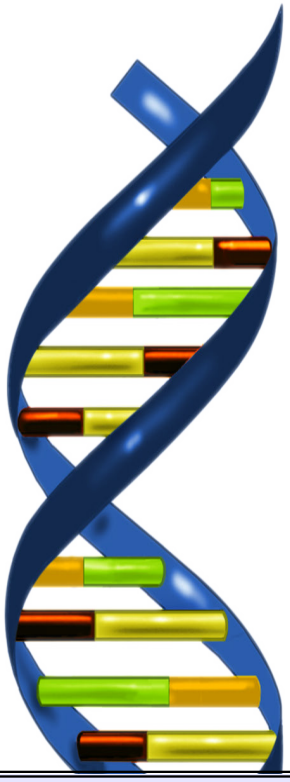


Python: Computación Física
Práctica 3 – Guión 4

Informática Aplicada a la Biología

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

1. Introducción

En esta cuarta práctica con Python nuestro objetivo es que veáis como la programación de ordenadores se utiliza en el mundo real. Cómo se programa un frigorífico, una farola o un semáforo. Es lo que se conoce como *computación física*: el uso de los ordenares para aplicaciones tangibles.

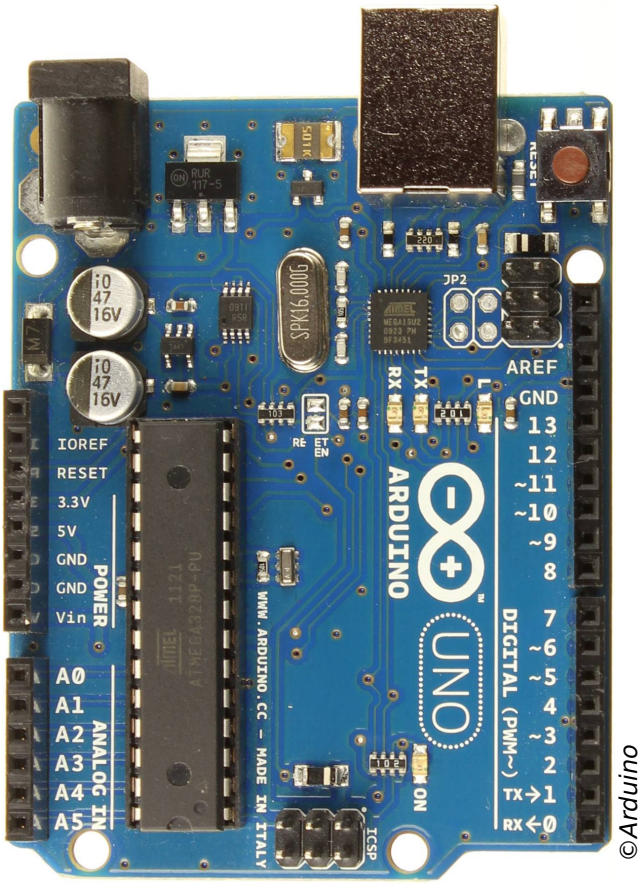
Es esta práctica utilizaremos un sistema de computación física que se conoce como Arduino. Es un dispositivo que nos va a permitir transformar las órdenes que demos en Python en señales eléctricas que puedan entender los dispositivos electrónicos.

Una vez que hayáis realizado la práctica deberíais comprender como se puede utilizar la programación para crear todo tipo de máquinas inteligentes: desde una alarma anti-incendios hasta un coche.

1.1. Conociendo Arduino

Vamos a comenzar conociendo un poco mejor el sistema con el que vamos a trabajar. Arduino es una plataforma de hardware libre, que permite transformar las órdenes dadas a un ordenador en acciones físicas: encendido de luces, movimiento de motores...

