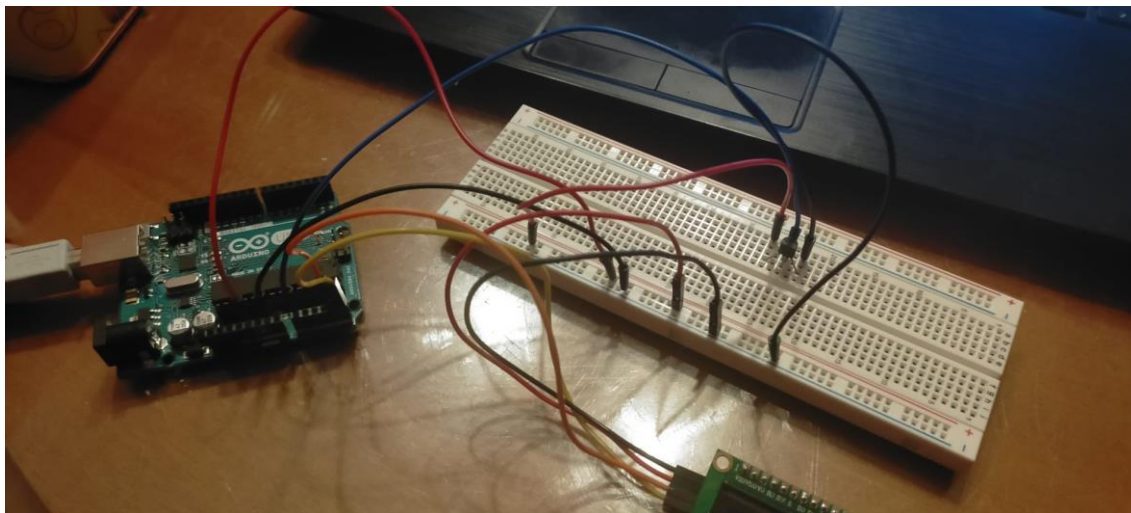
```

Resultado



4: Termómetro usando tmp36GZ que imprime el valor por el display LCD.

Conexión Hardware



Código

Arduino

```
#include <LiquidCrystal_I2C.h>

//Crear el objeto lcd dirección 0x3F y 16 columnas x 2 filas
LiquidCrystal_I2C lcd(0x3F,16,2);

int mensaje; //Mensaje que se deberá imprimir por el LCD

//Pin para el sensor
#define sensorPin A0

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

  Serial.begin(9600);

  // Inicializar el LCD
  lcd.init();

  //Encender la luz de fondo.
  lcd.backlight();
}

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

  // Lectura del sensor:
  int lectura = analogRead(sensorPin);

  // Convertir a voltios:
  float voltios = (lectura * 5.0)/1024.0;

  // De voltios a celcius:

  float temperatura = (voltios - 0.5) * 100;

  lcd.clear();
  lcd.print(temperatura);
  lcd.print((char) 223); // grados
  lcd.print("C");

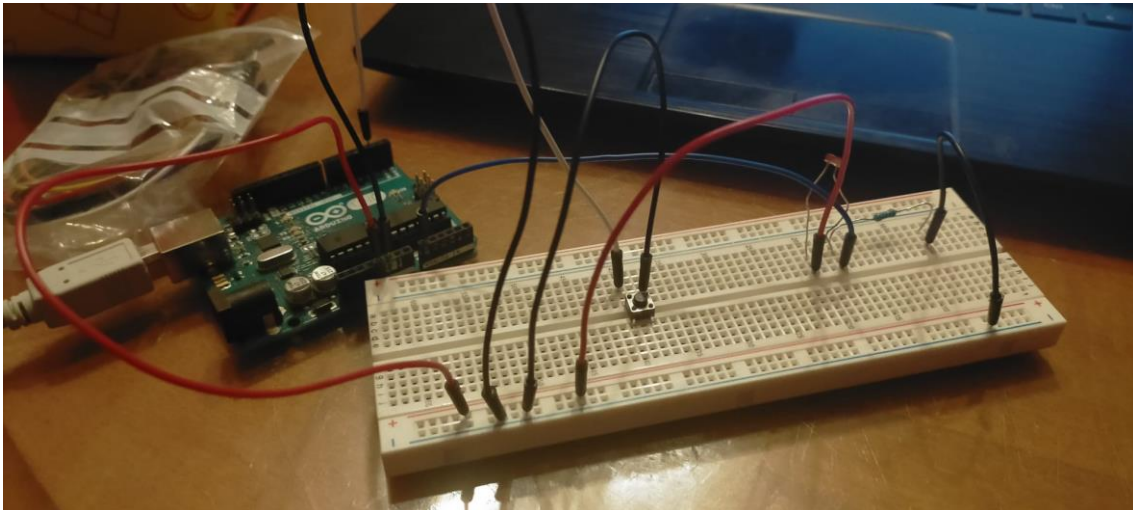
  delay(1000);
}
```

Resultado



5: Mediante attachInterrupt(), hacer que mediante un switch/pulsador se imprima por el PC vía serie los datos recogidos por el LDR (valores del divisor de tensión y valor nominal).

Conexión Hardware



Código

Phyton

