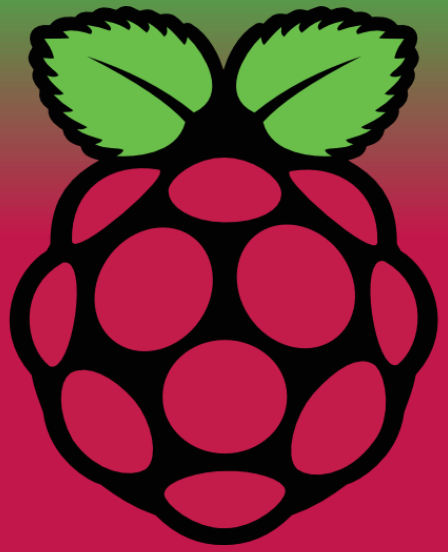


# 2024 Raspberry Pi Workshop

December 27, 2024

Thomas Trickel

# Agenda

Welcome

Raspberry Pi

Terminal

Python

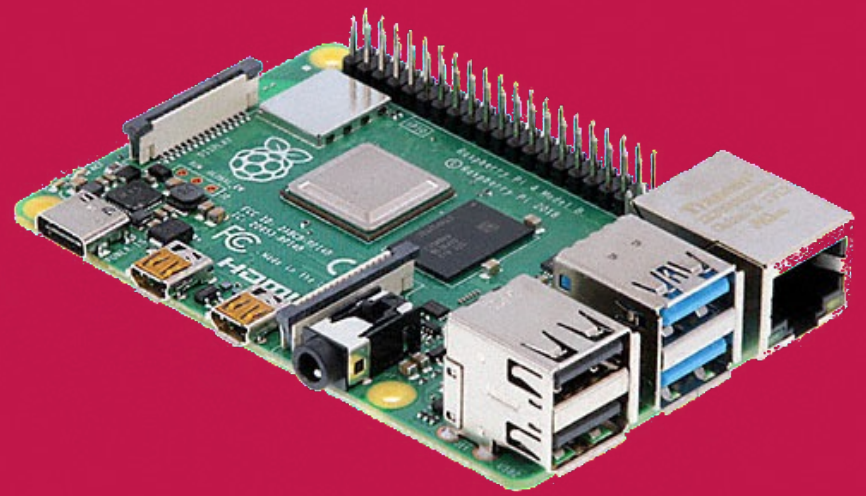
Blink LED

Break

Switch Controlled LED

Write to File on Switch Press

Cronjob Alarm

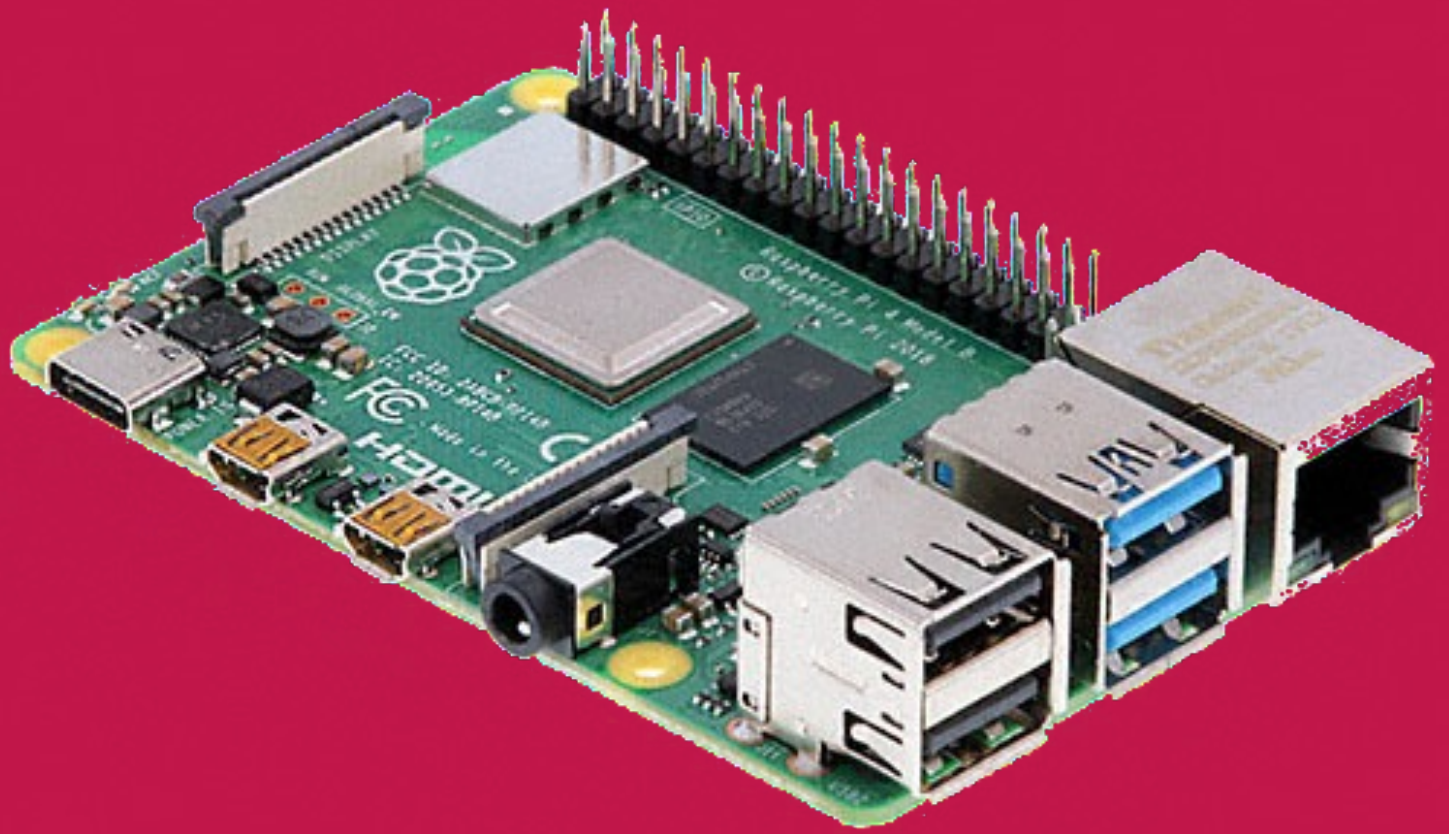


# Welcome

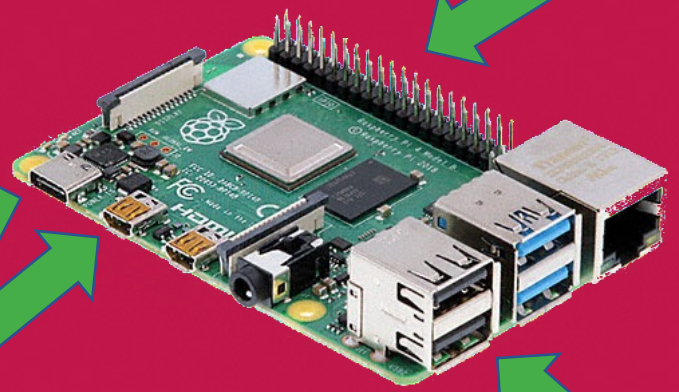
Name

"Coolest Tech" Used

# Raspberry Pi



# Raspberry Pi



GPIO

Power

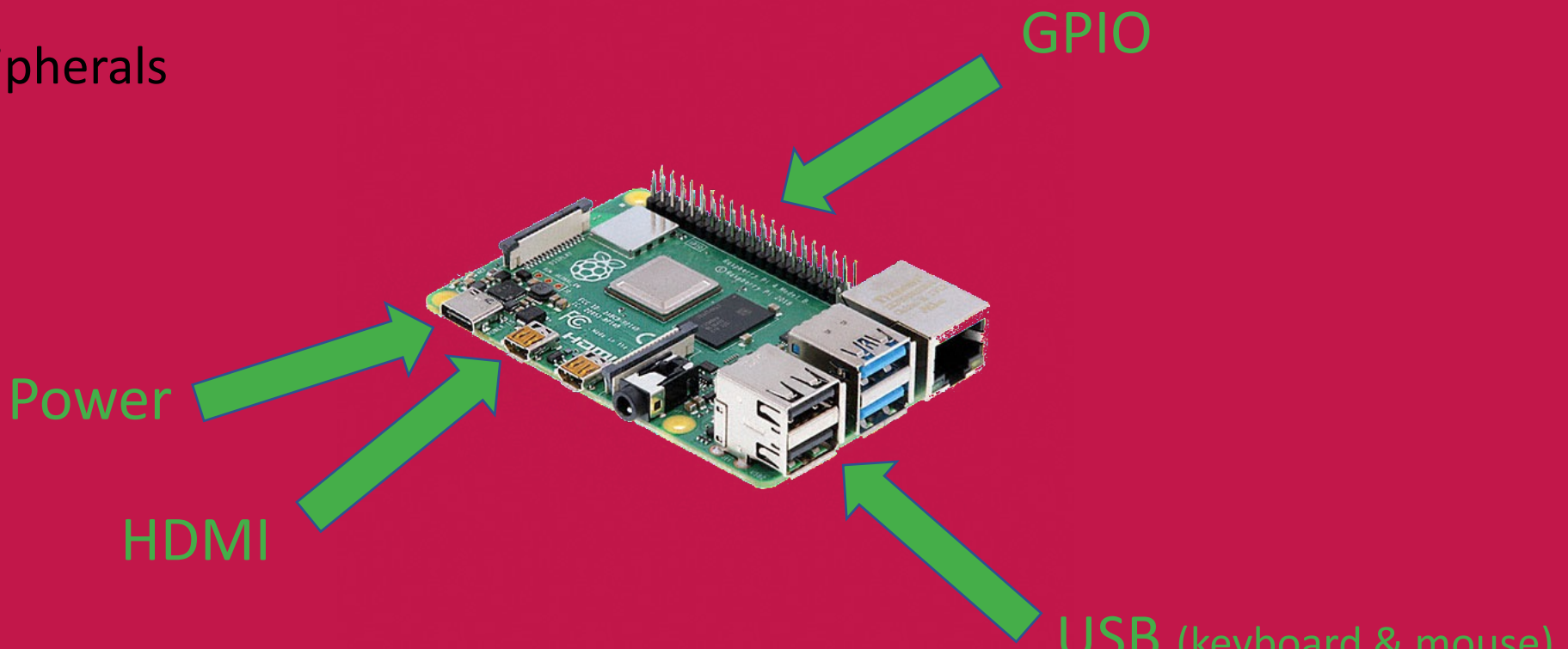
HDMI

USB (keyboard & mouse)