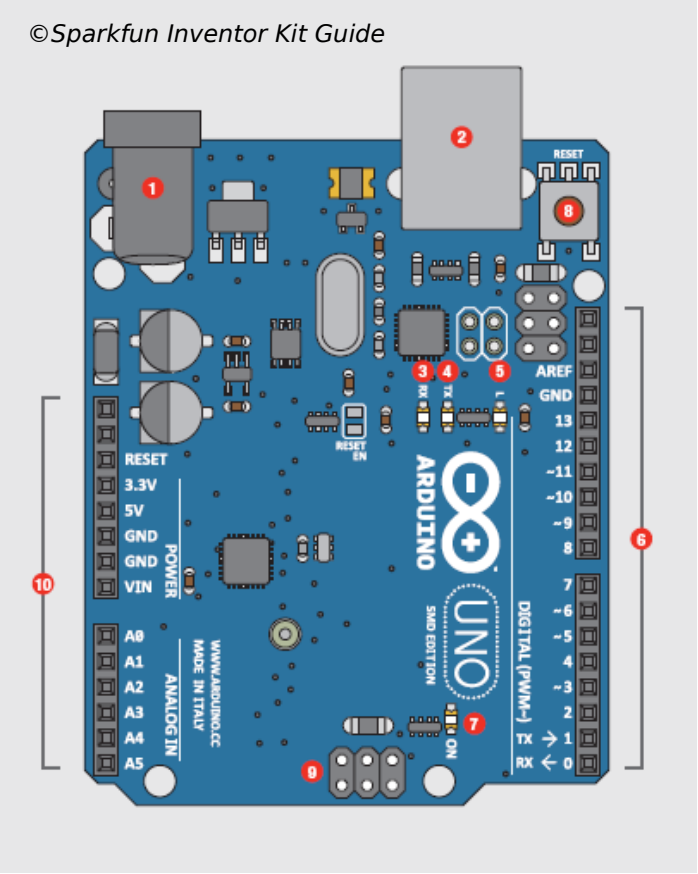
Para ellos Arduino utiliza un microcontrolador y un conjunto de circuitos que transmiten los impulsos eléctricos. Arduino está diseñado para facilitar el uso de la electrónica en proyectos multidisciplinares ya sean artísticos o científicos.



La placa Arduino con la que vamos a trabajar

Aunque Arduino puede parecer al principio un sistema bastante complicado no os preocupéis. Está diseñado para ser utilizado por personas sin la menor formación científica o técnica e incluso por niños (siempre que tengan más de doce años)

Vamos a comenzar viendo cuales son los principales componentes de Arduino. Para ello es recomendable que os fijéis en la siguiente figura, en ella se indican la finalidad de las distintas partes de Arduino.



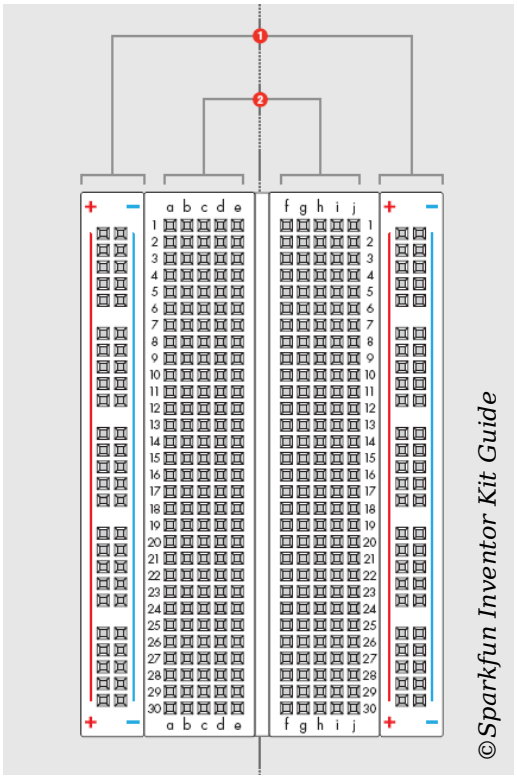
1 – Toma de corriente: Sirve para que Arduino pueda funcionar sin estar conectado a un ordenador
2 – Conexión USB: Permite conectar un ordenador y transmitir información entre el ordenador y Arduino.
6 – Pines digitales: Permiten a Arduino transmitir órdenes a máquinas electrónicas
7 – Luz de encendido: Se enciende cuando Arduino está funcionando.
8 – Botón de reset: Se utiliza cuando queremos resetear el Arduino.
10 – Pines analógicos y de potencia: Se utiliza para recibir información y suministrar energía a las máquinas que estemos controlando

Es importante que os fijéis en la numeración de los pines de salida, **6**, y de entrada, **10**. Cuando queramos dar órdenes o recibir información necesitaremos especificar qué pin vamos a usar.