```
import serial #Importamos serial para realizar la comunicación serie con Arduino desde visual studio

Arduino = serial.Serial('COM4', 9600) #Puerto serie

while True:
    print (Arduino.readline().decode()) #Imprime lo que recibe el puerto serie del PC
```

Mismo código que el primer ejercicio, pero en este caso solo se recibe información del puerto serie.

Arduino

```
int pinInter = 2; //El pin 2 posee interrupción,
//por lo que le vamos a asignar la entrada digital
//del botón este pin para que salte la interrupción cuando se pulse

int pinLDR = A0; //Pin asignado al terminal analógico del LDR

//Variables a rellenar
float lecturaLDR;
float ldrVoltage;
float resistorVoltage;
float valorNominalLDR;

//Anti-Rebote

long startTime = 0;
const int timeWait = 150;

//Valores constantes
#define MAX_ADC_READING 1023
#define ADC_REF_VOLTAGE 5.0
#define REF_RESISTANCE 10000 //Nuestra resistencia

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

    Serial.begin(9600);

    pinMode(pinInter, INPUT_PULLUP); //Cuando no esta pulsado está a 1 (no pulsado)
    pinMode(pinLDR, INPUT);
    attachInterrupt(digitalPinToInterrupt(pinInter), muestreo, FALLING); //Activación de la
    //interrupción en el pin 2, saltará cuando el pulsador está a 0 (pulsado)
}

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



```

void muestreo(){

    if(millis()-startTime > timeWait){ //Si no se mantiene un tiempo de
        //espera entre pulsación y pulsación

        detachInterrupt(digitalPinToInterrupt(pinInter)); //Desactiva la interrupción
        //para impedir conflictos
        |
        lecturaLDR = analogRead(pinLDR); //Lectura de LDR

        resistorVoltage = (float) lecturaLDR/MAX_ADC_READING * ADC_REF_VOLTAGE; //Pasamos la lectura a un
        //valor entre 0 y 5V

        ldrVoltage = ADC_REF_VOLTAGE - resistorVoltage;

        valorNominalLDR = ldrVoltage/ resistorVoltage * REF_RESISTANCE;

        Serial.println("-----");
        Serial.print("Voltaje de resistencia: ");
        Serial.println(resistorVoltage);

        Serial.print("Voltaje en ldr: ");
        Serial.println(ldrVoltage);

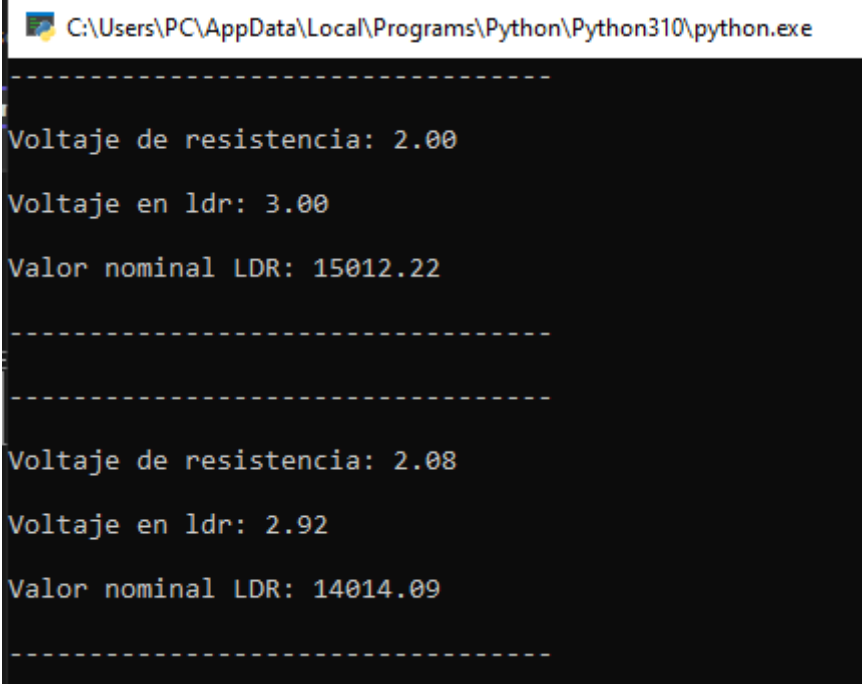
        Serial.print("Valor nominal LDR: ");
        Serial.println(valorNominalLDR);
        Serial.println("-----");

        startTime = millis(); //El tiempo pasado se captura para la condición de anti-rebote

        attachInterrupt(digitalPinToInterrupt(pinInter), muestreo, FALLING); //Activa la interrupción
        //de nuevo para que pueda volver a ser usada
    }
}