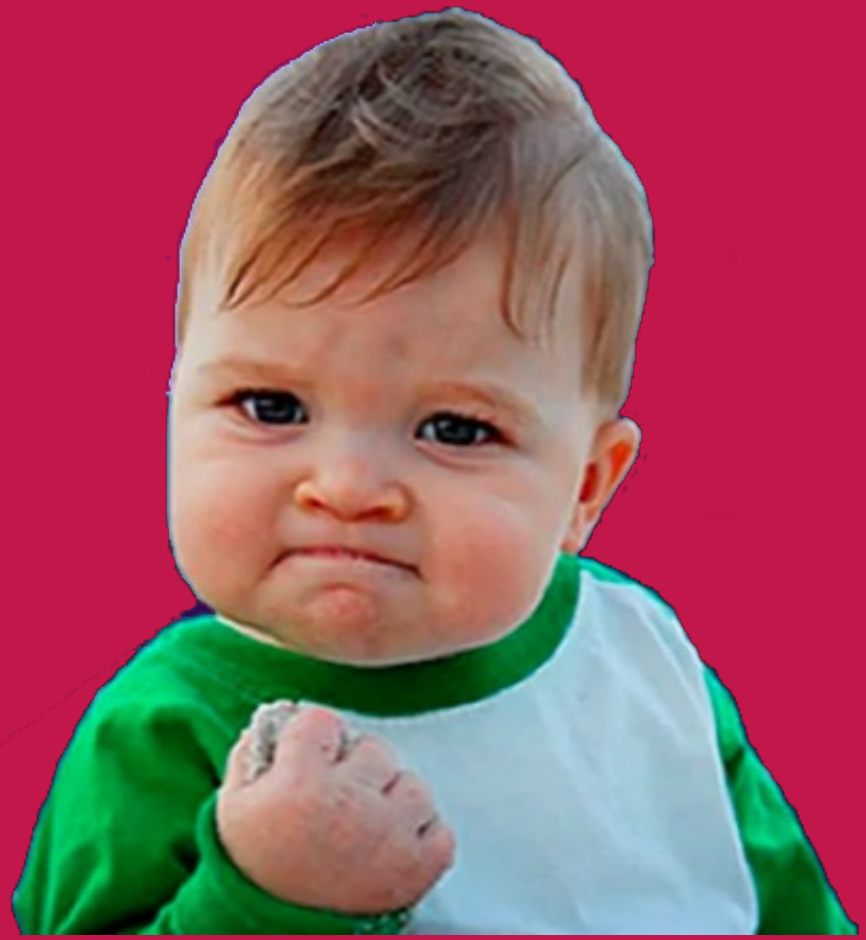
# Raspberry Pi

Connect Peripherals

Power On



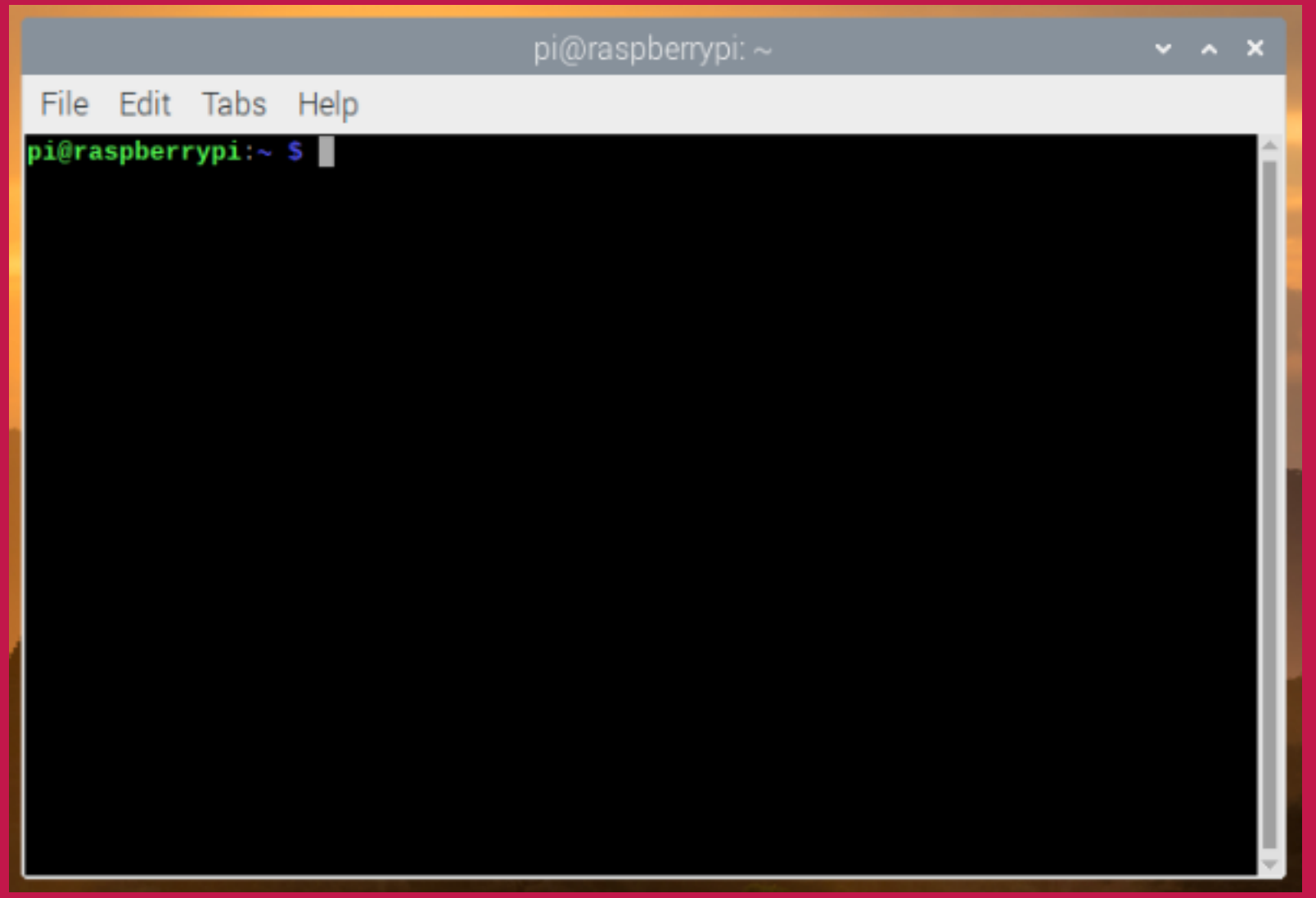
# Excellent!



# Terminal

"Old School"

- Text based
- Command Line Interface

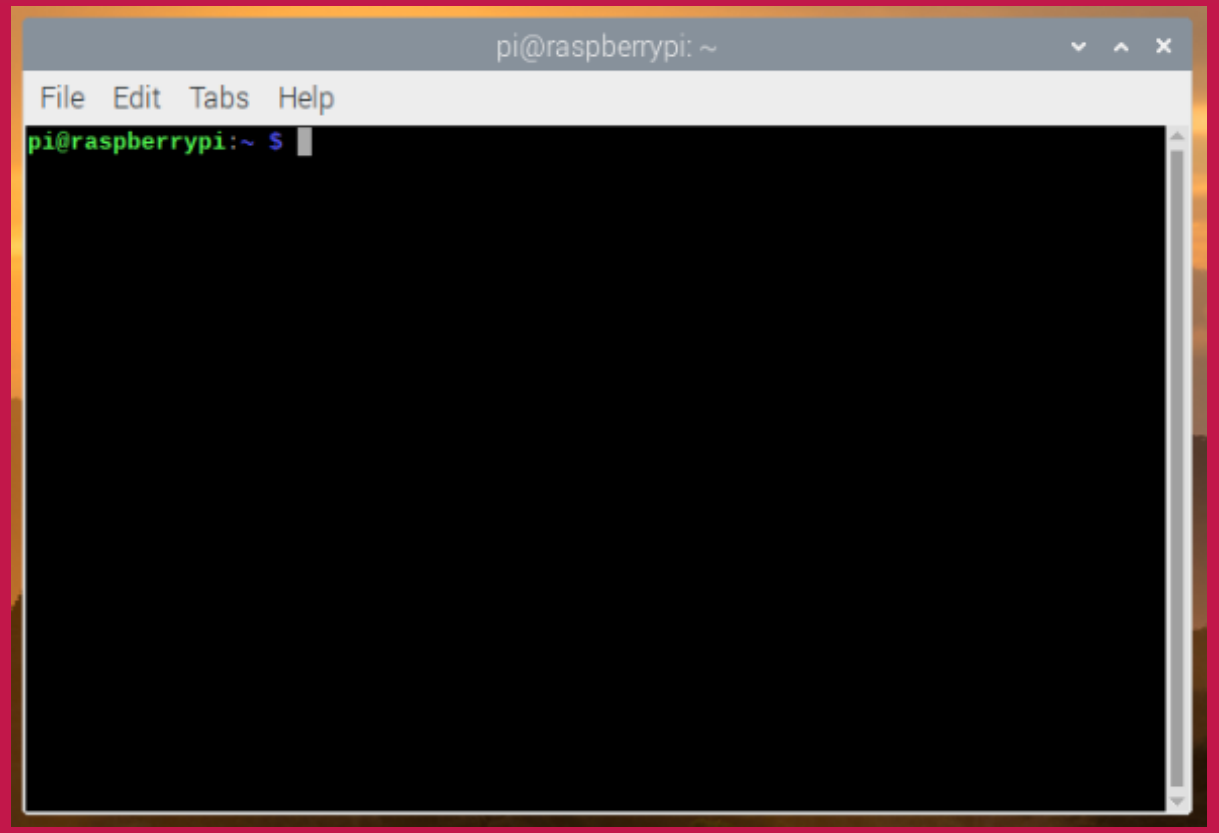


# Terminal

Often shown in tutorials

Faster system management

Direct execution of scripts (programs)



# Terminal

pwd

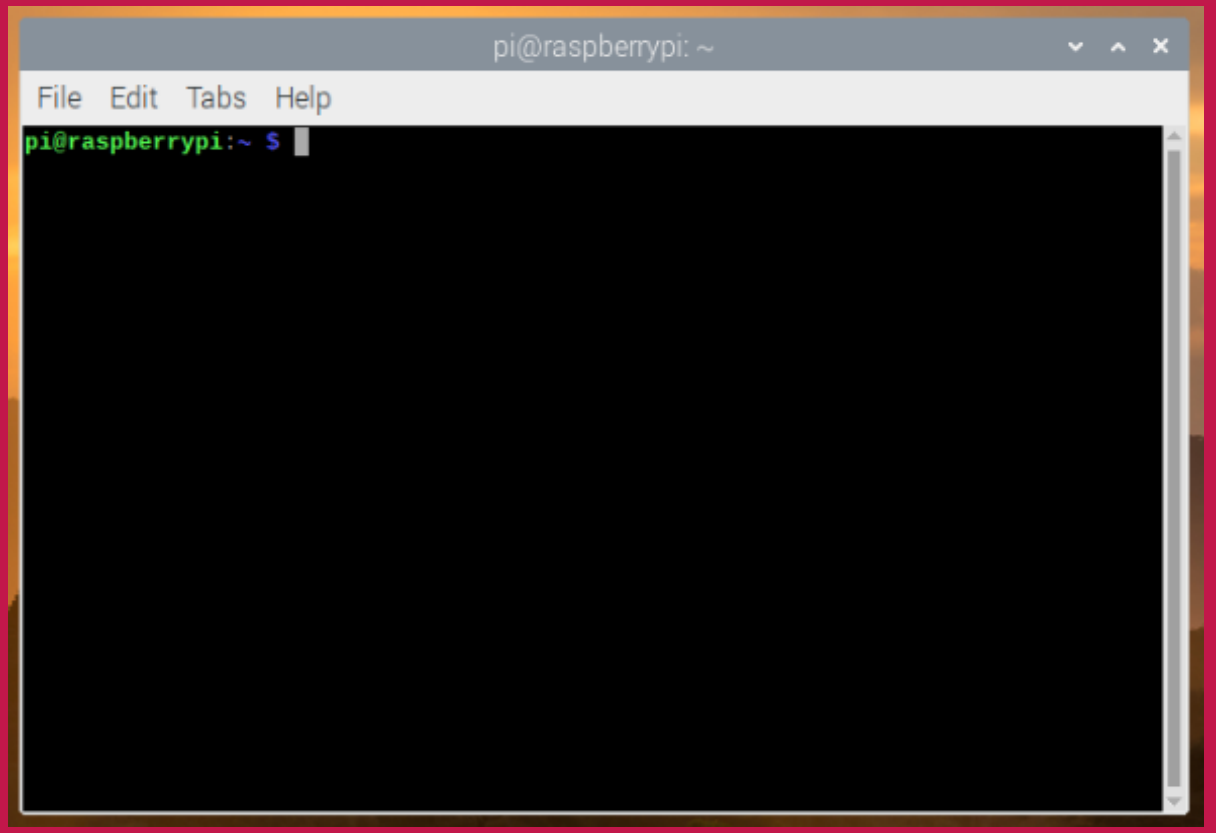
ls

cd

mkdir

nano

chmod

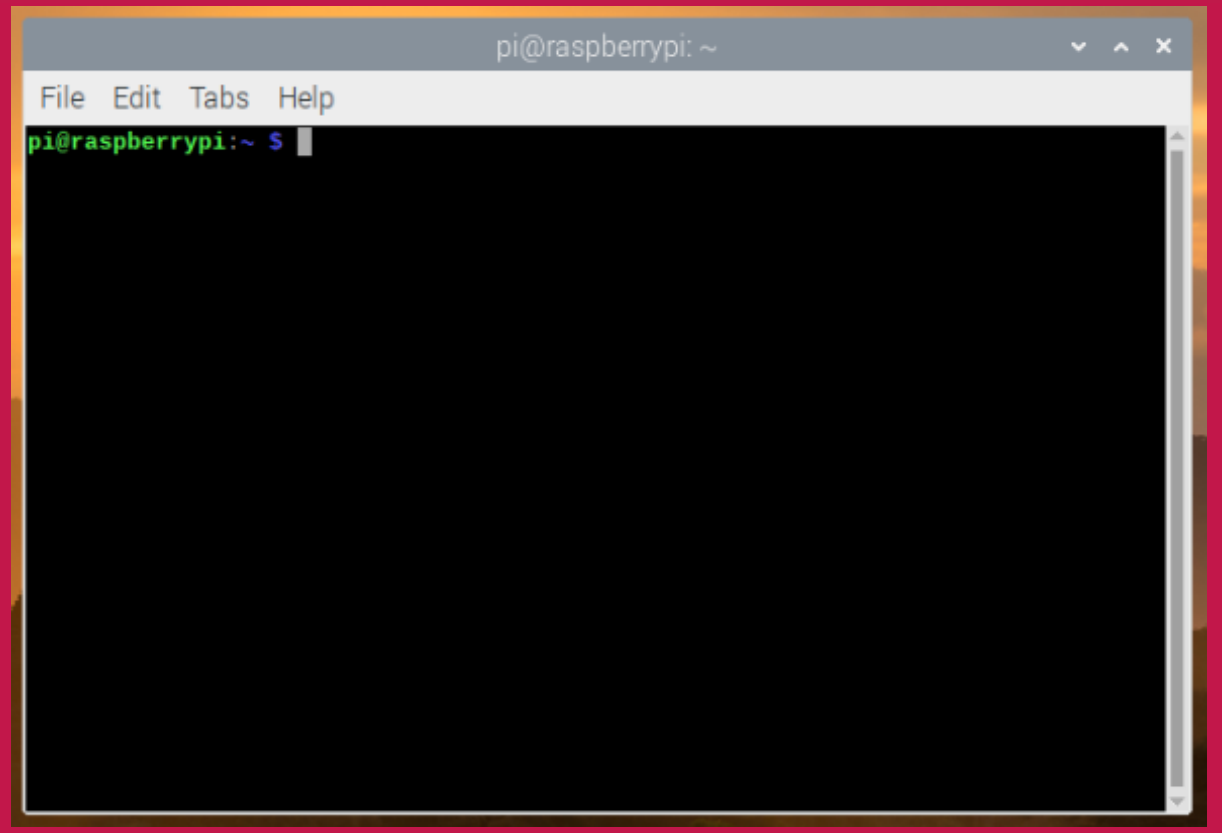


# Terminal

mkdir *<yourname>*

cd *<yourname>*

nano helloword.sh



# Nano

echo "hello world"