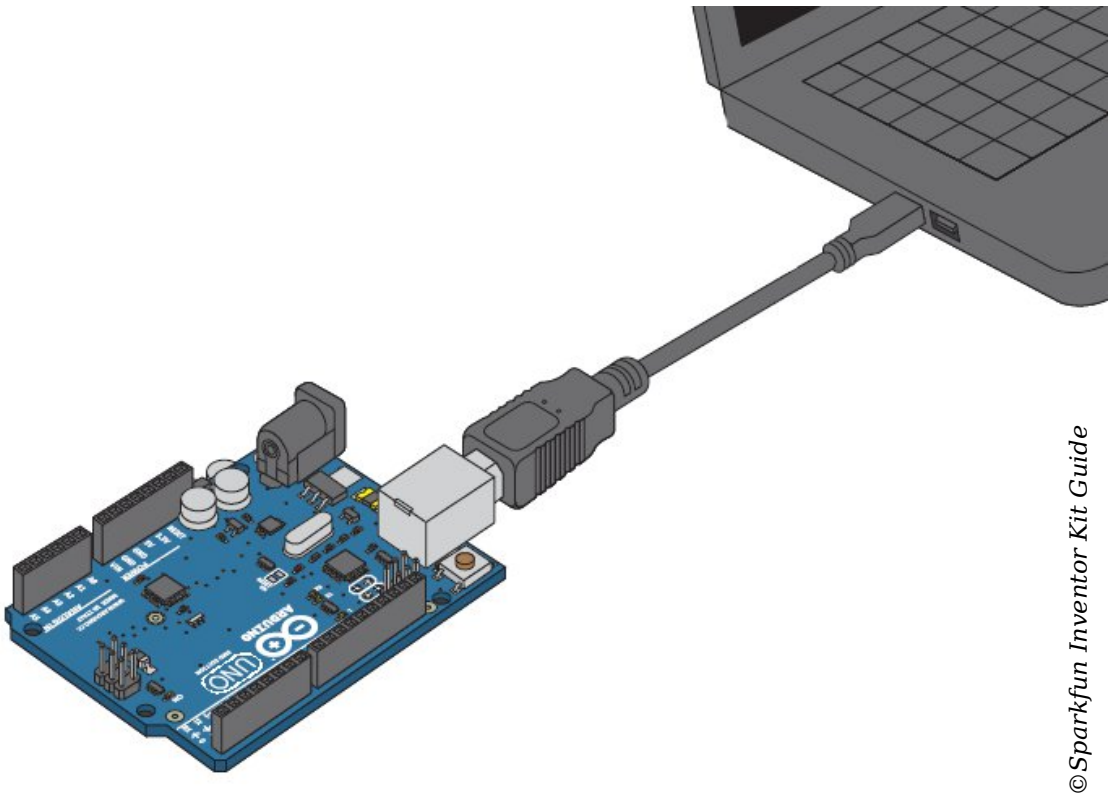
A la derecha del Arduino tenemos un conjunto de circuitos en una placa de pruebas. Aunque no es necesario comprender como funcionan estos circuitos sí es interesante tener una breve noción de cómo funcionan las placas de pruebas.

Placa de pruebas con la que vamos a trabajar

En una placa de pruebas tenemos un conjunto de pines en los que podemos colocar distintos dispositivos electrónicos. Los pines conectados entre sí. Si miráis la figura la zona 1 está conecta de manera vertical, todos los pines en la misma columna están conectados. En la zona 2 los pines están conectados de manera horizontal. Todos los pines en la misma fila están conectados.



Para comenzar lo más importante es **la conexión USB**, es la que nos permite conectar Arduino al ordenador. Deberíais tener una plataforma Arduino completa y un cable USB. Lo primero que tendréis que hacer es conectar Arduino al ordenador usando el cable USB. **Para ello deberéis seguir las instrucciones que os de vuestro profesor de prácticas.**



©Sparkfun Inventor Kit Guide

Como conectar el Arduino al ordenador.