```

Resultado



```

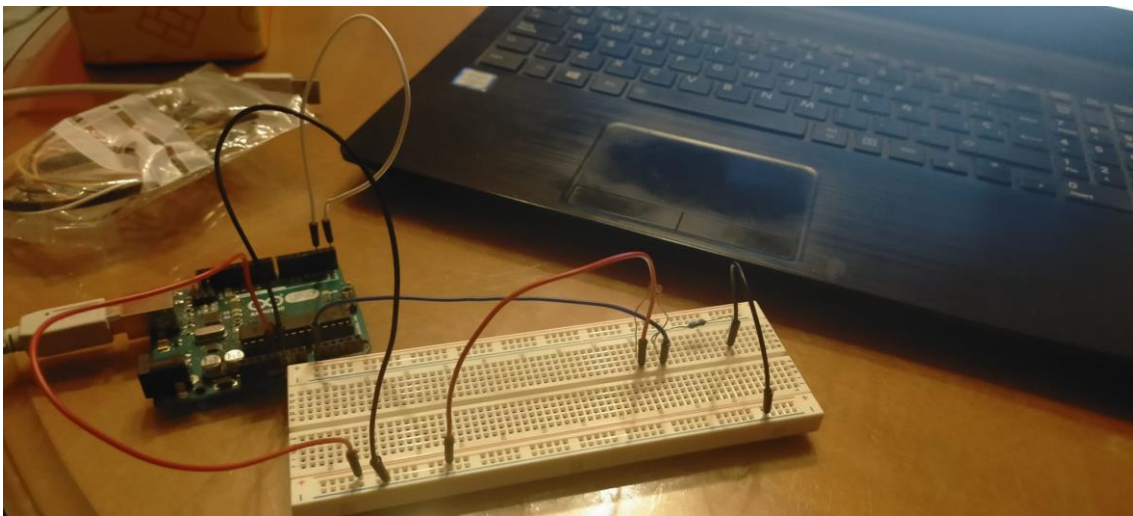
C:\Users\PC\AppData\Local\Programs\Python\Python310\python.exe
-----
Voltaje de resistencia: 2.00
Voltaje en ldr: 3.00
Valor nominal LDR: 15012.22
-----
-----
Voltaje de resistencia: 2.08
Voltaje en ldr: 2.92
Valor nominal LDR: 14014.09
-----

```

Solo cuando pulso el botón (key down) se imprime la información del LDR, en la imagen lo he pulsado dos veces.

6: Hacer que el arduino entre en power_down, y cuando reciba un byte se despierte y mande al PC vía serie los valores del LDR mencionados en el apartado anterior.

Conexión Hardware



La única modificación que hay que hacerle al montaje anterior es quitarle el botón y poner el cable que anteriormente leía del pulsador esta vez al pin 0, que es el pin RX, osea, recepción serie.

Código

Phyton

```
import serial #Importamos serial para realizar la comunicación serie con Arduino desde visual studio

Arduino = serial.Serial('COM4', 9600) #Puerto serie

while True:

    val = input("Introduce tecla por favor: ")

    if len(val) >= 1: #impedir problemas con los inputs vacíos

        Arduino.write(val.encode())#El valor introducido en val (en nuestro contexto una tecla),
                                #es transmitido por el puerto serie Arduino

        #Imprimen lo que recibe el puerto serie del PC
        print (Arduino.readline().decode())
        print (Arduino.readline().decode())
        print (Arduino.readline().decode())
        print (Arduino.readline().decode())
        print (Arduino.readline().decode())
```

Mismo código que el primer ejercicio, pero en este caso la impresión se hace varias veces para albergar los mensajes pasados vía serie.

