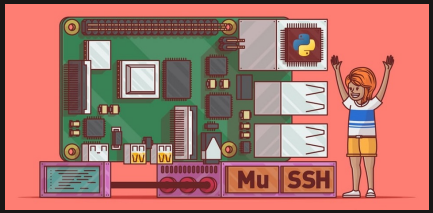
Creación de proyectos físicos con Python en la Raspberry Pi

Introducción al Internet de las Cosas



Marco Teran
Universidad Sergio Arboleda

2023

Contenido

Introducción

Introducción

- El Raspberry Pi es una placa de cómputo físico líder en el mercado.
- La Raspberry Pi es utilizada por aficionados y estudiantes para interactuar con el mundo que les rodea.
- Python viene preinstalado en la Raspberry Pi, lo que permite a los usuarios comenzar a construir sus proyectos de manera inmediata.

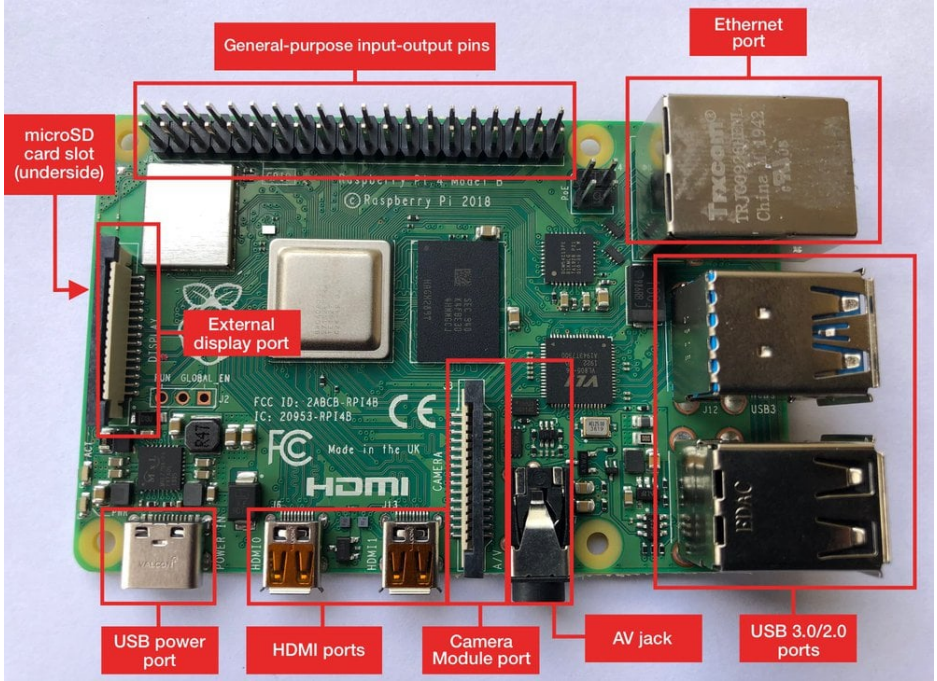
Introducción

- Se enseña cómo configurar y utilizar la Raspberry Pi.
- Se muestra cómo ejecutar Python en la Raspberry Pi, cómo leer datos de sensores físicos y cómo enviar datos a componentes externos.
- Con Python en la Raspberry Pi, se pueden crear proyectos únicos y personalizados.

GPIO

Raspberry Pi

- Raspberry Pi es una computadora de placa única desarrollada por la Raspberry Pi Foundation, una organización benéfica con sede en el Reino Unido.
- Fue diseñada originalmente para proporcionar a los jóvenes una opción de cómputo asequible para aprender a programar.
- Se ha convertido en una opción popular en las comunidades de creadores y bricolaje debido a su tamaño compacto, entorno completo de Linux y pines de entrada/salida de propósito general (GPIO).


General-purpose input-output pins
Ethernet port
microSD card slot (underside)
External display port
USB power port
HDMI ports
Camera Module port
AV jack
USB 3.0/2.0 ports
© Raspberry Pi 2018
FCC ID: 2ABCB-RPI4B
IC: 20953-RPI4B
Made in the UK
HDMI
USB3

Pines GPIO

- El Raspberry Pi cuenta con **cuarenta GPIO pins** en su parte superior que permiten conectar componentes externos.
- Los GPIO pins permiten una gran variedad de proyectos y casos de uso para el Raspberry Pi en la electrónica y la informática.