Una vez que hayáis conectado e instalado Arduino pedidle a vuestro profesor de prácticas que compruebe si todos los cables están correctamente conectados. Una vez que él os de el visto bueno podréis empezar con la práctica propiamente dicha.

Lo primero que tendremos que hacer es conectar Python al Arduino. Para ello necesitaremos conocer el puerto COM al que está conectado Arduino (deberíais tenerlo apuntado) e importar el módulo que nos permitirá conectarnos a la placa. El comando que tenéis que escribir en Python es:

```
>> from pyrduino import *
>> mi_arduino = Arduino('COM3')
      Config file: 'board.conf'
            - OUTPUT PINS: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
            - PWM PINS: [3, 5, 6, 9, 10, 11]
      Trying to connect with Arduino on port COM3... [OK] connected.
```

Fíjate que **hemos llamado a nuestro arduino 'mi_arduino',** a partir de ahora siempre que nos refiramos a él tendremos que utilizar ese nombre. Una vez que nos hayamos conectado a Arduino, podremos empezar a trabajar.

2. Primeros pasos en computación física

Nota: recuerda que lo primero que debes de hacer en cada programa es conectar la placa y configurar los pines digitales como salida, aunque no se usen (esto está explicado más abajo).

Una vez que ya habéis instalado Arduino y habéis comprobado que todas las conexiones son correctas podemos empezar a realizar programas que tendrán consecuencias tangibles. Comenzaremos con algo sencillo, encenderemos y apagaremos algunos leds.

Para ello tenemos instalados en la placa de pruebas cinco leds de color rojo o amarillo. Estos leds son como los de la figura y se van a encender y apagar según los comandos que indiquemos.

Leds rojos y amarillos que vamos a utilizar en esta práctica.

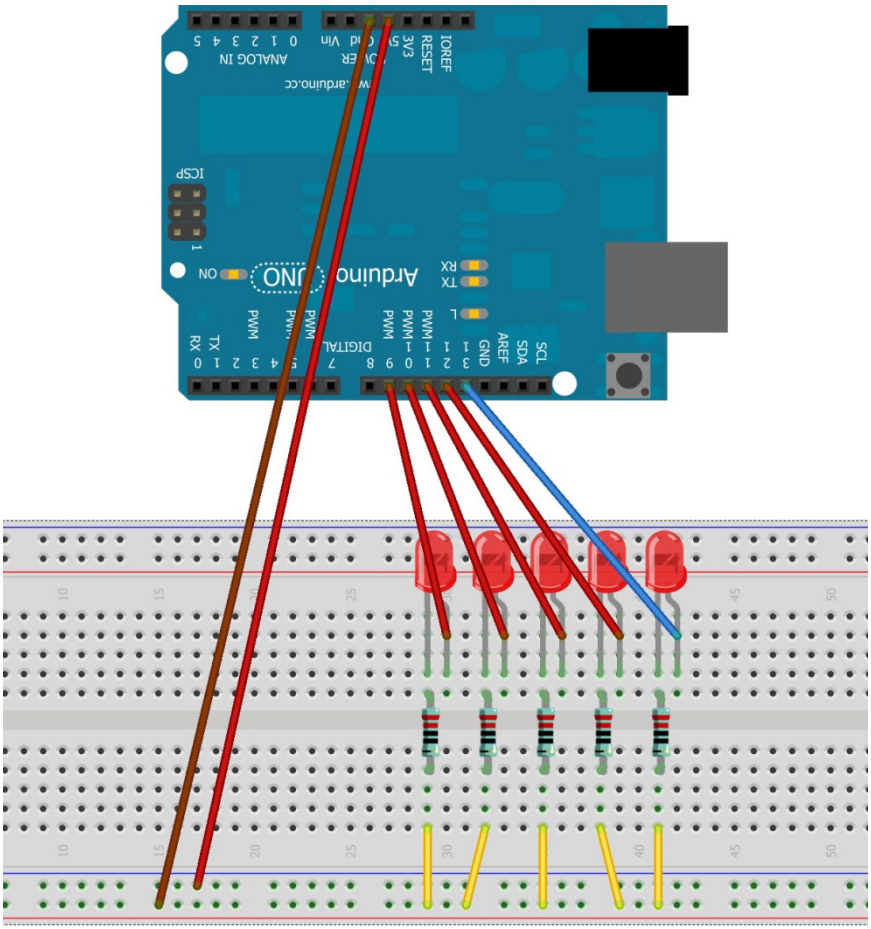
¿Qué es un LED?