Arduino

```
#include <LowPower.h>

int pinInter = 2; //El pin 2 posee interrupción,
//por lo que le vamos a asignar la entrada digital
//del botón este pin para que salte la interrupción cuando se pulse

int pinLDR = A0; //Pin asignado al terminal analógico del LDR

//Variables a rellenar
float lecturaLDR;
float ldrVoltage;
float resistorVoltage;
float valorNominalLDR;

//Valores constantes
#define MAX_ADC_READING 1023
#define ADC_REF_VOLTAGE 5.0
#define REF_RESISTANCE 10000 //Nuestra resistencia

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

    Serial.begin(9600);

    pinMode(pinInter, INPUT);
    pinMode(pinLDR, INPUT);
    attachInterrupt(digitalPinToInterrupt(pinInter), muestreo, FALLING); //Activación de la interrupción en el pin 2,
    //saltará cuando se envíe el primer byte de una transmisión serie dirección PC-Arduino
}
```

```

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

  //Ponemos el Arduino en Power Down

  LowPower.powerDown(SLEEP_FOREVER , ADC_ON , BOD_OFF); //El Arduino estará dormido para menor consumo,
  //con el conversor ASC activado y con el procesador a disposición de resetearlo en caso de que la
  //potencia que le llegue sea muy baja

}

void muestreo(){

  detachInterrupt(digitalPinToInterrupt(pinInter)); //Desactiva la interrupción
  //para impedir conflictos

  lecturaLDR = analogRead(pinLDR); //Lectura del LDR

  resistorVoltage = (float) lecturaLDR/MAX_ADC_READING * ADC_REF_VOLTAGE; //Pasamos la lectura a un valor entre 0 y 5V

  ldrVoltage = ADC_REF_VOLTAGE - resistorVoltage;

  valorNominalLDR = ldrVoltage/ resistorVoltage * REF_RESISTANCE;

  Serial.println("-----");
  Serial.print("Voltaje de resistencia: ");
  Serial.println(resistorVoltage);

  Serial.print("Voltaje en ldr: ");
  Serial.println(ldrVoltage);

  Serial.print("Valor nominal LDR: ");
  Serial.println(valorNominalLDR);
  Serial.println("-----");
  Serial.flush();

  attachInterrupt(digitalPinToInterrupt(pinInter), muestreo, FALLING); //Activa la interrupción
  //de nuevo para que pueda volver a ser usada

}

```

Resultado

```
C:\Users\PC\AppData\Local\Programs\Python\Python310\python.exe
Introduce tecla por favor: s
-----
Voltaje de resistencia: 2.18
Voltaje en ldr: 2.82
Valor nominal LDR: 12937.22
-----
Introduce tecla por favor: sdfghj
-----
Voltaje de resistencia: 2.09
Voltaje en ldr: 2.91
Valor nominal LDR: 13901.87
-----
Introduce tecla por favor: supercalifragilisticoexpialidoso
-----
Voltaje de resistencia: 2.05
Voltaje en ldr: 2.95
Valor nominal LDR: 14415.27
-----
Introduce tecla por favor:
```

Da igual cuán grande sea el texto que mandemos, solo saltará una vez.

7: Conectarse desde el PC a la Raspberry pi mediante SSH.

Configuración previa

Previamente, vamos a configurar la RaspBerry pi para que funcione correctamente, para así abordar la conexión SSH.

Instalación del SO de la RaspBerry pi

Para la configuración de la Raspberry pi (en mi caso es la que tengo), en primer lugar, vamos a instalar el sistema operativo, que lo haremos descargando el “Raspberry imager” de la página oficial:

<https://www.raspberrypi.org/software/>

y utilizando la imagen suministrada por el profesor en:

<https://coria.dte.us.es/~bellido/>

de esta forma, pasamos la imagen a la tarjeta micro sd de la placa.

Conexión física inicial de la RaspBerry pi

Hecho todo ello, ahora conectaremos un monitor vía HDMI-VGA/DVI/HDMI (Placa-Monitor), seguido de un teclado y un ratón para poder configurar en sí la placa (El ratón realmente no es necesario, o por lo menos a mí no me ha hecho falta). Finalmente, a nivel físico, conectamos el cable Ethernet (LAN), y el cable de alimentación (recomendable conectarla a alimentación directa desde regleta y no al portátil, si no, saltará cada dos por tres en la terminal de la Raspberry un mensaje de error “under voltage”).

Ahora, habiendo puesto el monitor con la conexión a la que hayas conectado la placa, podremos ver la terminal de la Raspberry, la cuál te pedirá usuario (pi) y contraseña (raspberry).

Conexión Ethernet

Entrando ya en la configuración, primero vamos a asignarle una dirección IP estática ethernet para poder conectarnos desde nuestro PC a esta vía SSH.

Ponemos el comando “`$ sudo nano /etc/dhcpd.conf`”, y descomentamos las líneas relacionadas con “interface eth0” (exceptuando la línea IPv6), y modificamos la información de la dirección IP estática, la puerta de enlace predeterminada, y los servidores DNS a disposición.