# Nano

echo "hello world"

<ctrl> o

<enter>

# Nano

echo "hello world"

<ctrl> o

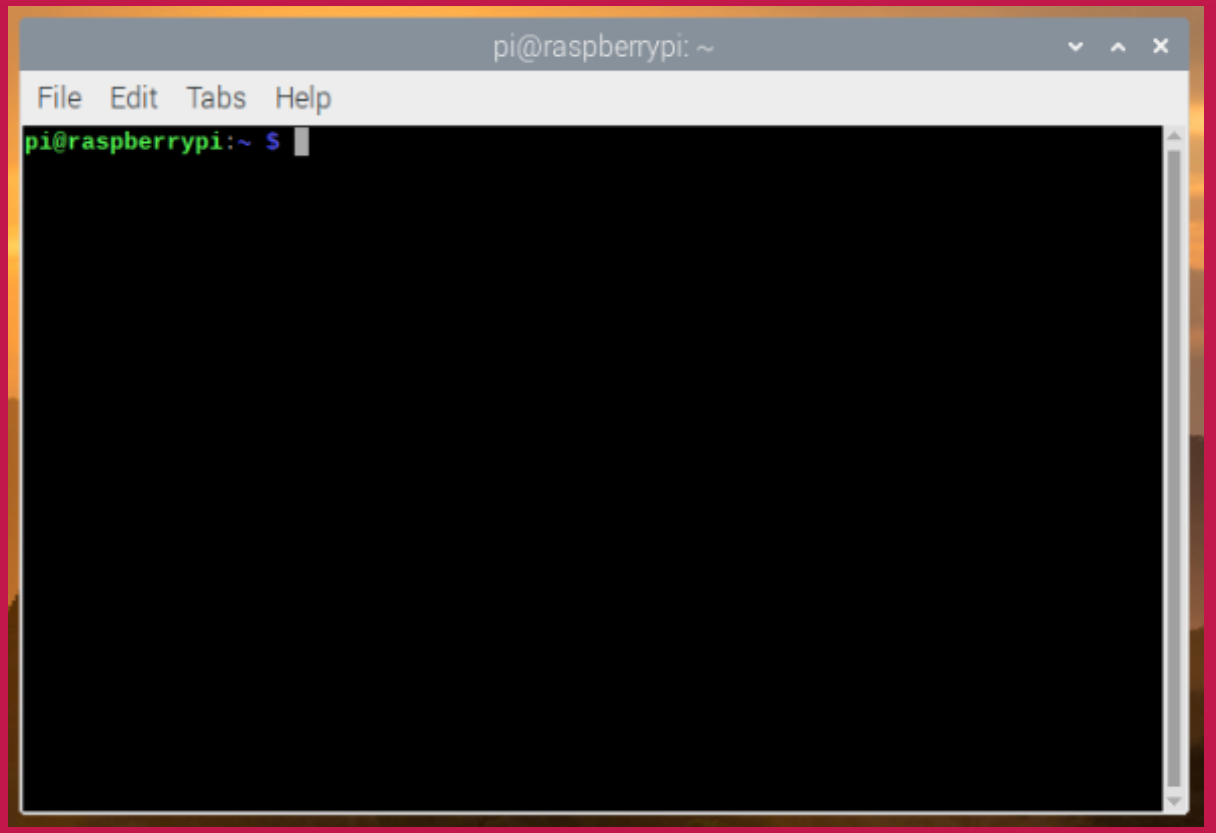
<enter>

<ctrl> x

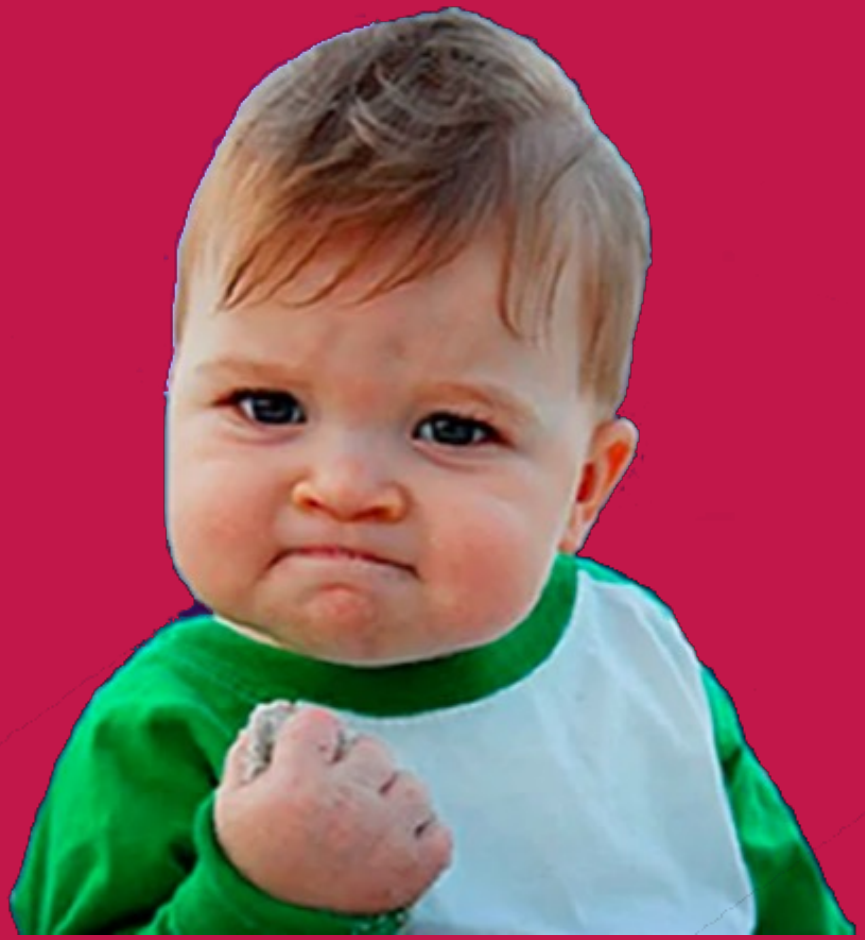
# Terminal

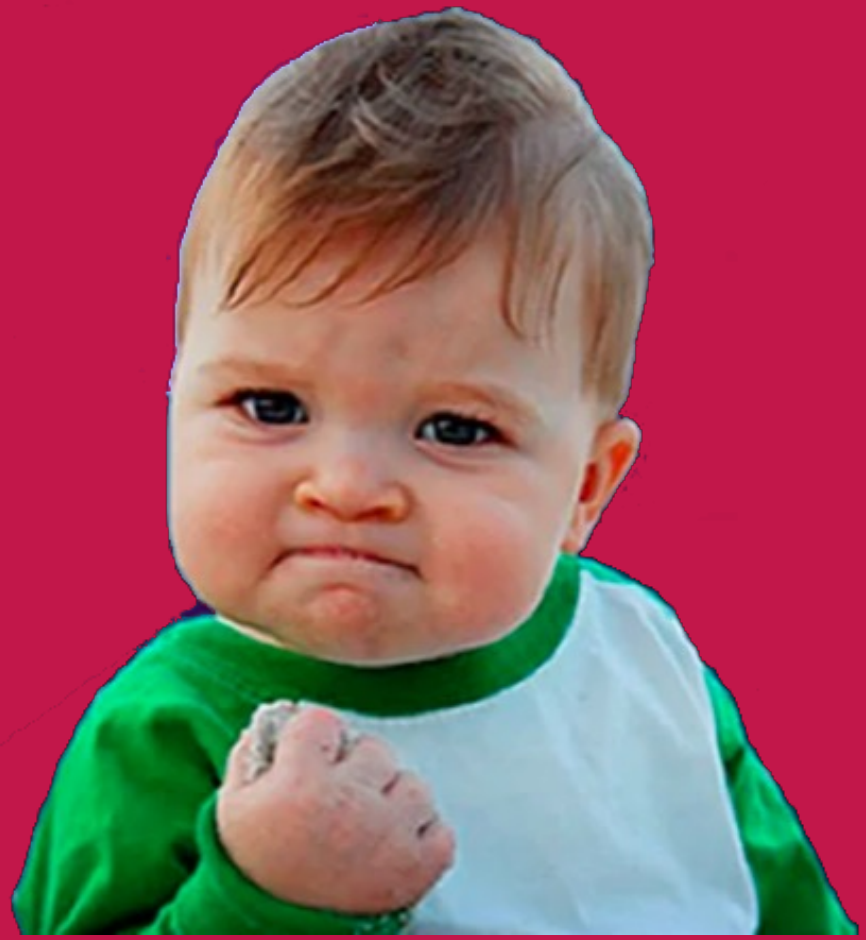
chmod 777 helloworld.sh

./helloworld.sh



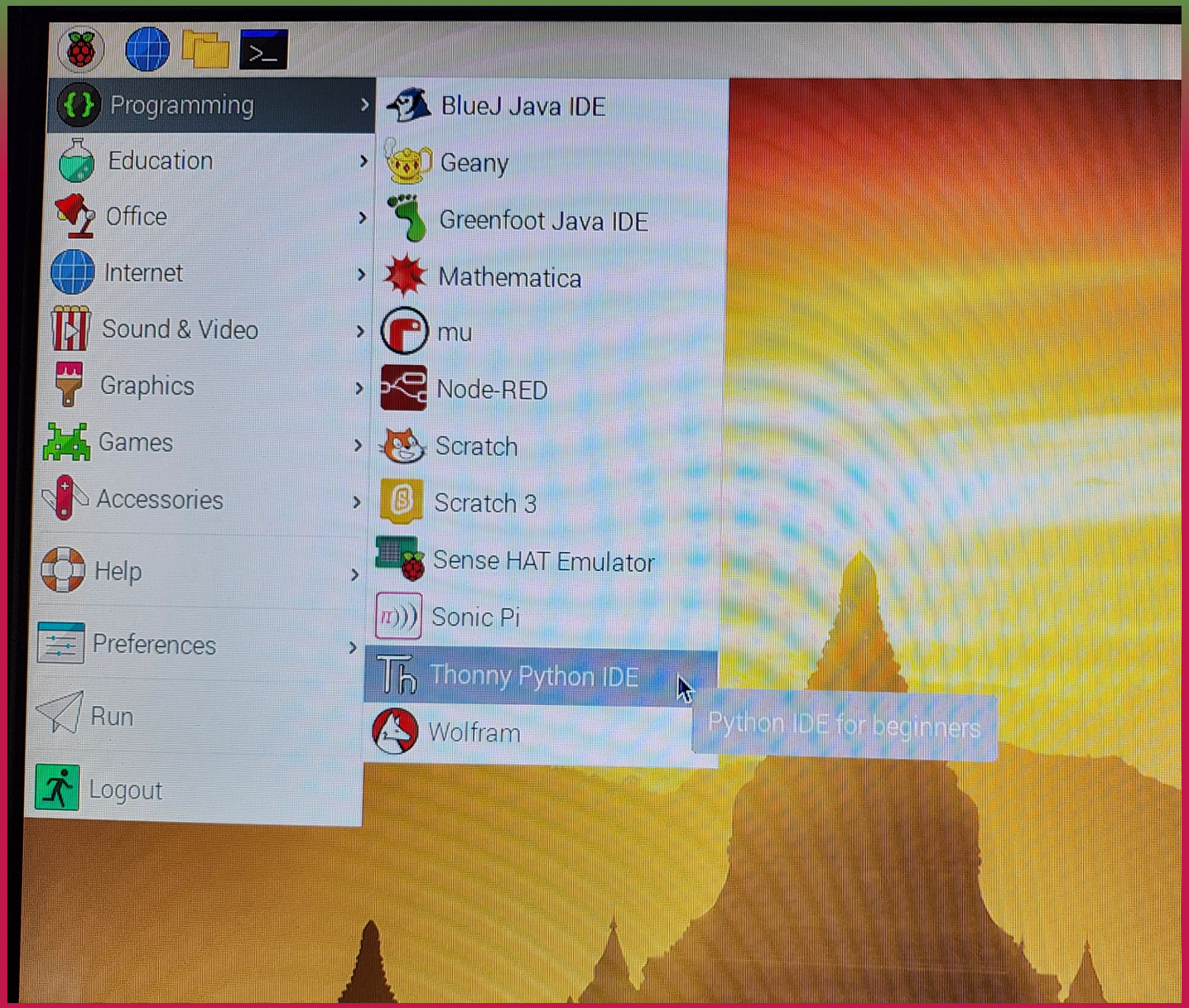
# Most Excellent!





