Deusto
Facultad de Ingeniería
Ingeniaritza Fakultatea

Actividad de Clase. Semana 6

Sensores para Sistemas Embebidos

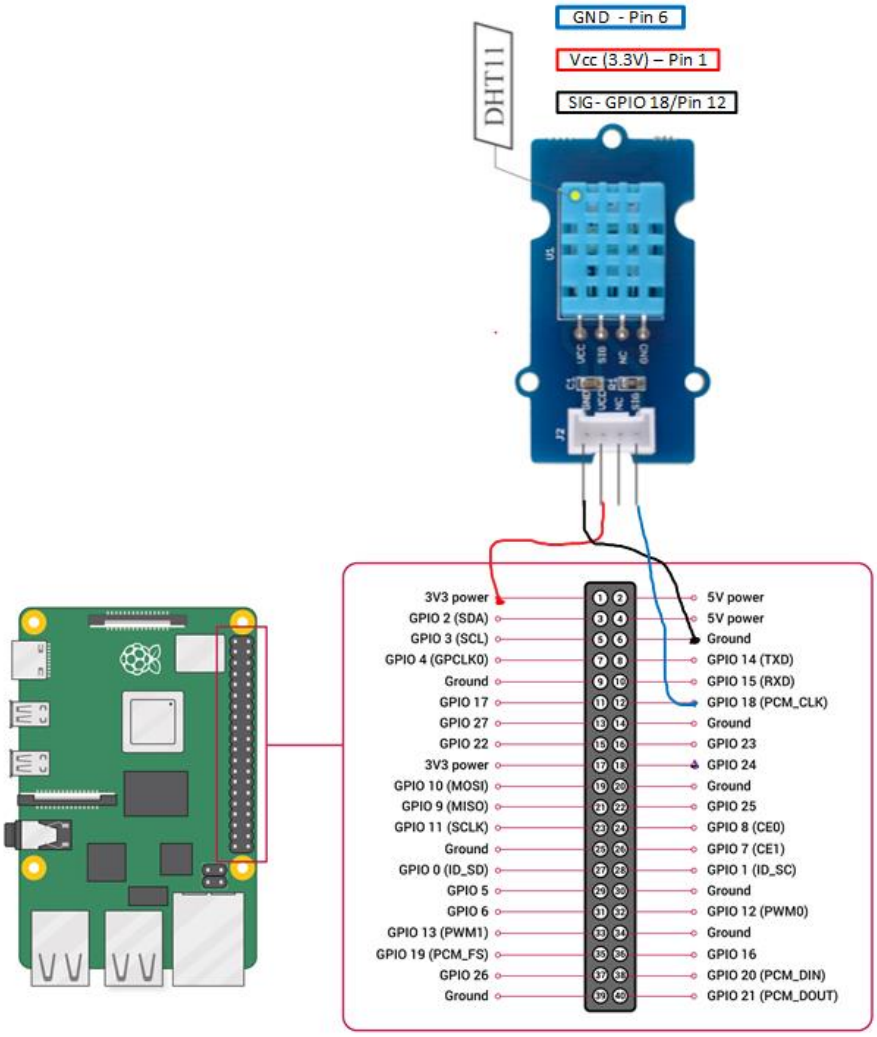
Instrucciones

En esta actividad vais a aprender a usar el sensor de Temperatura y Humedad **DHT111** (datasheet link), y realizar un programa en Python que permita leer los valores del sensor de forma selectiva con vuestra Raspberry Pi, y mostrarlos en la terminal.

Vais a trabajar en grupos de 2-3 personas, y la entrega se hará individual a través de ALUD.

Parte 1 (0.5 puntos)

Conectar el sensor DHT111 como se muestra en la Figura debajo, y ejecutar el fichero de ejemplo `temHum.py`. Debéis tener además en la misma carpeta el fichero `dht_config.py`, pues es donde realmente se hace la comunicación directa con el sensor. Comprobar que funciona correctamente. Debéis subir en ALUD una captura de pantalla donde se muestre que funciona correctamente el programa, imprimiendo por pantalla los valores de temperatura y humedad cada 1 segundo.



Los pines NC (not connected) se dejan sin conectar

Parte 2 (9.5 puntos)

Entrega: Programa de Python que desarrolléis.