Se trata de una revolucionaria fuente lumínica de gran eficiencia, larga vida y un consumo energético mucho más reducido que el de las fuentes tradicionales, como la bombilla incandescente. Considerada la iluminación del futuro, los LED tienen un enorme potencial en zonas subdesarrolladas, a las que no llega el suministro eléctrico.

Aunque los primeros Leds se crearon en los años 1960 es un científico japonés, *Shuji Nakamura*, el descubridor del proceso que ha permitido fabricar todo tipo de leds de una manera fácil y barata. Shuji Nakamura trabajaba en una pequeña empresa de iluminación japonesa cuando se le ocurrió la idea que permitiría fabricar leds. A pesar de no tener una gran formación científica y de que la empresa no le permitió trabajar en su idea, después de cinco años de duro trabajo consiguió revolucionar el mundo de la iluminación.

Para indicarle a un led que se encienda debemos fijarnos cual es el pin digital del arduino al que está conectado. Podéis verlo en el siguiente diagrama, auque es mejor que lo miréis directamente en la placa que se os ha dado. Seguramente vuestros leds estén conectados a *los pines 9, 10, 11, 12 y 13*.

Os tenéis que fijar en el pin del Arduino (de color azul), los números que aparecen en la placa de pruebas (de color rojo) no son importantes.



Esquema del circuito con el que estamos trabajando

Lo primero que deberás hacer siempre es indicar qué pines se van a usar como salida, ya que por defecto todos están definidos como entrada. Usa para ello la función setOutputPins (si no se le pasa nada a la función, todos los pines digitales se configurarán como salida):

```
>> mi_arduino.setOutputPins()
```

Esto debes hacerlo siempre en cada programa justo después de realizar la conexión con Arduino. Una vez que ya hemos identificado los pines a los que están conectados y los hemos configurado como salida, podemos comenzar a encender y apagar leds.

Para encender el led conectado al pin 11 el comando sería:

```
>> mi_arduino.ledOn(11)
```

Para encender los leds conectados a los pins 9,10 y 12 sería:

```
>> mi_arduino.ledOn([9,10,12])
```

Para apagar el led conectado al pin 9 el comando sería:

```
>> mi_arduino.ledOff(9);
```

Para apagar el led conectado a los pins 11, 12 y 13 el comando sería:

```
>> mi_arduino.ledOff([11,12,13])
```

Otra forma de hacerlo:

```
>> mi_arduino.ledOff(list(range(11,14)))
```

Al menos uno de los leds está conectado a Arduino mediante un cable largo. Desconecta ese cable del Arduino e intenta encender el led. ¿Qué pasa? Escríbelo como comentario en tu programa.

Ahora tienes que copiar el código de la siguiente función. ¿Qué es lo que hace?

```
import time
def destellos(ard,num):
      #Función destellos. Descripción de la función
      #Argumento de entrada
      #ard: Arduino conectado
      #num:

      pinLed1 = 13
      pinLed2 = 9

      for cont in range(num):
          ard.ledOff(pinLed1)
          ard.ledOn(pinLed2)
          time.sleep(1) #Para la ejecución un segundo
          ard.ledOn(pinLed1)
          ard.ledOff(pinLed2)
          time.sleep(1) #Para la ejecución un segundo

      apaga_led(ard,pinLed1);
      apaga_led(ard,pinLed2);
```

Para ejecutarlo deberás llamar a la siguiente función al final del programa (recuerda que antes de eso debes haber realizado la conexión con Arduino y configurado los pines como salida):

```
>> destellos(mi_arduino,4)
```

1. Modifica el código de destellos para que en vez de utilizar sólo dos leds utilice todos los leds disponibles (como si fuese una luz de navidad, un cylon o el coche

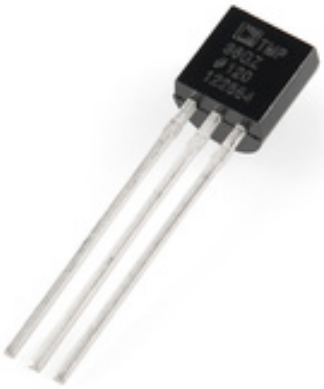
fantástico).