# Python

Using Thony IDE



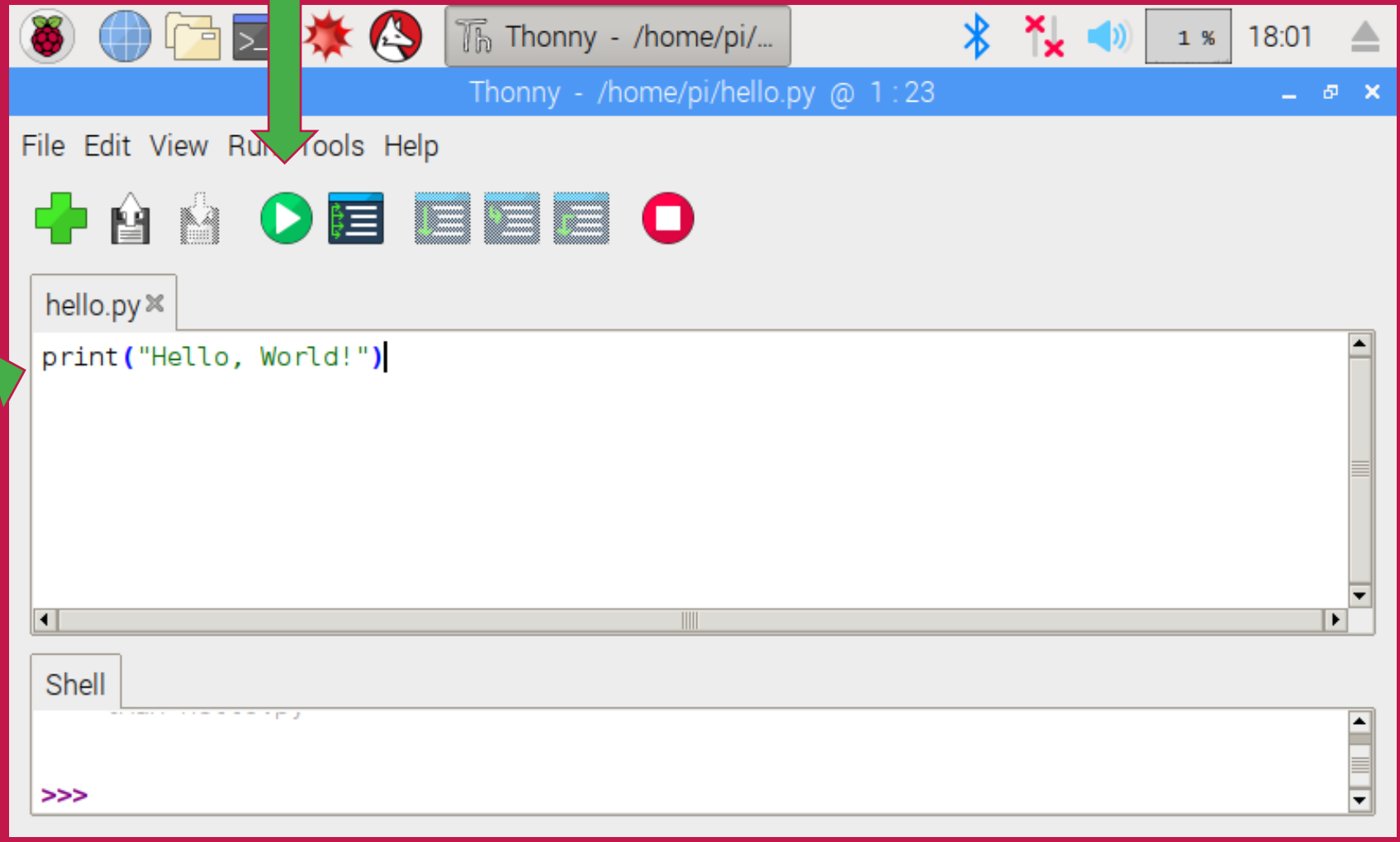
# Python

Type Program

Click Run Button

Run

Program

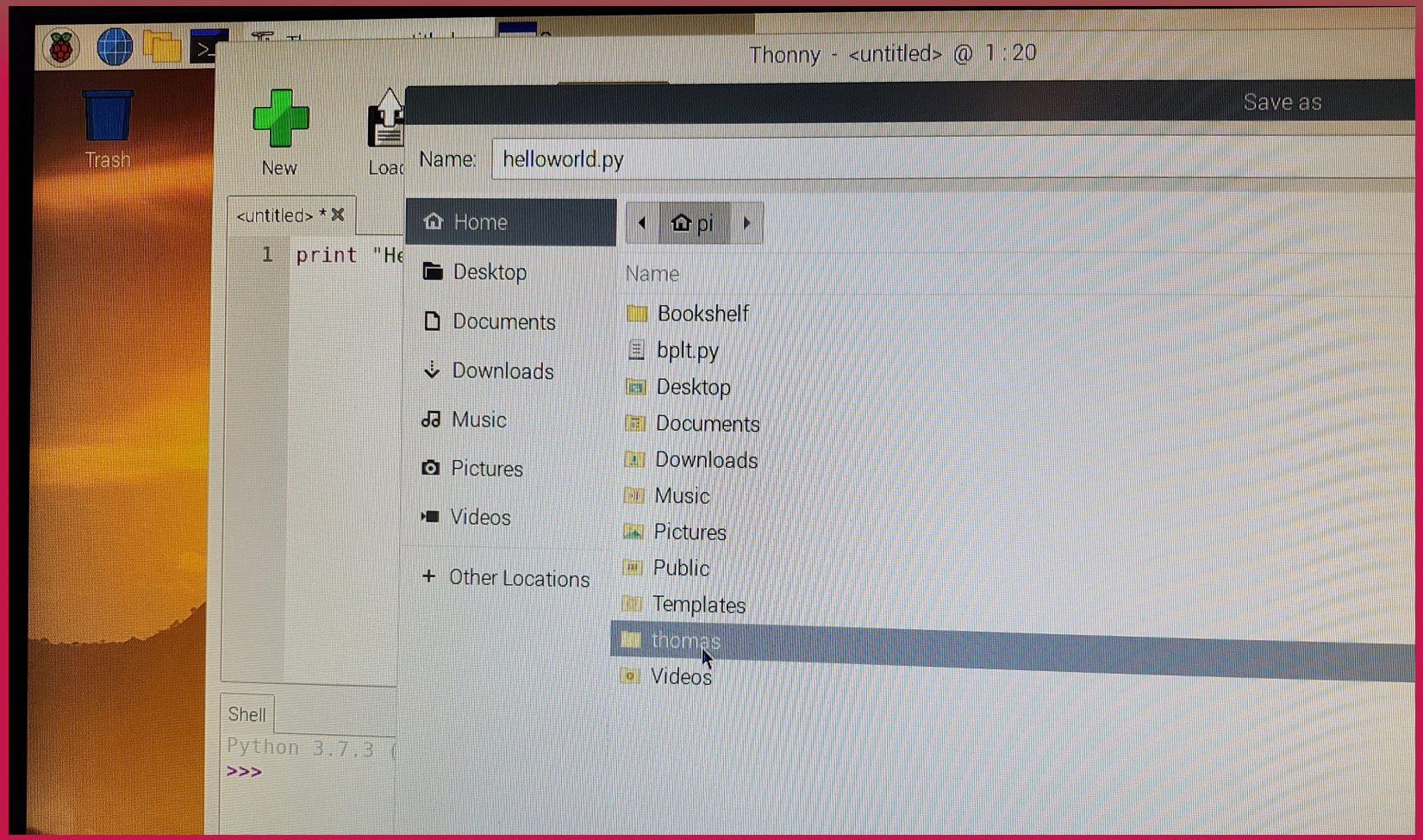


# Python

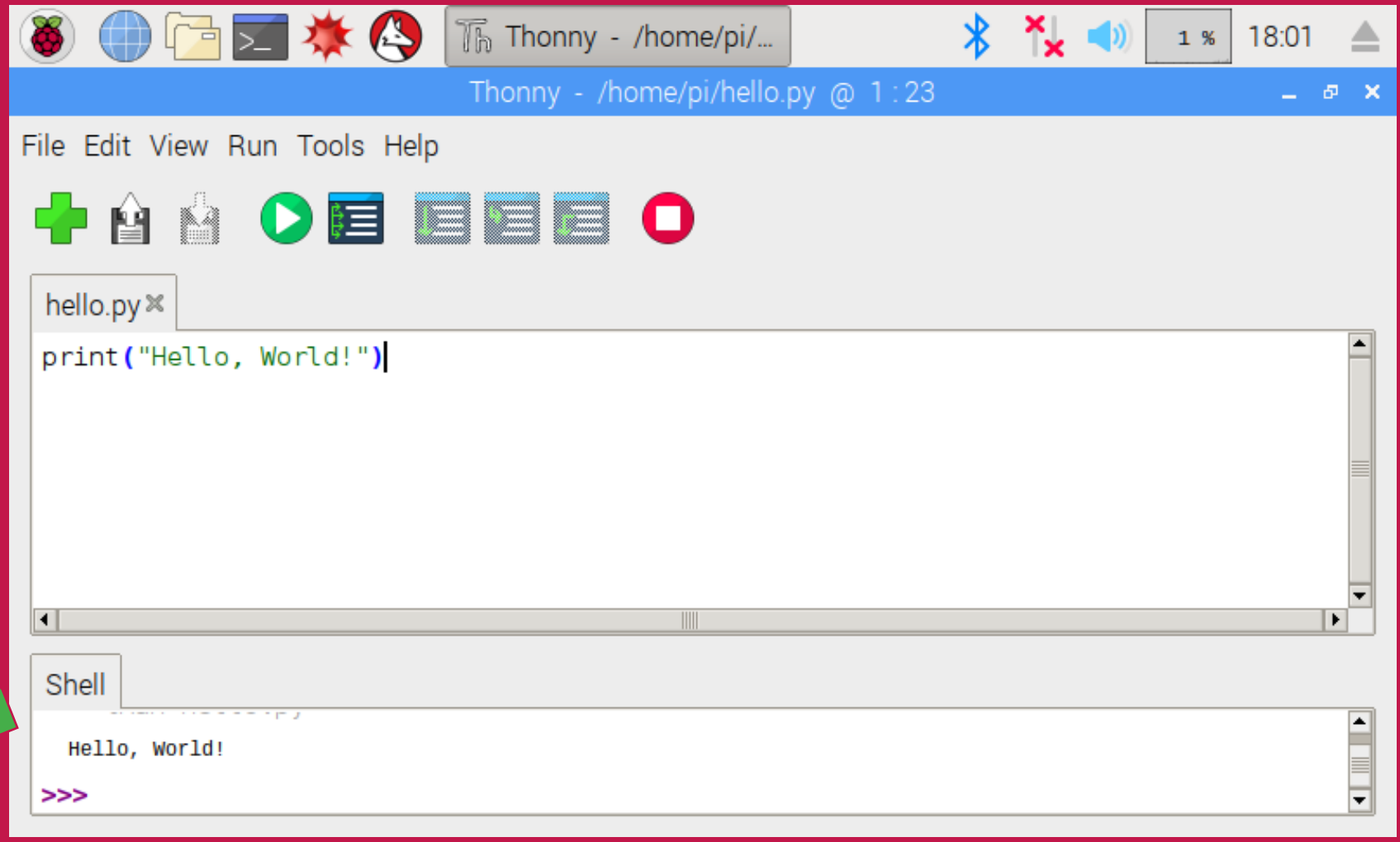
Name File

Select Folder

Click OK button



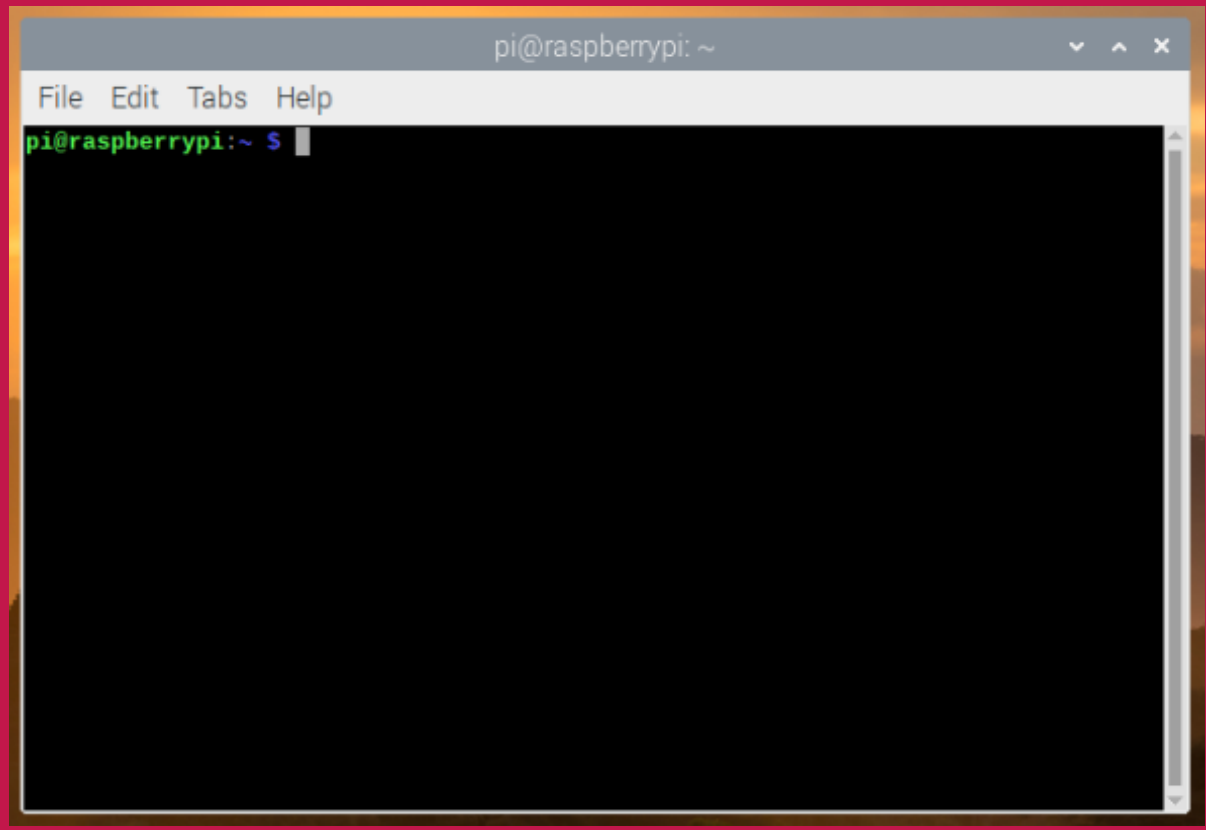
# Python



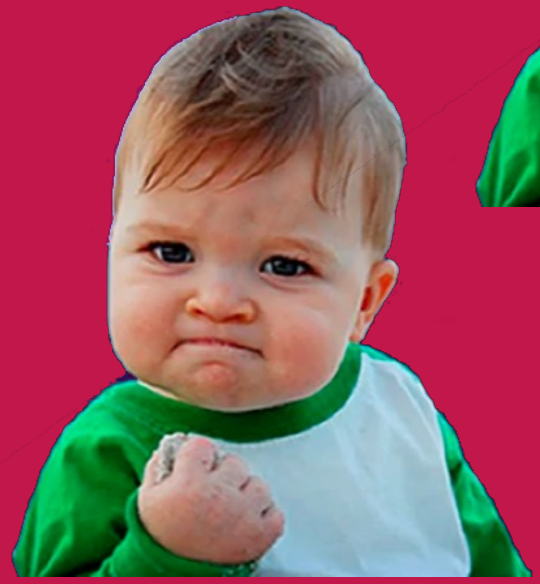

Program output

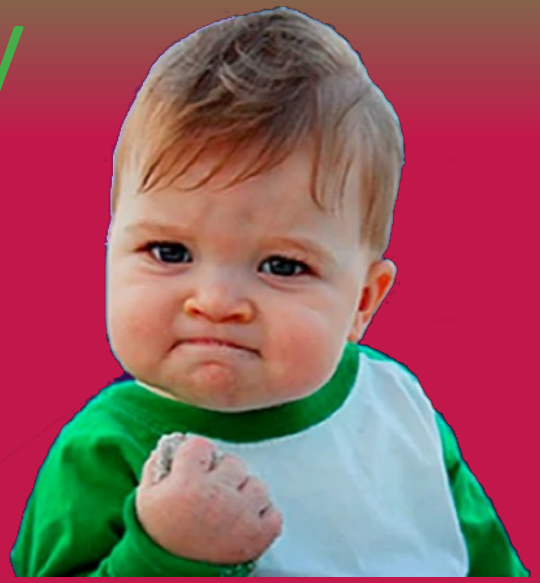
# Python via Terminal

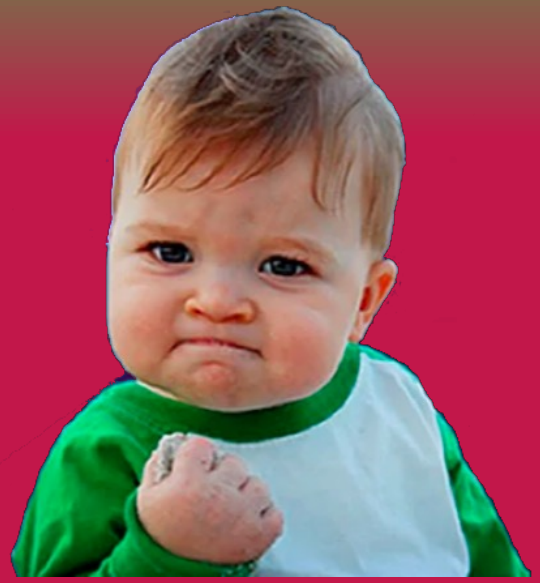
python ./helloworld.py



# Awesomly Excellent!



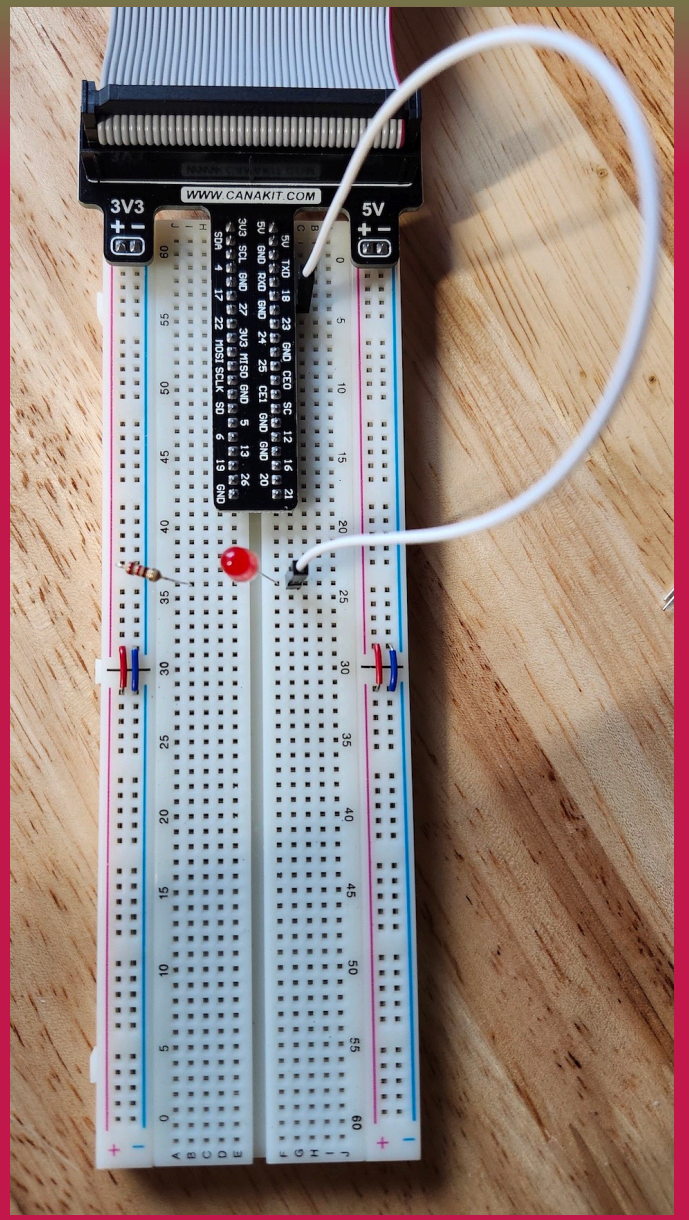




# GPIO LED

Hardware

Software



# https://trickel.org/thomas/skc/RPi.html

## 2024 Raspberry Pi Workshop

### Raspberry Pi

- 2024 Raspberry Pi Worksshop Slide Deck