Realizar el siguiente programa en Python, basado el el programa de ejemplo, que se debe ejecutar en la RPi. En esta actividad necesitaréis usar la RPi, el sensor anterior, y un switch que se os entregó el primer día. La figura debajo muestra las conexiones requeridas. Se debe entregar el fichero en Python del programa. Este programa sólo muestra valores de temperatura (no debe mostrar los valores de humedad)

El funcionamiento del programa debe ser el siguiente:

- Cuando el switch está en valor HIGH, se debe mostrar por pantalla (con print) el valor de temperatura que devuelve el sensor cada 1 segundo.
- Cuando el switch está en valor LOW, se para la lectura, y no se muestra nada por pantalla.
- Usad un *callback* para detectar los cambios de HIGH a LOW y de LOW a HIGH en el switch, para parar o activar la lectura del valor de temperatura, respectivamente.

Se os proporciona el programa `temHum_switch_incompleto.py`, que podéis usar como base para realizar el programa pedido. Sólo tenéis que programar las partes que se indica cómo `# ---- TO DO ----`

Recordad el ejemplo de *callback* que vimos en clase la semana pasada:

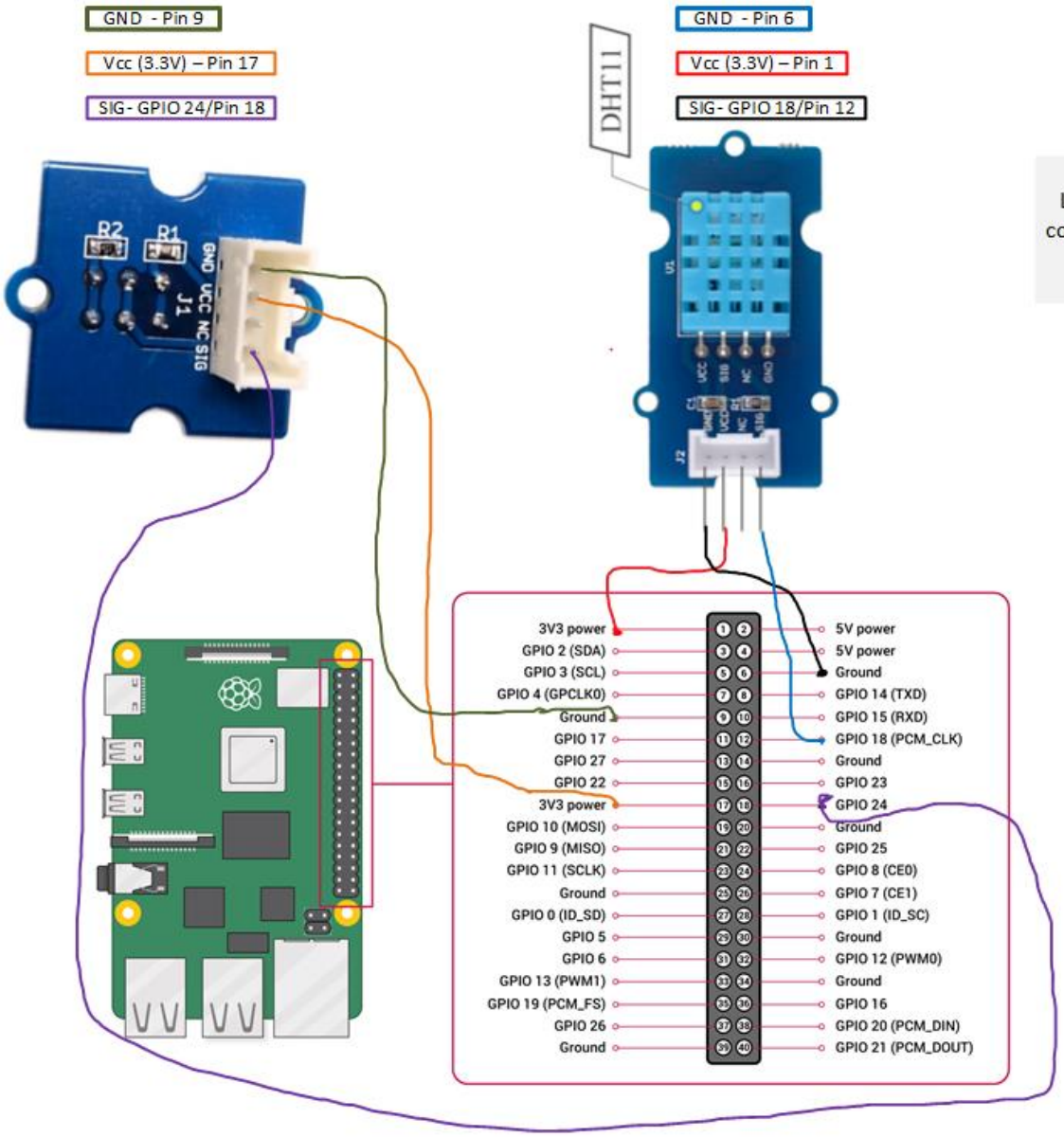
Función del callback

```
def procesar(numcanal):
     print("Detectado")
```