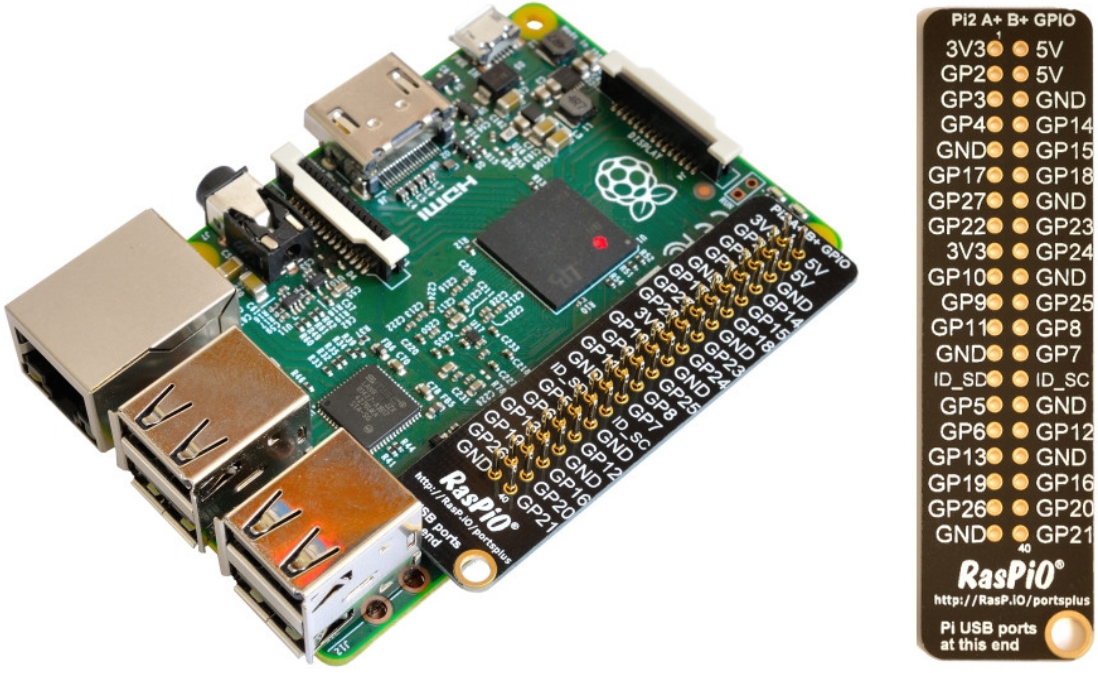
Pines GPIO

Figura 1: Distribución GPIO

El Raspberry Pi cuenta con cinco tipos diferentes de pines:

- GPIO: Pines de propósito general que pueden ser utilizados como entrada o salida.
- 3V3: Pines que suministran una fuente de alimentación de 3.3 V para los componentes. 3.3 V también es la tensión interna que suministran todos los pines GPIO.
- 5V: Pines que suministran una fuente de alimentación de 5 V, igual que la entrada de alimentación USB que alimenta al Raspberry Pi. Algunos componentes, como el sensor de movimiento infrarrojo pasivo, requieren 5 V.
- GND: Pines que proporcionan una conexión a tierra para los circuitos.
- ADV: Pines de propósito especial que son avanzados y no se cubren en este tutorial.

Más información: `https://es.pinout.xyz/`


Pi2 A+ B+ GPIO
1
3V3 5V
GP2 5V
GP3 GND
GP4 GP14
GND GP15
GP17 GP18
GP27 GND
GP22 GP23
3V3 GP24
GP10 GND
GP9 GP25
GP11 GP8
GND GP7
ID_SD ID_SC
GP5 GND
GP6 GP12
GP13 GND
GP19 GP16
GP26 GP20
GND GP21
40
RasPiO
http://RasP.iO/portsplus
Pi USB ports at this end


Raspberry Pi GPIO BCM numbering
5V
UART
14 TXD
15 RXD
18 PWM0
23
24
25
8 CE0
7 CE1
I2C0
1 ID_SC
12 PWM0
16
SPI1
20 MOSI
21 SCLK
2
1
40
39
SDA 2
SCL 3
GCLK0 4
17
27
22
MOSI 10
MISO 9
SCLK 11
ID_SD 0
GCLK1 5
GCLK2 6
PWM1 13
MISO 19
26
3V3
I2C1
3V3
SPI0
pinout.xyz

Rpi.GPIO

Rpi.GPIO

La biblioteca RPi.GPIO es una biblioteca exclusiva de Python que proporciona interacciones básicas con los pines GPIO, pero aún no tiene implementación de ningún protocolo de conexión. Se pueden descargar los archivos de Python del proyecto desde Pypi.org, y la página de inicio del proyecto está alojada en Sourceforge.