En mi caso, dentro del laboratorio, mi dirección IP ha sido la 10.1.15.89/24 (Estoy en el puesto 89), con puerta de enlace 10.1.15.78/24, y con servidor DNS 8.8.8.8.

Para poder avanzar en mi casa, le he añadido tres líneas extras con la información anterior pero con los datos relacionados a mi red, para comentarla y descomentarla (respectivo para las del laboratorio), para poder conectarme vía ethernet tanto allí como aquí (ello implica conectarme previamente sin SSH para descomentar/comentar las líneas, pero bueno... también puedo hacer el cambio antes de salirnos de la conexión SSH si soy consciente de que la próxima vez que me conecte va a ser en la otra red).

Modificación hecha, reiniciamos el servicio dhcpd.service mediante “`$ sudo systemctl restart dhcpd.service`”.

Para comprobar la conexión, hacemos ping con www.google.es o con nuestra puerta de enlace como ejemplos sencillos.

Fecha y hora

A continuación, vamos a poner “en hora” a la Raspberry pi, para ello ponemos el comando “`$ sudo nano /etc/systemd/timesyncd.conf`”, y nos vamos a la línea donde está comentada la palabra NTP.

La descomentamos y ponemos “NTP = hora.rediris.es”.

Modificación hecha, reiniciamos el servicio systemd-timesyncd mediante “`$ sudo systemctl restart systemd-timesyncd`”

Para comprobarlo escribimos “\$ date”

Configuración en raspi-config

Ahora, para terminar la configuración previa, vamos a escribir el comando “\$ sudo raspi-config”, y vamos a configurar principalmente la localización geográfica, la zona horaria, el teclado (facilita bastante la experiencia estando acostumbrado a usar el teclado español), y la zona wifi.

Nos vamos a la opción “localisation options”, y realizamos las modificaciones anteriormente dichas:

1. Locales: es_ES
2. Timezone: Europe/Madrid
3. Keyboard: pc 105; teclado spanish
4. Wifi country: ES

¡No salgas todavía de raspi-config! Todavía falta una cosa que configurar, y es el SSH.

SSH

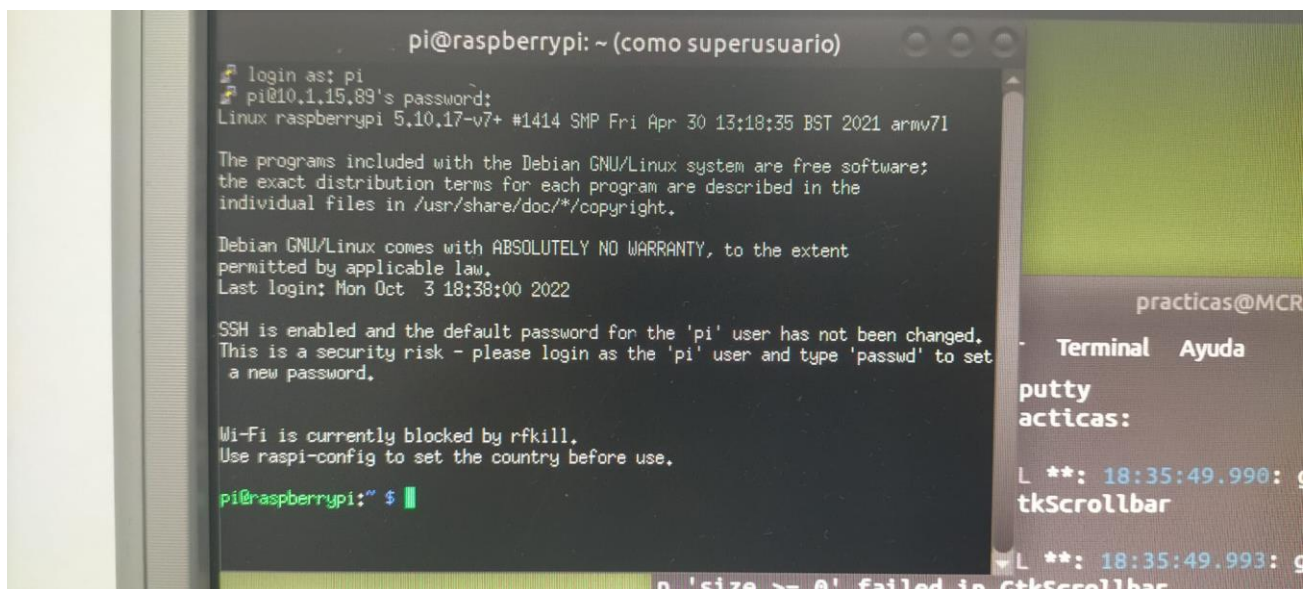
En raspi-config, vamos a activar el protocolo SSH (Secure SHell), que se activa entrando en “interfacing options”, y aceptando la activación, ya tenemos la Raspberry lista para conectarnos desde el PC.