2. Modifica el valor dado en `pause(1)` para que las luces se muevan más deprisa.
3. ¿Qué pasa si utilizamos `pause()` en vez de `pause(1)`? *Aviso: La respuesta no es que se queda parado.*

3. Percibiendo el mundo exterior

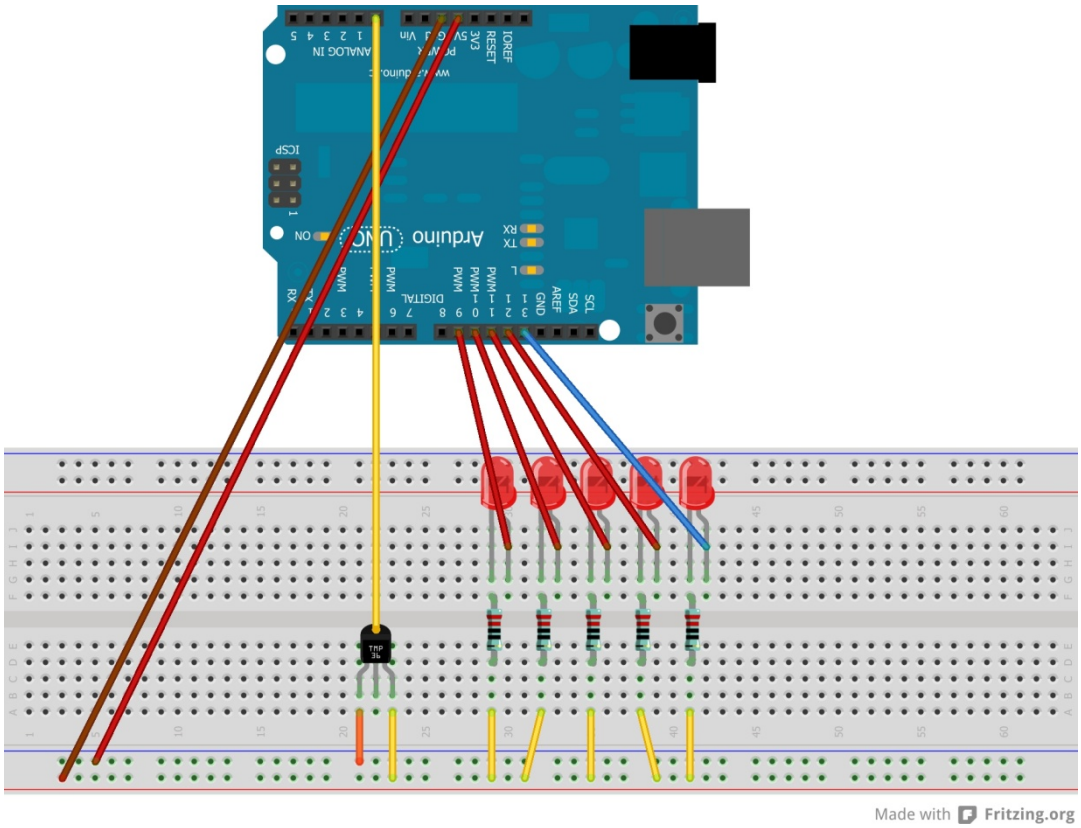
Nota: Debes tener los leds apagados para realizar correctamente esta sección.

Una vez que hemos aprendido como utilizar un programa para encender y apagar leds vamos a ver como percibir el mundo exterior utilizando sensores electrónicos. Para ello vamos a comenzar midiendo la temperatura de la sala.



Sensor de temperatura que vamos a utilizar en esta práctica.

Para leer la temperatura debemos fijarnos cual es el pin analógico al cual está conectado nuestro sensor. Podéis verlo en el siguiente diagrama, aunque es mejor que lo miréis directamente en la placa que se os ha dado. Seguramente vuestro sensor de temperatura esté conectados a*l pin analógico 0*.