- CanaKit Raspberry Pi Quick Start Guide
- Blink LED Python code
  - Image of LED on Breadboard
- Switch Controlled LED Python code
  - Image of Switch on Breadboard

### Internet of Things

- Welcome to Adafruit IO
- Digital Output from AIO
- Digital Input to AIO

### Useful Links

- Raspberry Pi Foundation Web Site
- **Adafruit IO**
  - Adafruit IO Basics: Feeds
  - Adafruit IO Basics: Dashboards
  - Adafruit IO Blocks

# GPIO LED - Hardware

Digital Output

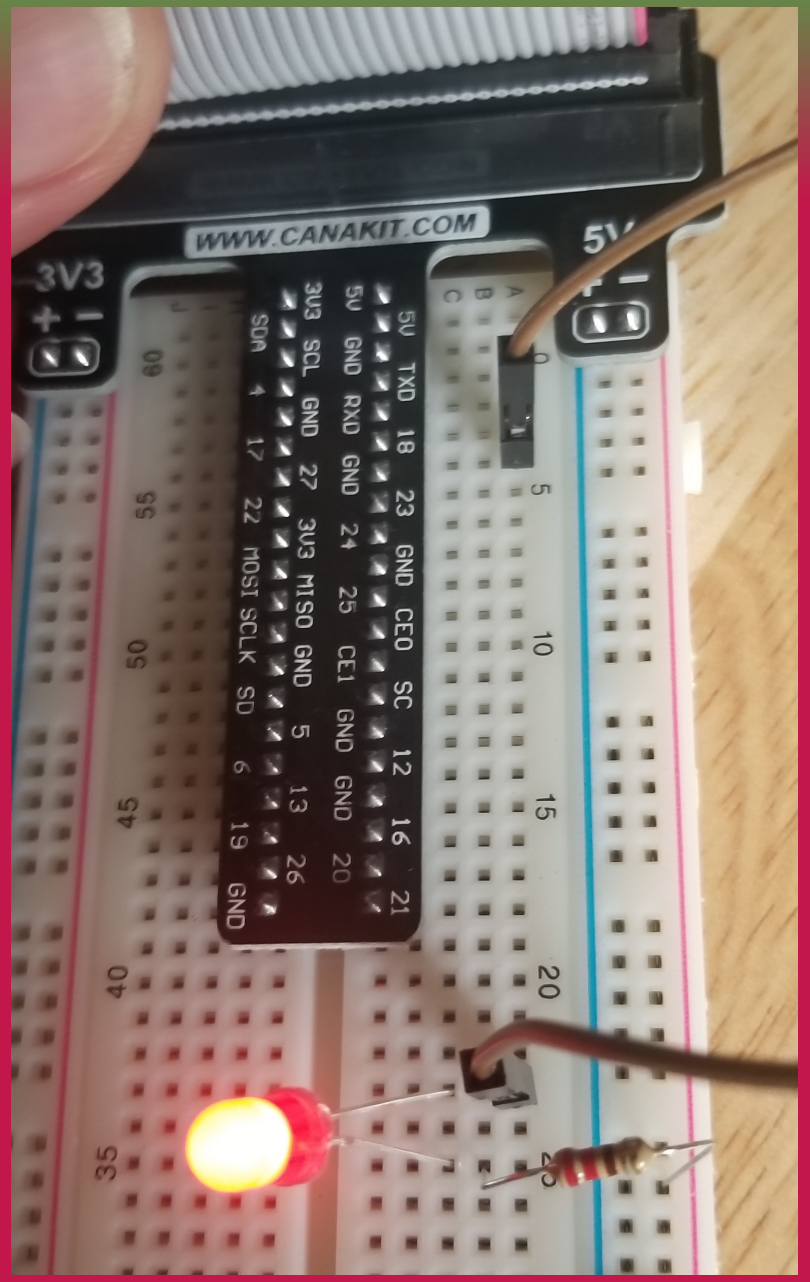
Polarity matters on LED

- Short leg is Cathode, connect to Resistor to - (GND)

Resistor 220Ω (Red Red Brown)