En mi caso, al estar utilizando Windows, he utilizado Putty, que es un cliente SSH con el que me puedo conectar a la placa indicándole la dirección IP que esta posee para conectarse a ella (siempre y cuando estemos en la misma red, claro, por lo que en el laboratorio he tenido que hacer uso de uno de los ordenadores de ahí).

En Linux lo podemos hacer con Putty también o mediante el comando “\$ ssh -X pi@PI_Configurada_Anteriormente”.

Si la conexión es correcta, ya estarás viendo, ya sea en la terminal de Putty, como en la de Linux, la terminal de la RaspBerry pi pidiendo el usuario y la contraseña. De esta forma, ya solo hay que conectarle a nivel físico a la Raspberry el cable de alimentación y el cable Ethernet.

Resultado en el laboratorio:



```

pi@raspberrypi: ~ (como superusuario)
login as: pi
pi@10.1.15.89's password:
Linux raspberrypi 5.10.17-v7+ #1414 SMP Fri Apr 30 13:18:35 BST 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Oct  3 18:38:00 2022

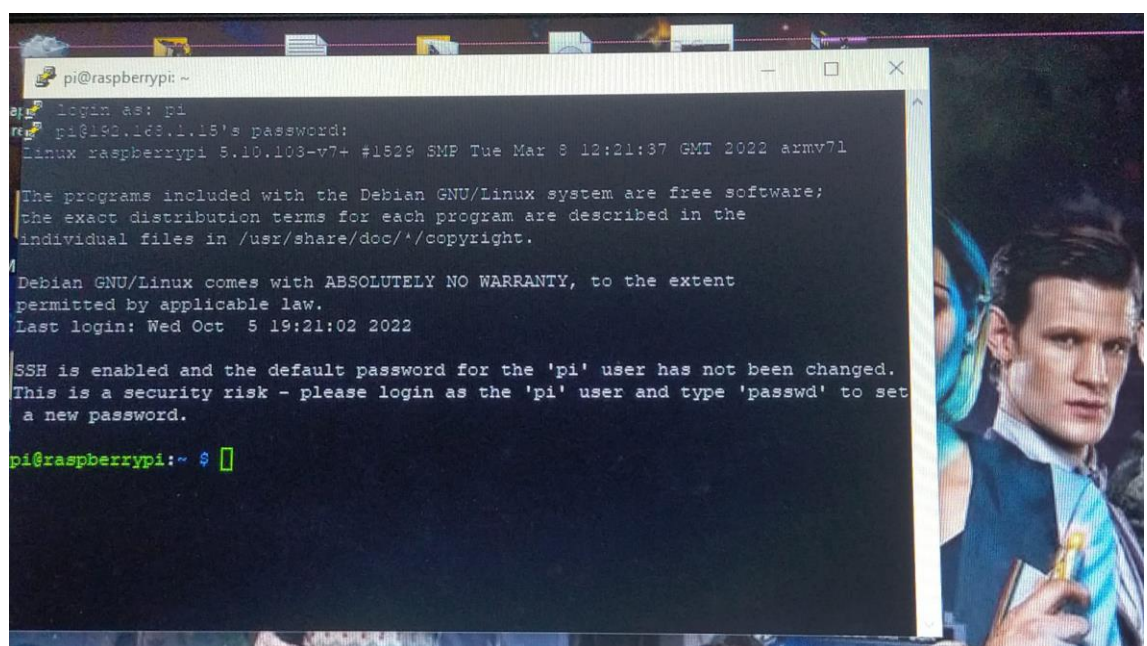
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~ $