Esquema del circuito incluyendo el sensor de temperatura con el que estamos trabajando

Una vez que ya hemos identificado el pin al que está conectado podemos comenzar a ver cuál es la temperatura. Si el sensor está conectado al pin 0, sería:

```
>> mi_arduino.readTemp(0)
14.6
```

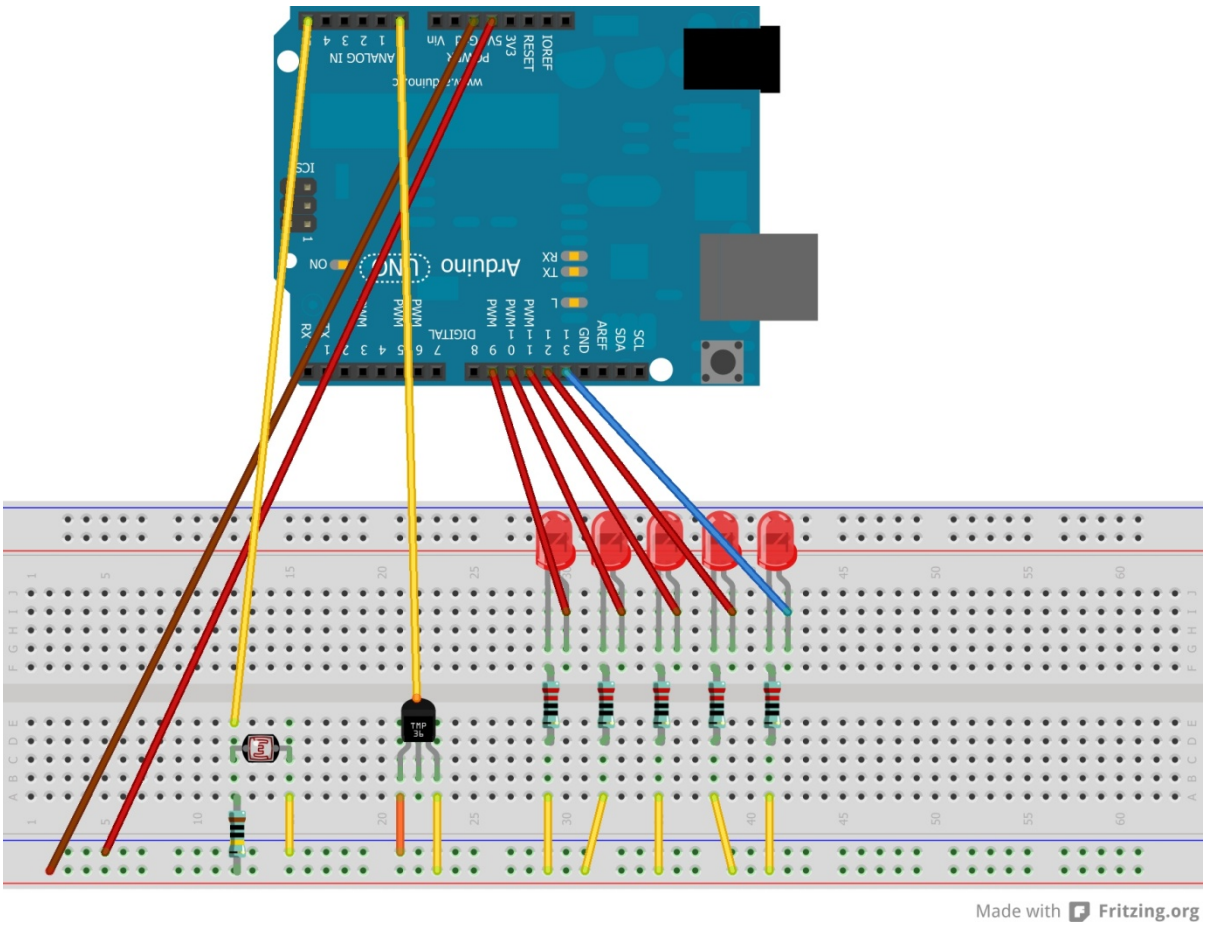
Fíjate que la salida de la función `readTemp` va a depender de la temperatura de la sala. No tiene por qué coincidir con el valor del guión.

Pon ahora tus dedos en la parte negra del sensor de temperatura. Espera unos segundos y vuelve a medir la temperatura. ¿Qué ha pasado? Describe qué resultados has obtenido.