- Es una biblioteca en Python que se utiliza para controlar la interfaz GPIO en la Raspberry Pi.
- Fue desarrollada por Ben Croston y se lanzó bajo una licencia de software libre MIT.
- Proporciona un conjunto de archivos de Python y fuente que está incluido en Raspbian de manera predeterminada.
- La documentación, que incluye programas de ejemplo, se encuentra disponible en el proyecto Wiki.

Instalación de la librería Rpi.GPIO en Raspberry Pi

- **Paso 1:** Abrir la terminal en la Raspberry Pi.
- **Paso 2:** Actualizar la lista de paquetes disponibles usando el siguiente comando de apt-get:
  - `sudo apt-get update`
- **Paso 3:** Instalar la última versión de Python usando el siguiente comando de apt-get:
  - `sudo apt-get install python3`
- **Paso 4:** Instalar pip para Python 3 usando el siguiente comando de apt-get:
  - `sudo apt-get install python3-pip`
- **Paso 5:** Actualizar pip usando el siguiente comando de pip:
  - `sudo pip3 install --upgrade pip`
- **Paso 6:** Instalar la librería Rpi.GPIO usando el siguiente comando de pip:
  - `sudo pip3 install Rpi.GPIO`
- **Paso 7:** Verificar la instalación usando un programa que haga uso de la librería Rpi.GPIO.

Rpi.GPIO

Para utilizar RPi.GPIO en el resto de tu script de Python, necesitas poner la siguiente declaración en la parte superior de tu archivo:

```
import RPi.GPIO as GPIO
```

- Esta declaración **importa** el módulo RPi.GPIO, y va un paso más allá al proporcionar un nombre local – GPIO – que llamaremos para hacer referencia al módulo a partir de ahora.
- Es importante destacar que el módulo RPi.GPIO viene instalado por defecto en las versiones más recientes de Raspbian Linux.

Pin Numbering Declaration

GPIO.BOARD vs GPIO.BCM

- GPIO.BOARD: Esquema de numeración de placa. Los números de los pines siguen los números de los pines en el encabezado P1.
- GPIO.BCM: Números de pin específicos del chip Broadcom. Estos números de pin siguen el sistema de numeración de nivel inferior definido por el cerebro del chip Broadcom de Raspberry Pi.

Recomendación de uso del Pi Wedge: Si estás usando Pi Wedge, se recomienda usar la definición GPIO.BCM, ya que estos son los números serigrafiados en la PCB. GPIO.BOARD puede ser más fácil si estás conectando directamente al encabezado.

Pin Numbering Declaration

Configuración del número del sistema

Para especificar en tu código qué sistema de numeración se está utilizando, utiliza la función GPIO.setmode(). Por ejemplo...

- GPIO.setmode(GPIO.BCM) activará los números de pin específicos del chip Broadcom.
- GPIO.setmode(GPIO.BOARD) activará el esquema de numeración de placa.

Pin Numbering Declaration

Configuración de un canal como entrada o salida

Para configurar un canal como entrada o salida, llama a cualquiera de las siguientes funciones, donde channel es el número de canal basado en el sistema de numeración que especificaste al llamar a setmode.

- GPIO.setup(channel, GPIO.IN)
- GPIO.setup(channel, GPIO.OUT)

Digital Output

Digital Output

Los Outputs en la Raspberry Pi son pines que se utilizan para enviar señales a otros componentes del circuito. Estos pines pueden estar configurados como salida digital o salida PWM.