```

Resultado en mi casa:



```

pi@raspberrypi: ~
login as: pi
pi@190.168.1.15's password:
Linux raspberrypi 5.10.103-v7+ #1529 SMP Tue Mar 8 12:21:37 GMT 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Oct  5 19:21:02 2022

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

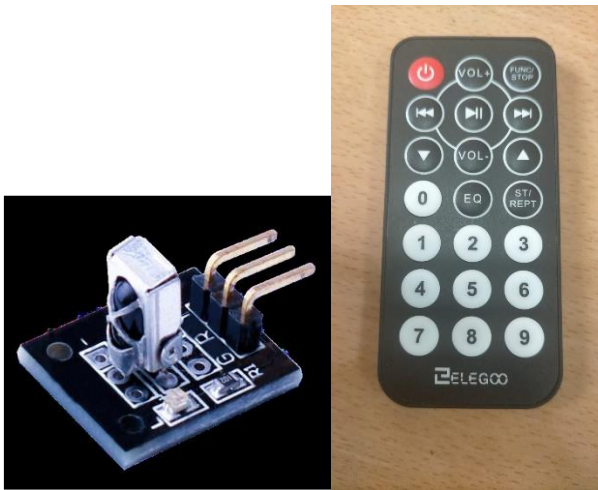
pi@raspberrypi:~ $

```

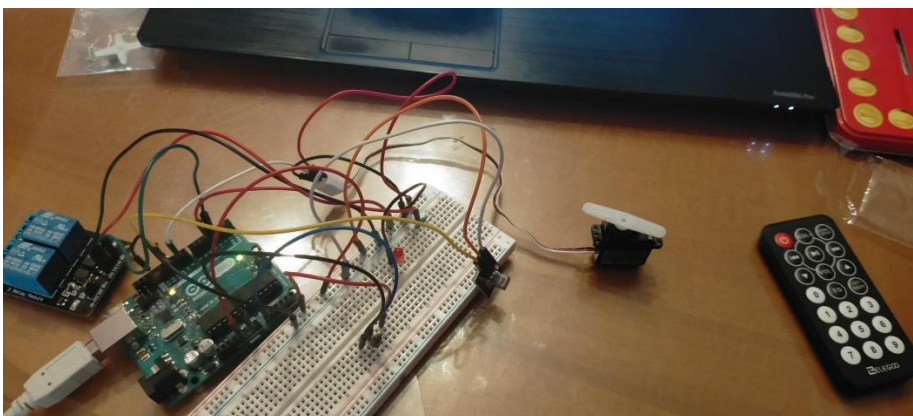
8: EXTRA: infrarrojos

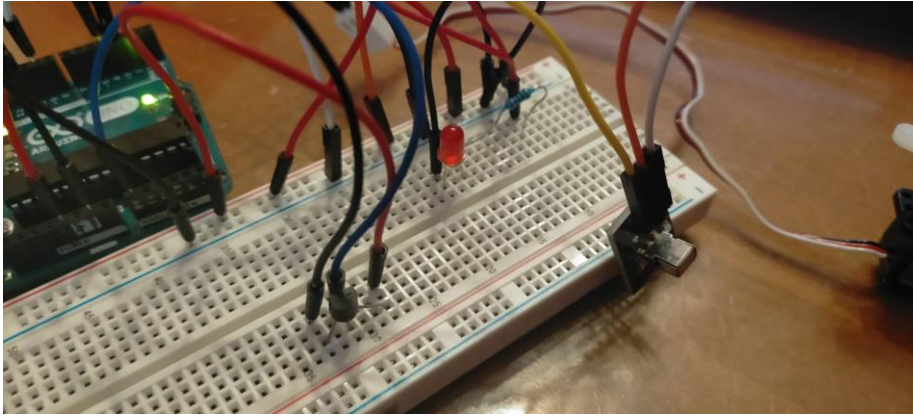
Como ejercicio extra, he creado un código que lo que hace es, usando un mando a distancia y un receptor IR (infrarrojos), controlar tres de los componentes usados en tres de los anteriores códigos, el relé, el servo, y el tmp36GZ, junto a también un LED (Simularía el encendido/apagado de la máquina, aunque lo he puesto más bien como un “juguete” al tener el botón POWER del mando disponible).

Receptor IR y mando a distancia:



Conexión Hardware





Código

Phyton

```
import serial #Importamos serial para realizar la comunicación serie con Arduino desde visual studio
Arduino = serial.Serial('COM4', 9600) #Puerto serie
while True:
    print (Arduino.readline().decode()) #Imprime lo que recibe el puerto serie del PC
```

Es el mismo código que se usó para el ejercicio 5.