Wire

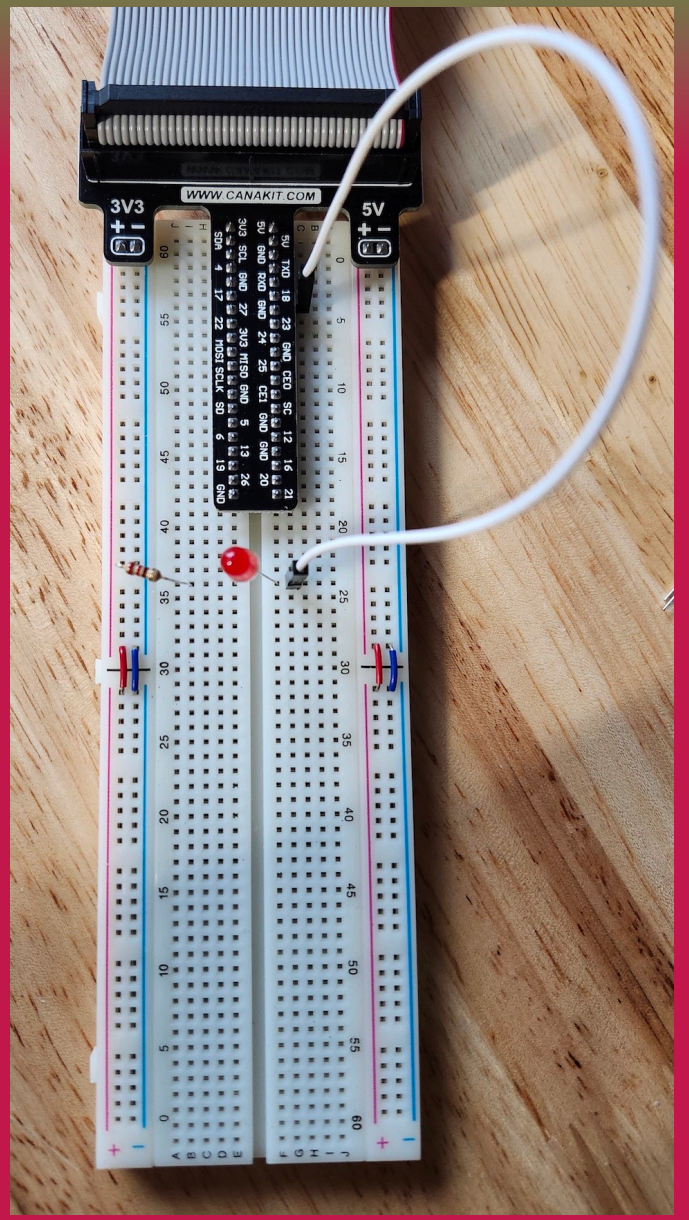
- LED Anode to GPIO 18



# GPIO LED - Software

Thony

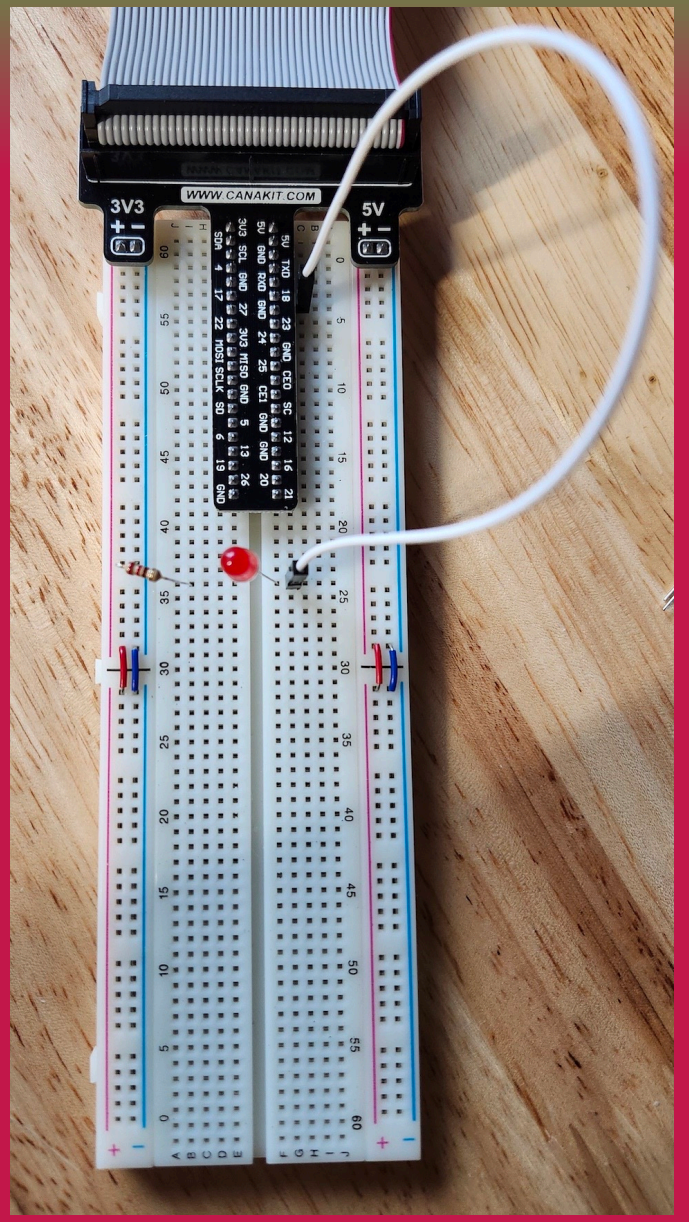
```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
    GPIO.output(18, True)
    time.sleep(1)
    GPIO.output(18, False)
    time.sleep(1)
```



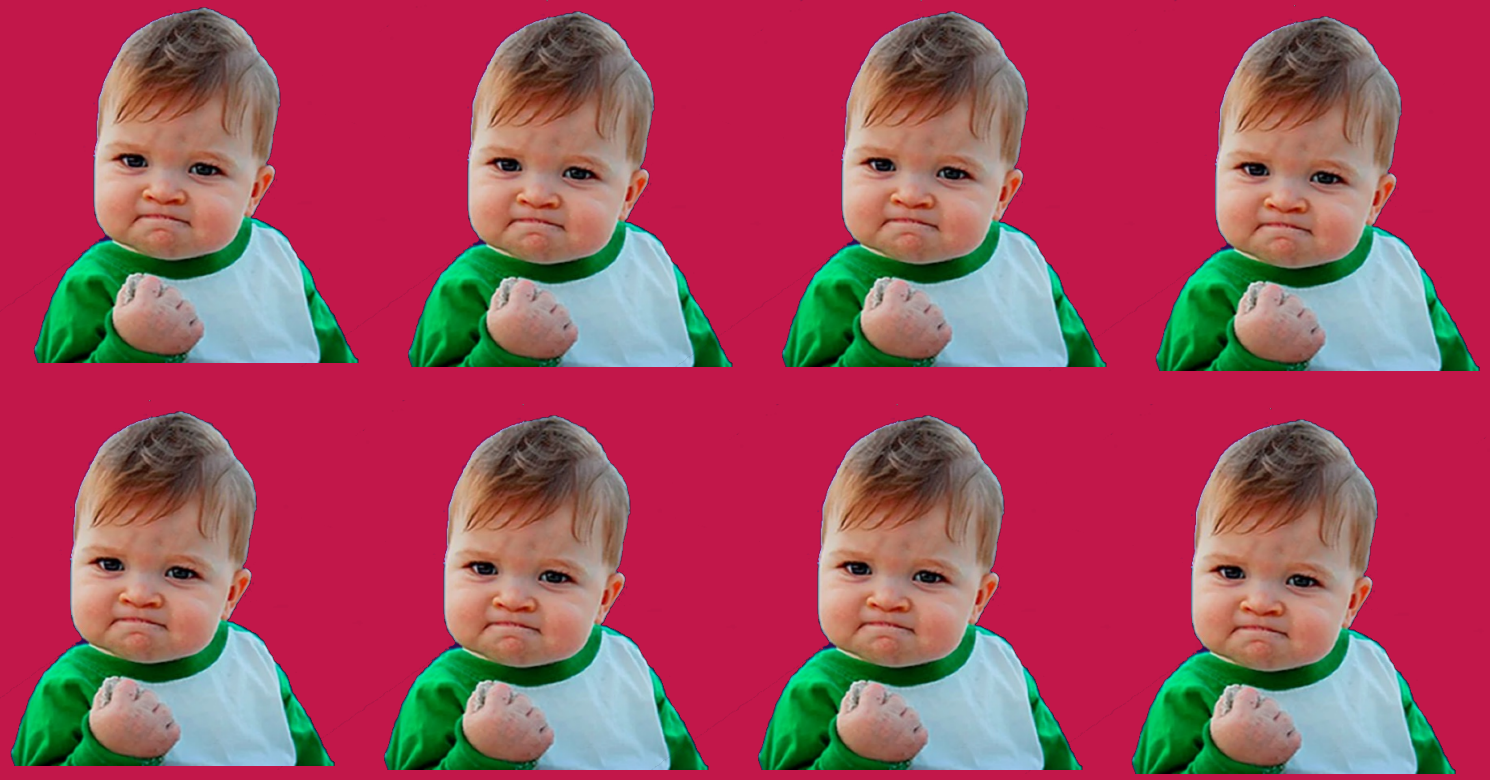
# Make LED Blink Faster

Thony

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
    GPIO.output(18, True)
    time.sleep(1)
    GPIO.output(18, False)
    time.sleep(1)
```



# Most Awesomly Excellent!




BREAK
TIME !!

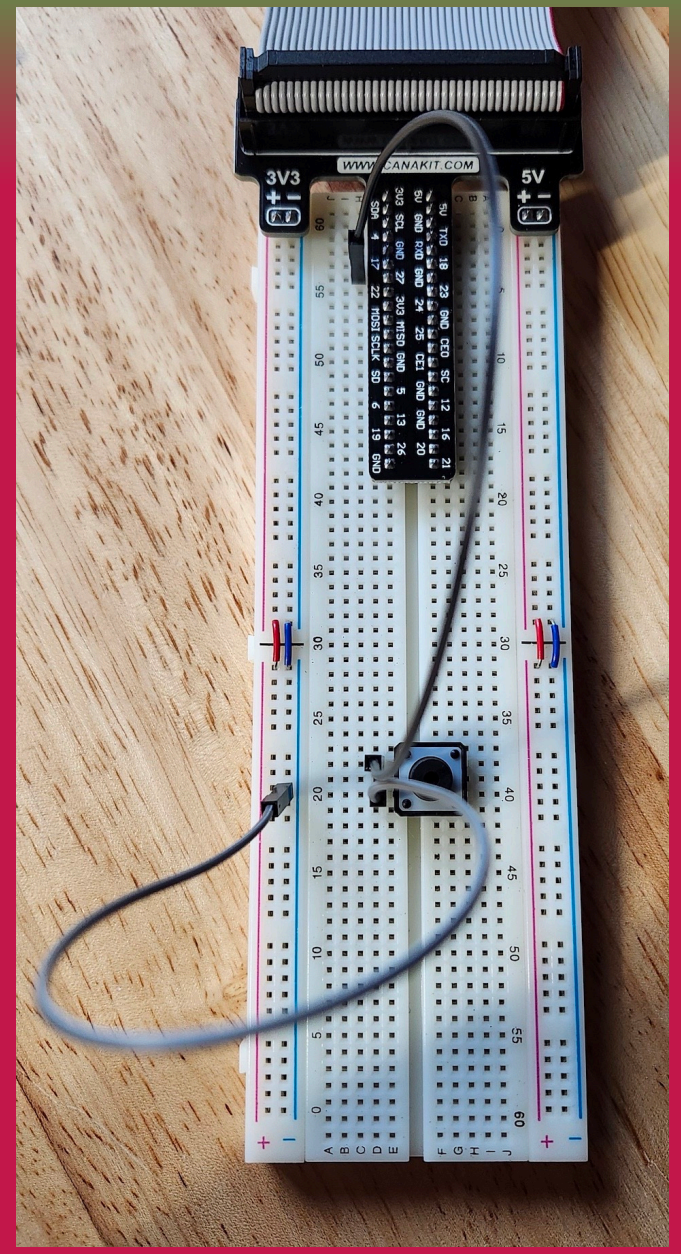
# GPIO Switch

Hardware

Software

Pressing switch will turn on LED

LED will turn off when switch is released



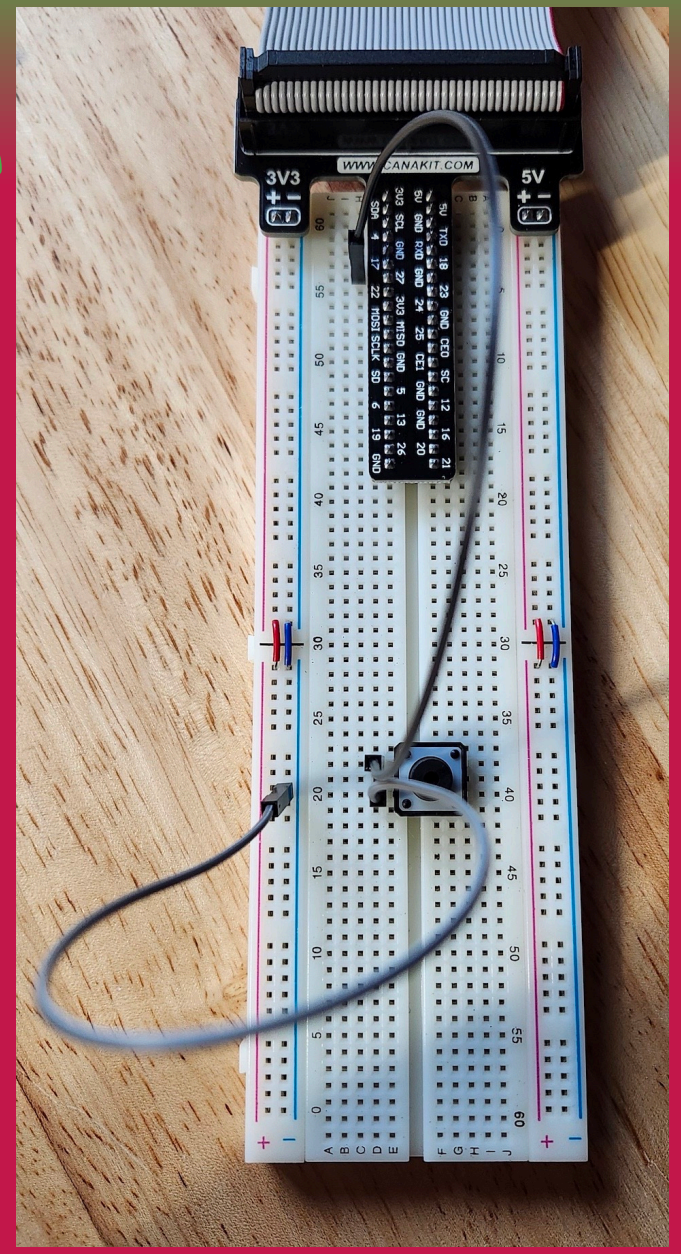
# GPIO Switch - Hardware

Digital Input

Polarity does NOT matter on Switch

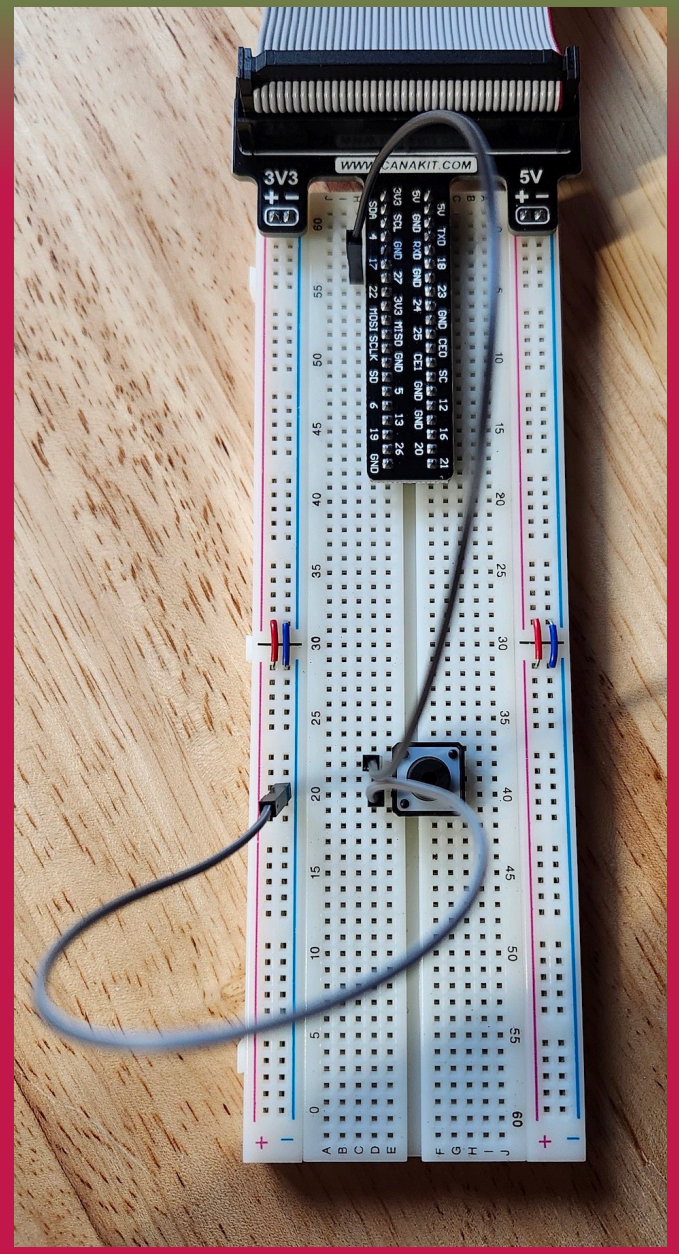
Wires

- GPIO 17
- - (GND)

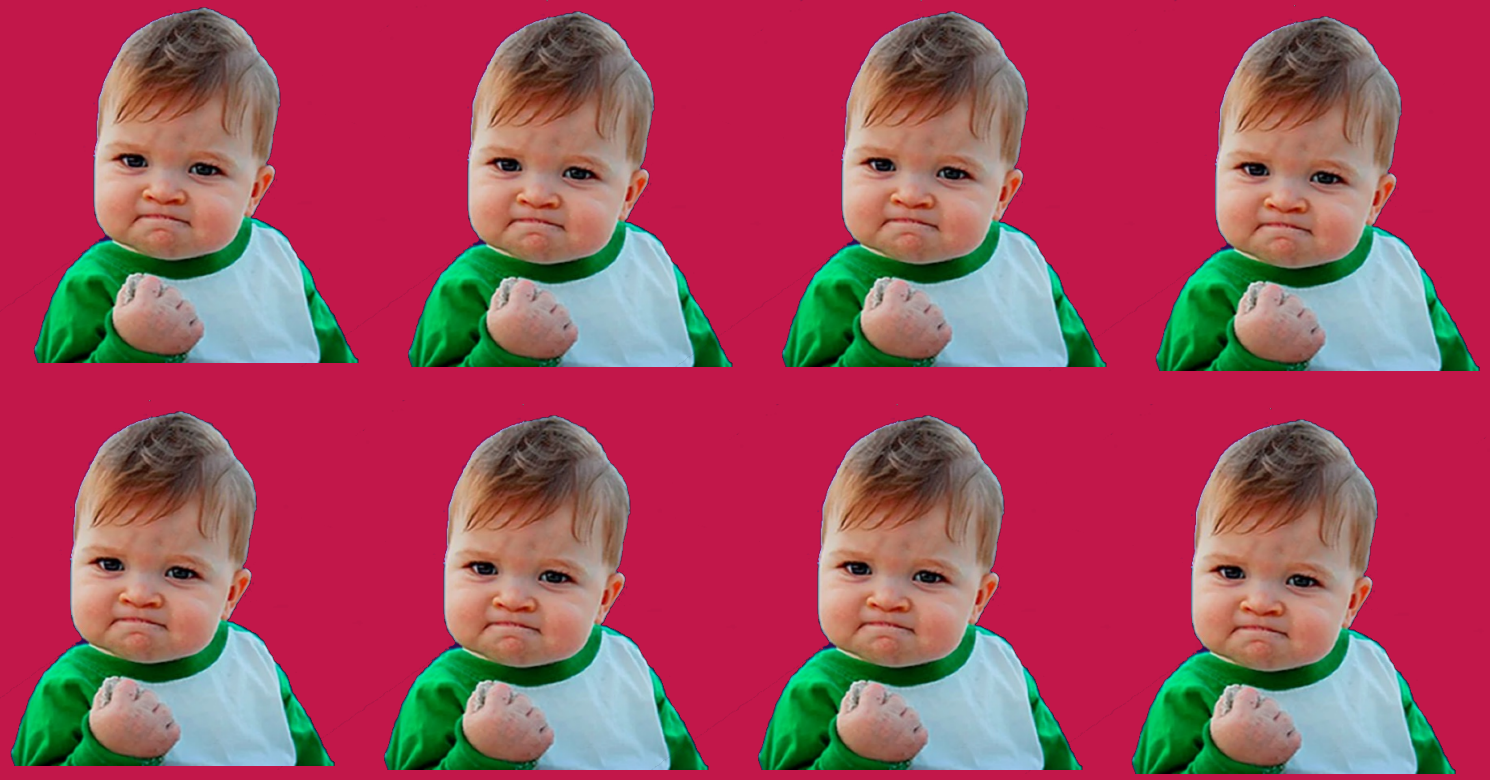


# GPIO Switch - Software