- Los pines configurados como salida digital pueden ser escritos en alto o en bajo.
- La función GPIO.output([pin], [GPIO.LOW, GPIO.HIGH]) se utiliza para establecer un pin como alto o bajo.
- Los valores de alto y bajo corresponden a 3,3 V y 0 V, respectivamente.
- En lugar de GPIO.LOW y GPIO.HIGH, también se pueden usar los valores 0, 1, True o False para establecer el valor de un pin.

PWM (.Analog") Output

PWM (.Analog") Output

El PWM (.Analog") Output en la Raspberry Pi es un pin que se puede configurar para enviar señales analógicas. Solo un pin puede ser configurado para el envío de señales PWM, que es el pin 18 (i.e. pin 12 en la placa).

- Para configurar el PWM, se utiliza la función GPIO.PWM([pin], [frecuencia]).
- Es recomendable asignar la instancia a una variable.
- El método pwm.start([duty cycle]) se utiliza para establecer un valor inicial.
- El valor de [duty cycle] puede ser cualquier valor entre 0 y 100.
- La función pwm.ChangeDutyCycle([duty cycle]) se utiliza para ajustar el valor del PWM Output.
- El valor de [duty cycle] puede ser ajustado entre 0 y 100.
- La función pwm.stop() se utiliza para apagar el PWM Output.

Digital Inputs

Digital Inputs

Los Inputs en la Raspberry Pi son pines que se utilizan para recibir señales de otros componentes del circuito. Estos pines pueden ser configurados como entrada digital.

- La función GPIO.input([pin]) se utiliza para leer el valor de un pin.
- La función input() devuelve un valor booleano (True o False) que indica si el pin está en estado alto o bajo.
- Se puede utilizar una instrucción if para determinar si el valor del pin es alto o bajo.

Resistencias Pull-Up/Down

Resistencias Pull-Up/Down

Las resistencias Pull-Up/Down son elementos utilizados para mantener un nivel de voltaje constante en un circuito, incluso cuando no se está aplicando una entrada de señal.

- Para usar una resistencia Pull-Up en un pin, agregar `pull_up_down=GPIO.PUD_UP` como tercer parámetro en `GPIO.setup()`
- Para usar una resistencia Pull-Down, utilizar `pull_up_down=GPIO.PUD_DOWN`
- Si no se declara nada en ese tercer valor, ambas resistencias serán deshabilitadas.

Retardos

Retardos

Los retardos son utilizados para disminuir la velocidad de ejecución de un script en Python.

- Para incluir retardos en un script, se debe incluir el módulo `time`.
- A lo largo del script, se puede utilizar `time.sleep([segundos])` para dar un descanso al script.
- Se pueden utilizar decimales para establecer precisamente el retraso deseado.
- Por ejemplo, para un retraso de 250 milisegundos, escribir `time.sleep(0.25)`
- El módulo time incluye muchas otras funciones útiles, además de sleep.

Limpieza de Recursos

Limpieza de Recursos

Garbage Collecting es el proceso de liberar recursos que ya no son necesarios en un script.

- Para liberar los recursos utilizados por el script en los GPIOs, se utiliza el comando `GPIO.cleanup()` al final del script.
- Aunque el Pi sobrevivirá si se olvida este comando, es una buena práctica incluirlo siempre que sea posible.

Entrada GPIO basada en eventos en Python

- Queremos que nuestro programa emita un solo mensaje cuando se presiona un botón, en lugar de emitir continuamente mensajes.
- Para lograr esto, necesitamos usar eventos GPIO en Python.
- Un evento GPIO llama a una función de Python cuando se desencadena un evento.
- Un evento puede ser un cambio de estado en un pin (de bajo a alto o de alto a bajo) o un cambio en la señal del pin (subida o bajada).
- En nuestro caso, queremos detectar cuando se presiona el botón, lo que se llama borde de subida.
- Antes de configurar el evento, debemos escribir la función de callback que se ejecutará cuando se detecte el evento.
- La función de callback es una función de Python regular que puede contener cualquier código de Python.

Pulsador táctil

Pulsador táctil

Se conectará un botón táctil al Raspberry Pi para registrar una señal ON al cerrar el circuito electrónico al presionarlo. El botón táctil se usará para ejecutar diferentes funciones de Python dependiendo de su estado.