Asociar un callback a un GPIO

```
GPIO.add_event_detect(numero_del_GPIO, evento, callback=procesar)
```

evento puede ser:
GPIO.RISING
GPIO.FALLING
GPIO.BOTH



Los pines NC (not connected) se dejan sin conectar

Apéndice

A continuación, se proporciona información útil acerca de cómo se procede a leer los valores de temperatura y humedad del sensor.

El sensor nos devuelve los datos en serie, con el siguiente formato de 40 bits

> 8bit humidity integer data + 8bit decimal data +8 bit temperature and humidity data + 8bit temperature decimal integer data +8 bit parity bit.

Definición de cálculo del **bit de paridad:** este bit se usa para comprobar la correcta recepción de la información, o si la trama se ha recibido corrupta. Se obtiene de la siguiente forma:

```
8bit high humidity + 8bit low humidity +8 bit high temperature + 8bit low temperature
```

El resultado de la operación debe ser igual al byte de paridad recibido.

- Ejemplo de recepción correcta en base al bit de paridad

Example One：40 receives the data to:

0011 0101	0000 0000	0001 1000	0000 0000	0100 1101
High humidity 8	Low humidity 8	High temperature 8	Low temperature 8	Parity bit

Calculated as follows:

0011 0101+0000 0000+0001 1000+0000 0000= 0100 1101

Receive data is correct:

Humidity: 0011 0101=35H=53%RH

Temperature:0001 1000=18H=24℃

- Ejemplo de recepción incorrecta (error en el bit de paridad)

Example Two: The received data is 40:

0011 0101	0000 0000	0001 1000	0000 0000	0100 1001
High humidity 8	High humidity 8	High temperature 8	High temperature 8	Parity bit

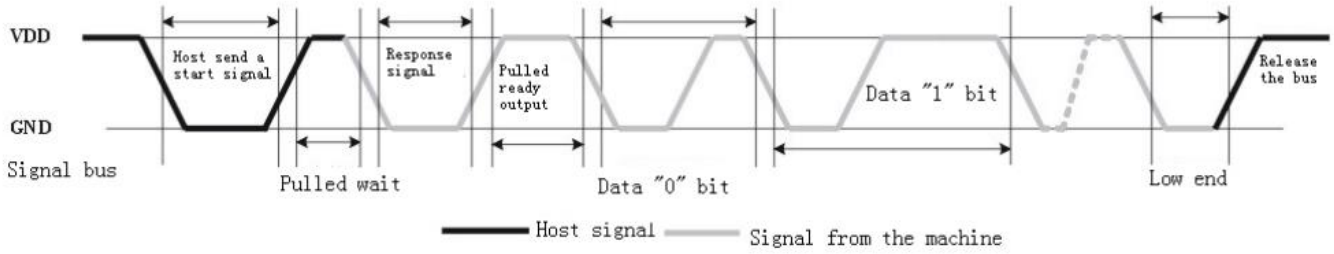
Calculated as follows:

0011 0101+0000 0000+0001 1000+0000 0000 = 0100 1101

01001001 is not equal to 01001101

The received data is not correct, give up, again receiving data.

Diagrama temporal: la recepción de los bits debe hacerse teniendo en cuenta las estrictas restricciones temporales. Para detalles de los tiempos, consultad el datasheet



Formato de los bits 1 y 0 recibidos