```
#SwitchControlledLED.py
#
#note:
#       button is negative logic i.e. high (True) when not pressed
#       thus turning off (False) the LED when GPIO.input(17) is True (high)

import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN)

while True:
    if GPIO.input(17, True)
        GPIO.output(18, True)
    else:
        GPIO.output(18, False)
```



# Most Awesomly Excellent!



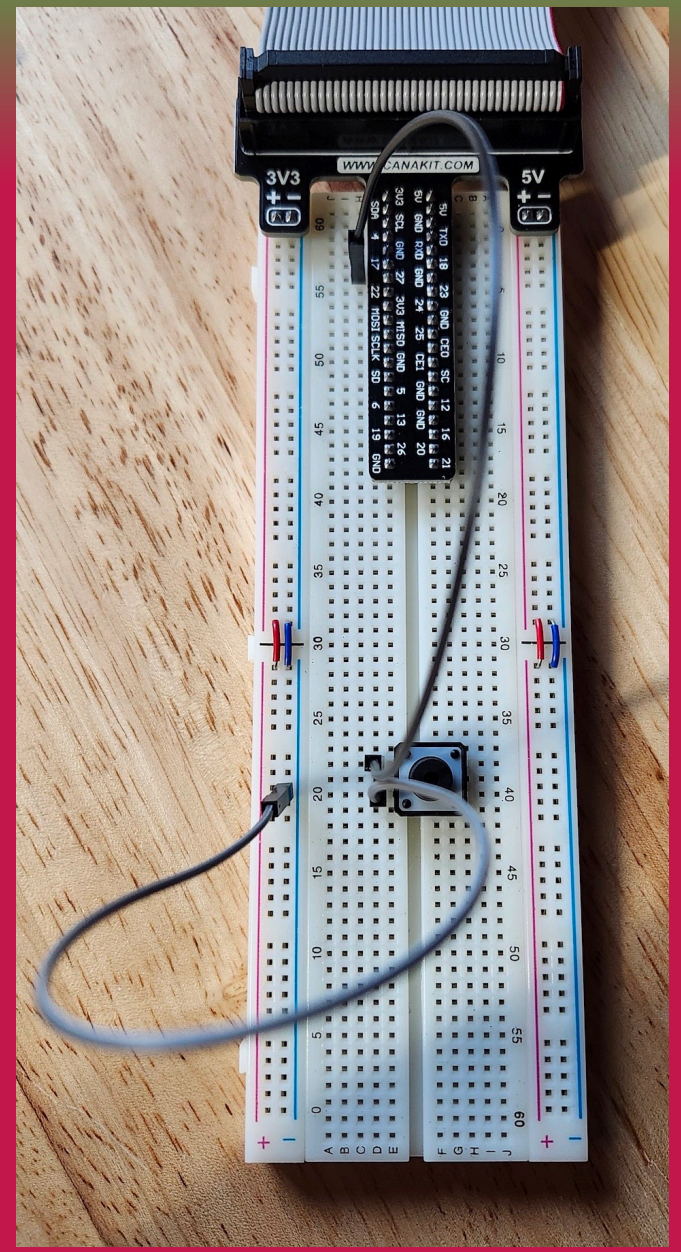
# File Write

Hardware

Software

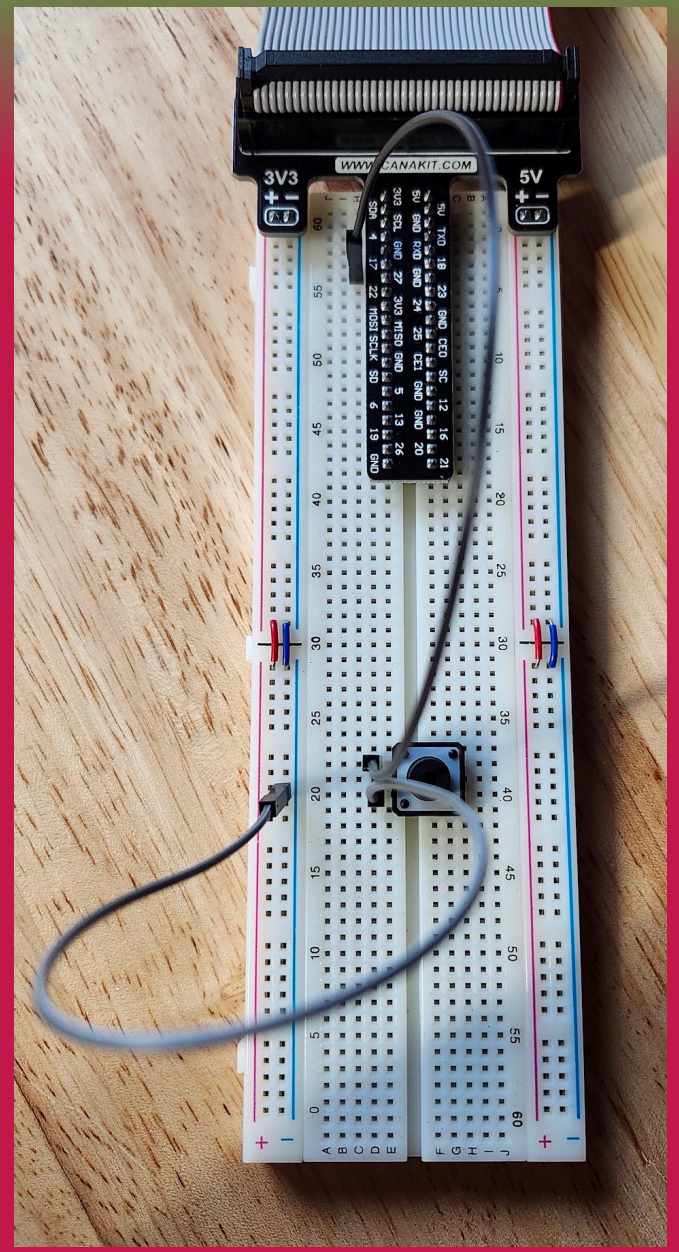
Pressing switch will

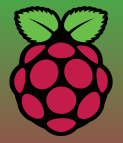
write current time to file



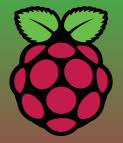
# GPIO Switch - Hardware

Same





```
import RPi.GPIO as GPIO
import time

# Define the GPIO pin for the switch
BUTTON_PIN = 17

#Define a path and name for the file the time switch presses will be logged to
LOG_FILE = "/home/pi/button_time_log.txt"

# Set up GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(18, GPIO.OUT)

# Function to log the time when the button is pressed
def log_time():
    file = open(LOG_FILE, "a")
    current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
    file.write(f"Button pressed at {current_time}\n")
    print(f"Button pressed at {current_time}")
```



```
GPIO.output(18, False)

# Main loop to wait for button press

print("Waiting for button press...")
while True:
    # Detect button press (falling edge, when button is pressed)
    if GPIO.input(BUTTON_PIN) == GPIO.LOW:
        log_time()
        GPIO.output(18, True)

    # Wait for button to be released
        while GPIO.input(BUTTON_PIN) == GPIO.LOW:
            time.sleep(0.1)
            time.sleep(0.1)
            GPIO.output(18, False)
```

# GPIO Switch

Use terminal to view file