1. Se conectará un jumper hembra-macho desde el pin GND del Raspberry Pi hasta el riel negativo del protoboard.
2. El botón táctil se colocará en el centro del protoboard, cruzando la ranura.
3. Se conectará un jumper macho-macho desde el riel negativo del protoboard hasta la fila donde se encuentra la pata inferior izquierda del botón.
4. Se conectará un jumper hembra-macho desde el pin GPIO4 del Raspberry Pi hasta la fila del protoboard donde se encuentra la pata inferior derecha del botón.
5. La señal ON registrada al presionar el botón se utilizará para activar diferentes acciones.

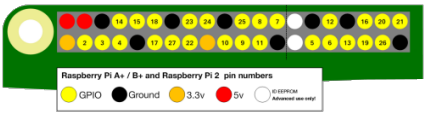
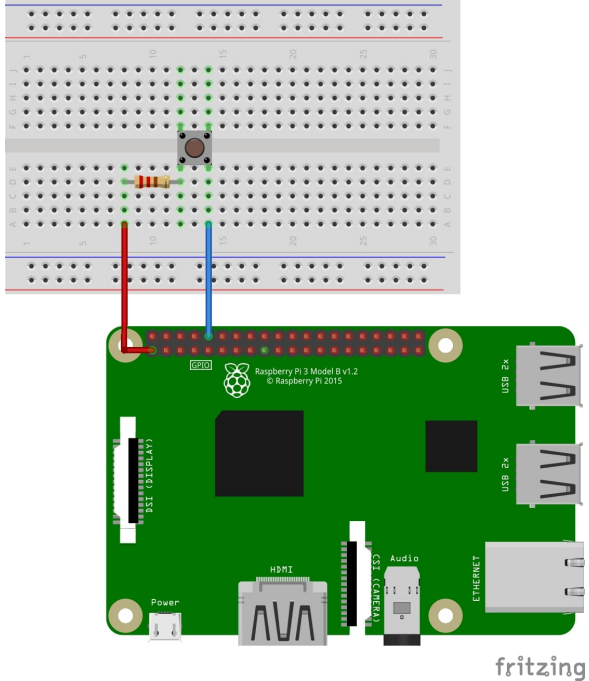


Figura 2: Distribución GPIO

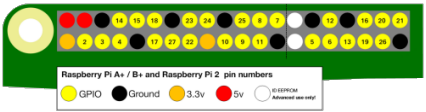

GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
CSI (CAMERA)
HDMI
Audio
ETHERNET
Power
fritzing

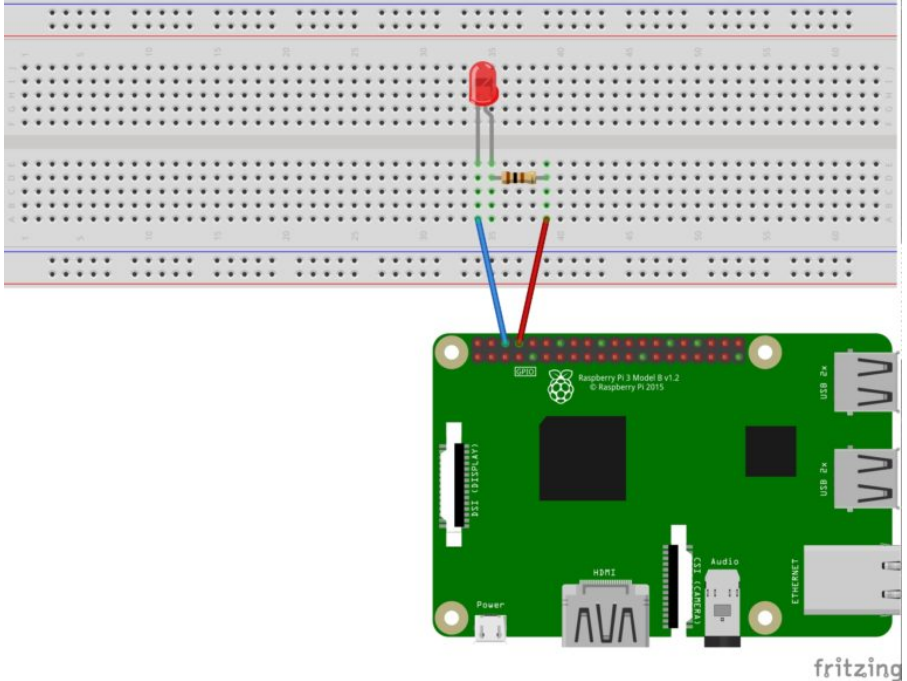
```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

def button_callback(channel):
print("Button was pushed!")

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering

# Set pin 10 to be an input pin and set initial value to be pulled low (off)
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Setup event on pin 10 rising edge
GPIO.add_event_detect(10,GPIO.RISING,callback=button_callback)

message = input("Press enter to quit\n\n") # Run until someone presses enter

GPIO.cleanup() # Clean up
```