Arduino

```

#include "IRremote.h"
#include <Servo.h>

#define sensorPin A0 //Para temperatura

//Pines
int receiver = 11; // Pin del receptor IR
int servo = 9;
int rele = 3;
int serie = 2;
int led = 4;

//Lectura del sensor de temperatura
int lectura;
float voltios;
float temperatura;

//Grados del servo
int gradosServo = 0;

//Objetos
IRrecv irrecv(receiver);    // Objeto creado de 'irrecv'
decode_results results;     // Objeto creado de 'decode_results'
Servo myservo; // objeto servo

void setup()
{
  Serial.begin(9600);
  Serial.println("Esperando botón");
  irrecv.enableIRIn(); // Activación del receptor
  myservo.attach(9); // Pin 9 asignado al servo
  pinMode(rele, OUTPUT);
  pinMode(led, OUTPUT);
  digitalWrite(rele, HIGH);
  digitalWrite(rele, HIGH);
}

void loop()
{
  if (irrecv.decode(&results)) // Hemos recibido una señal IR?
  {
    TraductorIR(); //Decodificación
    irrecv.resume(); // Recibe el siguiente valor
  }
}

```

```

// TOMA LAS ACCIONES BASADAS EN EL CÓDIGO IR RECIBIDO
void TraductorIR(){

// Decodificación

    switch(results.value)
    {
        //ENCENDER/APAGAR LED
        case 0xFFA25D: Serial.print("POWER: ");
        digitalWrite(led,!digitalRead(led));

        if(digitalRead(led) == HIGH){
            Serial.println("Led encendido");
        }

        else{
            Serial.println("Led apagado");
        }
        break;

        //ACTIVAR/DESACTIVAR RELÉ
        case 0xFFE21D:
            Serial.print("FUNC/STOP:");
            digitalWrite(rele,!digitalRead(rele));

            if(digitalRead(rele) == LOW){
                Serial.println("Relé encendido");
            }
        else{
            Serial.println("Relé apagado");
        }

        break;

        case 0xFF629D: Serial.println("VOL+");
        break;

        case 0xFF22DD: Serial.println("FAST BACK");
        break;
        case 0xFF02FD: Serial.print("PAUSE: ");

        //LECTURA DEL SENSOR E IMPRESIÓN POR PUERTO SERIE:
        lectura = analogRead(sensorPin);

        // Convertir a voltios:
        voltios = (lectura * 5.0)/1024.0;

        // De voltios a celcius:

        temperatura = (voltios - 0.5) * 100;

        Serial.print(temperatura);
        Serial.println("°C");
        break;
        case 0xFFC23D: Serial.println("FAST FORWARD");
        break;
        case 0xFFE01F: Serial.println("DOWN");
        break;
    }
}

```

```

case 0xFFA857: Serial.println("VOL-");
break;
case 0xFF906F: Serial.println("UP");
break;
case 0xFF9867: Serial.println("EQ");
break;
case 0xFFB04F: Serial.println("ST/REPT");
break;

//0°
case 0xFF6897: Serial.print("0: ");
gradosServo = 0;
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//90°
case 0xFF30CF: Serial.print("1: ");
gradosServo = 90;
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//135°
case 0xFF18E7: Serial.print("2: ");
gradosServo = 135;
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//180°
case 0xFF7A85: Serial.print("3: ");
gradosServo = 180;
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 5
case 0xFF10EF: Serial.print("4: ");
gradosServo += 5;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 10
case 0xFF38C7: Serial.print("5: ");
gradosServo += 10;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

```



```

//grados + 20
case 0xFF5AA5: Serial.print("6: ");
gradosServo += 20;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 30
case 0xFF42BD: Serial.print("7: ");
gradosServo += 30;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 40
case 0xFF4AB5: Serial.print("8: ");
gradosServo += 40;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 50
case 0xFF52AD: Serial.print("9: ");
gradosServo += 50;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
Serial.println(gradosServo);
break;

//grados + 60
case 0xFFFFFFFF: Serial.print(" REPEAT");

gradosServo += 60;
if(gradosServo > 180){
    gradosServo -= 180;
}
myservo.write(gradosServo); //Indicamos al servo a qué grados se tiene que poner
break;

default:
    Serial.println("Botón desconocido");
}

delay(500);
}

```

Resultado

```
Esperando botón
POWER: Led encendido
POWER: Led apagado
FUNC/STOP:Relé encendido
FUNC/STOP:Relé apagado
PAUSE: 29.59°C
0: 0
1: 90
2: 135
3: 180
4: 5
5: 15
6: 35
7: 65
```

Conclusión

He podido cumplir los objetivos de manera sencilla, teniendo en cuenta que ya poseía agilidad con la plataforma Arduino, y siendo lo que más me ha costado la estrategia para poder programar Raspberry vía SSH tanto en casa como en el laboratorio. Me ha parecido bastante interesante el poder tomar contacto con la Raspberry pi, ya que había escuchado hablar de ella, pero no había tomado contacto con una hasta ahora. Teniendo en cuenta el motivo por el que estamos tratando Arduino y Raspberry, que lo veremos en el bloque 2, más ganas le he echado a cada uno de los ejercicios, y más ganas aún tengo de abordar el siguiente bloque.