Ahora vamos a proceder a medir la luminosidad en el laboratorio. Para ello vamos a utilizar un sensor que nos da una medida entre cero y uno. Cero será una oscuridad total y uno el máximo de luminosidad.

Sensor de luminosidad que vamos a utilizar en esta práctica.

Para leer la luminosidad debemos fijarnos cual es el pin analógico al cual está conectado nuestro sensor. Podéis verlo en el siguiente diagrama, auque es mejor que lo miréis directamente en la placa que se os ha dado. Seguramente vuestro sensor de temperatura esté conectados a*l pin analógico 5*.



Esquema del circuito incluyendo el sensor de luminosidad con el que estamos trabajando

Una vez que ya hemos identificado el pin al que está conectado podemos comenzar a ver cual es la luminosidad. Si el sensor está conectado al pin 5, sería:

```
>> mi_arduino.readLight(5)
0.8213
```

La salida de la función `lee_luminosidad` va a depender de la luminosidad de la sala. No tiene por qué coincidir con el valor del guión.

Cubre ahora con tus dedos el sensor de luminosidad. Vuelve a medirla. ¿El nuevo valor es mayor o menor que el anterior? Comenta los resultados tu programa Python.

4. Ejercicios

Para terminar, se proponen al alumno una serie de ejercicios para afianzar los conocimientos adquiriros. Así mismo, se pretende que el alumno sea capaz de imaginar posibles aplicaciones reales utilizando lo aprendido hasta ahora.

1. Modifica el código de la función `encendido_automatico` para que funcione correctamente. Luego describe cual es su funcionalidad y cómo se consigue implementar dicha funcionalidad.

```
import time

def encendido_automatico(ard):
      #funcion encendido_automatico
      #Enciende los leds si es de noche
      #Si hay luz los apaga
      #-------------------------------------
      #Repito durante unos diez segundos
      for cont in range(50):

          #Veo qué luminosidad hay
          luz = ard.readLight(5)

         #Si es menor que un valor dado enciendo la farola
        if luz < 0.9:
            ard.ledOn([9,10,11,12,13])
        else: #Apago la farola
            ard.ledOff([9,10,11,12,13])
        time.sleep(.2) #para unas decimas antes de volver a repetir
```

La función se llama con el siguiente comando:

```
>> encendido_automatico(mi_arduino)
```

2. Escribe la función `destellos_luminosidad` modificando la función `destellos` para que la frecuencia de los destellos dependa de la luminosidad del ambiente. A menor luminosidad mayor frecuencia de destellos.

3. Analiza el código de la función `escribe` e indica qué hace. Para ello debes indicar cual es su funcionalidad y cómo se consigue implementar dicha funcionalidad.

```
def escribe (ard,num):
      #Saca por leds el número que introduzcamos en binario
      #Máximo número 31
      #Se utilizan los pines del 13 al 9
      #Argumento de entrada
      #ard: Arduino conectado
      #num: el número que vamos a pasar a binario

      #Los leds con los que voy a trabajar
      rangoLed = range(9,14)

      for cont in rangoLed
          val = num%2 #Calculo el resto de dividir por dos
```

```
        num = num/2 #Calculo la parte entera del cociente

        #Si la cifra en binario vale 1
        if val == 1:
           ard.ledOn(cont); #enciendo el led
        else:
           ard.ledOff(cont); #apago el led
```

La función se llama con el siguiente comando:

```
>> escribe(mi_arduino,9)
```

4. Escribir una función de nombre `escribe_temperatura` que use las funciones `readTemp` y `escribe`. Esta función lee la temperatura del sensor de temperatura y la escribe en binario utilizando los leds disponibles. Crear una versión alternativa de nombre `lecturaContinua` en la que el Arduino muestre de manera continua la temperatura en los leds. *Nota: para poder realizar correctamente la medida de la temperatura los leds deben estar apagados.*

4.1. ¿Qué hemos aprendido?

- Los principios de la computación física.
- Cómo funcionan algunos componentes electrónicos: Leds, sensores de temperatura, iluminación...
- Como utilizar lo aprendido en programación para diseñar máquinas que hagan lo que queremos.