Blinking led

Blinking led

- Blinking Led es el equivalente a "Hola Mundo."[en] microcontroladores.
- Se utiliza el numerado de pines de la placa, ya que es intuitivo.
- Se usa la biblioteca Rpi.GPIO para controlar los periféricos.
- Es necesario conectar un pin a tierra (GND) y otro a la salida (GPIO).
- Para el pin positivo se debe elegir uno con las letras GPIO.
- Se utiliza el pin 8 para el positivo y el 6 para tierra.




GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
CSI (CAMERA)
HDMI
Audio
ETHERNET
Power
fritzing

```python
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
# Set pin 8 to be an output pin and set initial value to low (off)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)

while True: # Run forever
    GPIO.output(8, GPIO.HIGH) # Turn on
    sleep(1) # Sleep for 1 second
    GPIO.output(8, GPIO.LOW) # Turn off
    sleep(1) # Sleep for 1 second
```

Sensor DHT11

Sensor DHT11

Soporte de CircuitPython para los dispositivos de temperatura y humedad DHT11 y DHT22

Dependencias

Este controlador depende de:

- Adafruit CircuitPython

Asegúrese de que todas las dependencias estén disponibles en el sistema de archivos de CircuitPython. Esto se logra fácilmente descargando el paquete de bibliotecas y controladores de Adafruit.

Conexión de sensores de humedad DHT

Conexión en Raspberry Pi

Es fácil conectar estos sensores a su Raspberry Pi. Nuestro código puede utilizar cualquier pin GPIO, pero utilizaremos el pin GPIO 4 para nuestros diagramas y código. Una vez que lo tenga funcionando, simplemente puede adaptar el código para cambiar a cualquier otro pin GPIO (por ejemplo, pin 18). También puede tener tantos sensores DHT como desee, pero no pueden compartir el pin de datos: ¡cada sensor necesita un pin de datos único! Para los sensores DHT11 y DHT22, no olvide conectar una resistencia de 4.7K - 10K desde el pin de datos a VCC.

Configuración en Python Vamos a utilizar una biblioteca especial llamada adafruit_blinka (llamada así por Blinka, la mascota de CircuitPython) para proporcionar la capa que traduce la API de hardware de CircuitPython a cualquier biblioteca que proporcione la placa Linux.

Instalación de CircuitPython Libraries en la Raspberry Pi

Instalación de CircuitPython Libraries en la Raspberry Pi

- Ejecuta las actualizaciones estándar con los siguientes comandos:
  - `sudo apt-get update`
  - `sudo apt-get upgrade`
  - `sudo apt-get install python3-pip`
- Actualiza `setuptools` con el siguiente comando:
  - `sudo pip3 install --upgrade setuptools`
- Ten en cuenta que el soporte de Python2 ha sido eliminado, por lo que deberás usar `pip3` y `python3` como comandos o establecer Python 3 como la instalación predeterminada de Python.
- Puede que necesites reiniciar antes de instalar `Blinka`. El script `raspi-blinka.py` te informará si es necesario.
- Hemos creado un script para asegurarnos de que tu Pi esté correctamente configurada e instalar `Blinka`. Solo se necesitan unos pocos comandos para ejecutarlo y la mayoría de ellos son para instalar las dependencias.
  - `cd ~`
  - `sudo pip3 install --upgrade adafruit-python-shell`
  - `wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/r`
  - `sudo python3 raspi-blinka.py`
  - `sudo pip3 install adafruit-blinka`

Instalación de CircuitPython Libraries en la Raspberry Pi

- Ejecuta las actualizaciones estándar con los siguientes comandos:
  - `sudo apt-get update`
  - `sudo apt-get upgrade`
  - `sudo apt-get install python3-pip`
- Actualiza `setuptools` con el siguiente comando:
  - `sudo pip3 install --upgrade setuptools`
- Ten en cuenta que el soporte de Python2 ha sido eliminado, por lo que deberás usar `pip3` y `python3` como comandos o establecer Python 3 como la instalación predeterminada de Python.

Instalación de CircuitPython Libraries en la Raspberry Pi

- Puede que necesites reiniciar antes de instalar `Blinka`. El script `raspi-blinka.py` te informará si es necesario.
- Hemos creado un script para asegurarnos de que tu Pi esté correctamente configurada e instalar `Blinka`. Solo se necesitan unos pocos comandos para ejecutarlo y la mayoría de ellos son para instalar las dependencias.
  - `cd ~`
  - `sudo pip3 install --upgrade adafruit-python-shell`
  - `wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/master/r`
  - `sudo python3 raspi-blinka.py`
  - `sudo pip3 install adafruit-blinka`

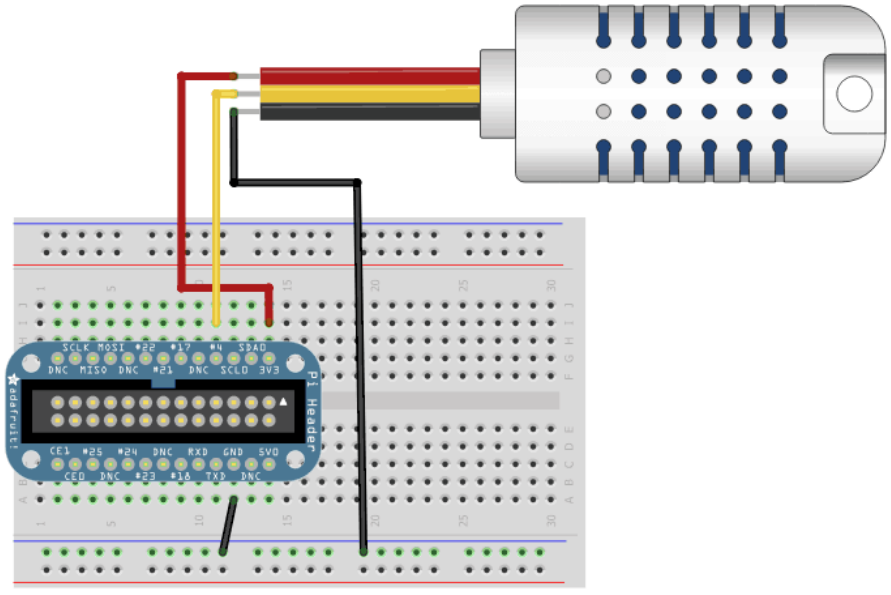
Instalación de la biblioteca CircuitPython-DHT

Para comunicarse con el sensor DHT, también es necesario instalar una biblioteca. Como estamos usando Adafruit Blinka (CircuitPython), podemos instalar bibliotecas de CircuitPython directamente en nuestra pequeña placa Linux.
En este caso, vamos a instalar la biblioteca CircuitPython_DHT. Esta biblioteca funciona con los sensores DHT22 y DHT11.
Para instalar la biblioteca CircuitPython-DHT, ejecute el siguiente comando:

```
pip3 install adafruit-circuitpython-dht
sudo pip3 install adafruit-circuitpython-dht
```

Además, para completar la instalación, se requiere instalar la librería libgpiod2 con el siguiente comando:

```
sudo apt-get install libgpiod2
```


SCLK MOSI #22 #17 #4 SDAD
DNC MISO DNC #21 DNC SCLD 3V3
CE1 #25 #24 DNC RXD GND 5VD
CED DNC #23 #18 TXD DNC
Pi Header

```
import time
import board
import adafruit_dht

# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT11(board.D18)
# dhtDevice = adafruit_dht.DHT22(board.D18, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )
```

```
except RuntimeError as error:
    # Errors happen fairly often, DHT's are hard to read, just keep going
    print(error.args[0])
    time.sleep(2.0)
    continue
except Exception as error:
    dhtDevice.exit()
    raise error

time.sleep(2.0)
```

¡Muchas gracias por su atención!

¿Preguntas?

Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: marco.teran@usa.edu.co

UNIVERSIDAD
SERGIO ARBOLEDA