```
cd ~
ls
more button_time_log.txt
```

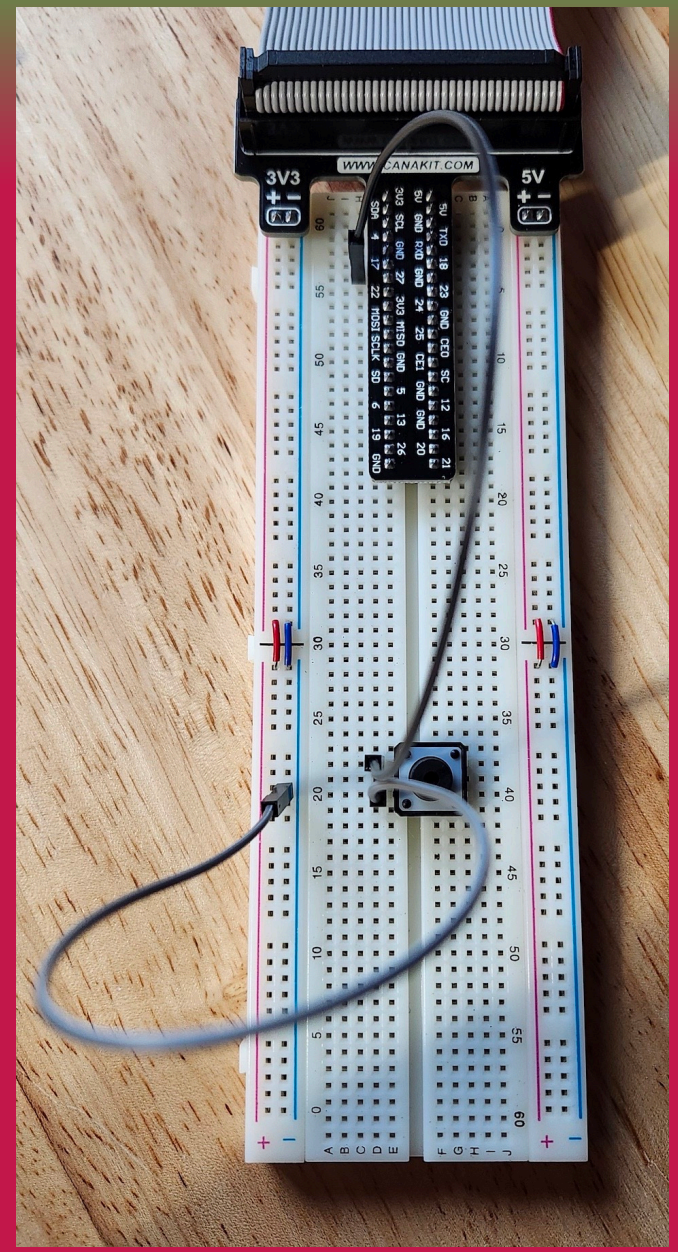
# Superbly Excellent!

# Cronjob Alarm

Hardware

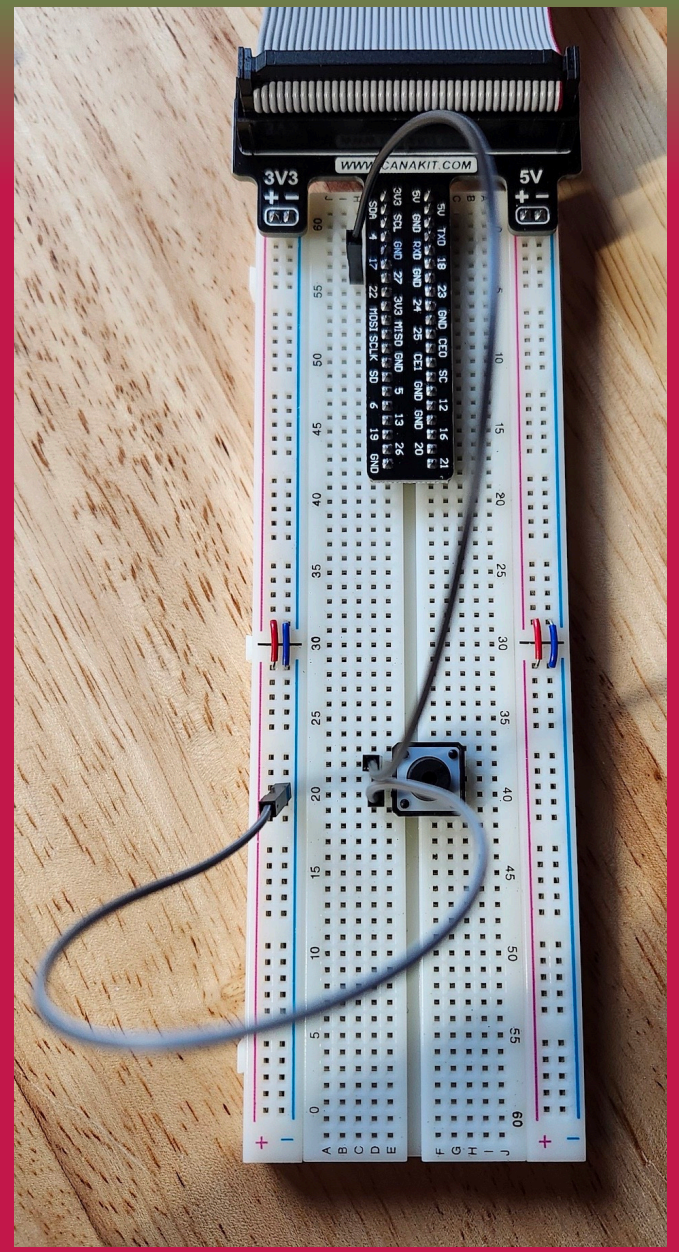
Software

At a specific time blink an LED until a switch is pressed



# Cronjob - Hardware

Same



# Cronjob - Software

BlinkLEDuntilButtonPress.py

alarm.sh

Create cronjob

# Cronjob – Software

BlinkLEDuntilButtonPress.py

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(17, GPIO.IN)

print(GPIO.input(17))

while GPIO.input(17) == True:
    print("here")
    GPIO.output(18, True)
    time.sleep(1)
    GPIO.output(18, False)
    time.sleep(1)
```

# Cronjob – Software

## alarm.sh

Use Nano editor to create file with one line

nano alarm.sh

python /home/pi/thomas/BlinkLEDuntilButtonPress.py

<ctrl>o
<ctrl>x

# Cronjob – Software

## cronjob

In terminal

chmod 777 alarm.sh

crontab -e

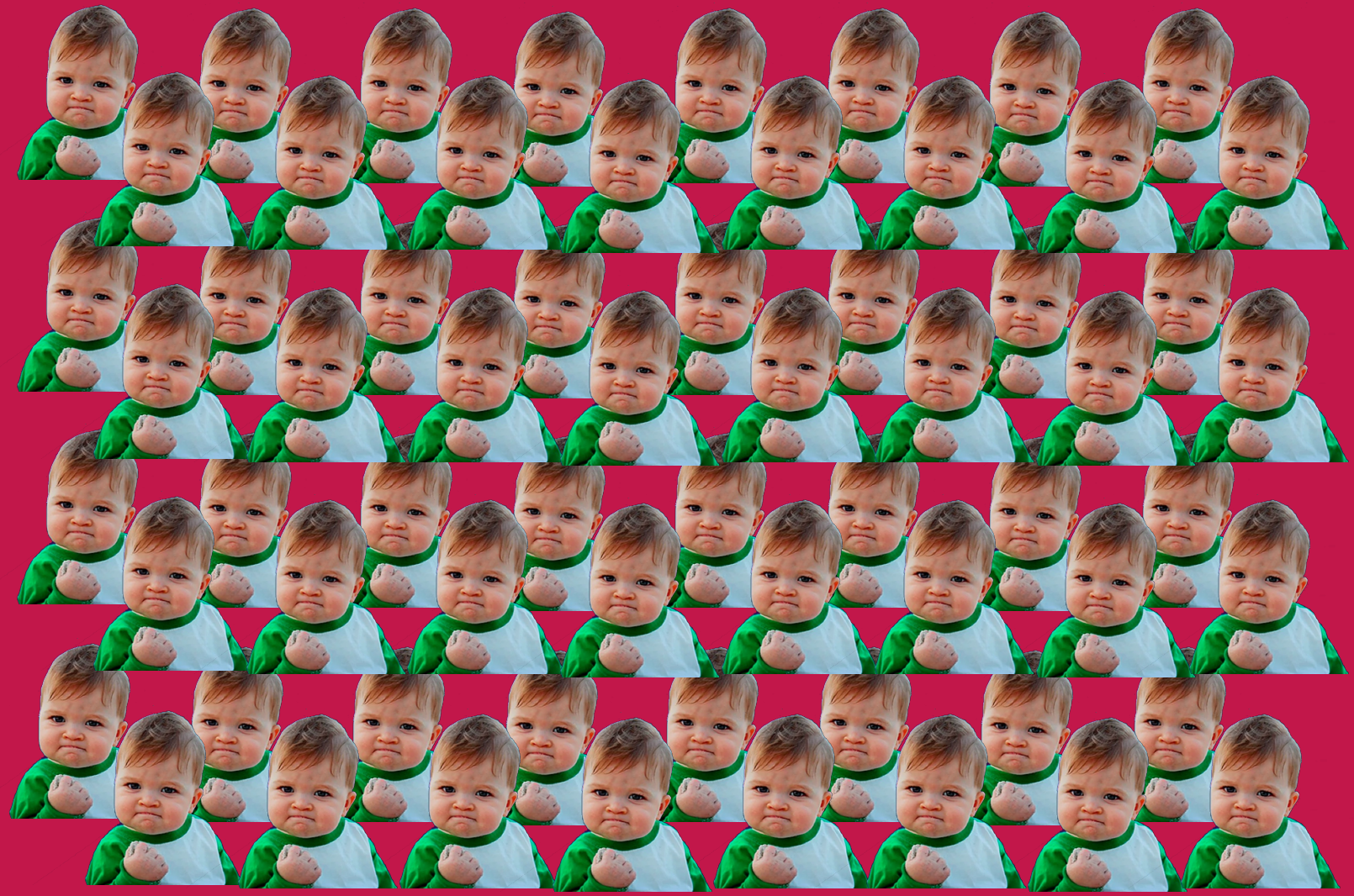
```
# Edit this file to introduce tasks to be run by cron.
#
# ...
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

Add this line 

15 13 * * * /home/pi/<yourname>/alarm.sh

# Cronjob

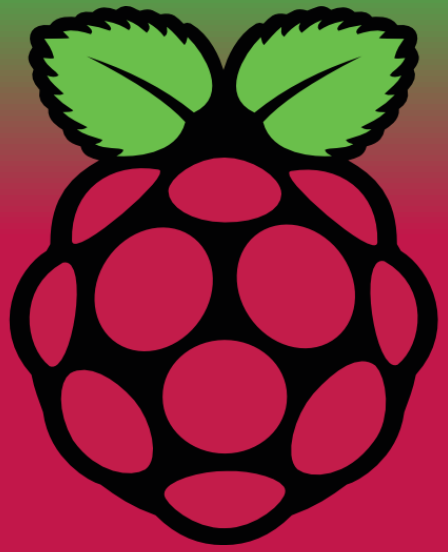
# Unbelievably Excellent!

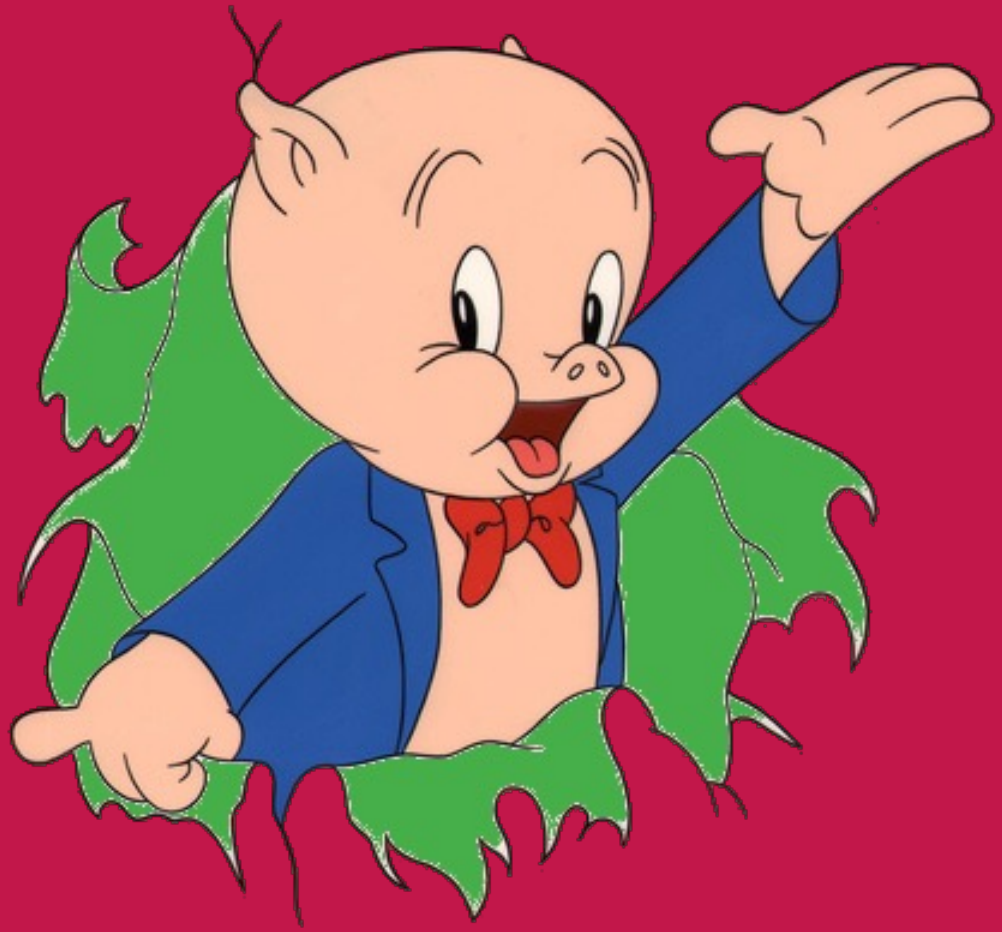


# 2024 Raspberry Pi Workshop

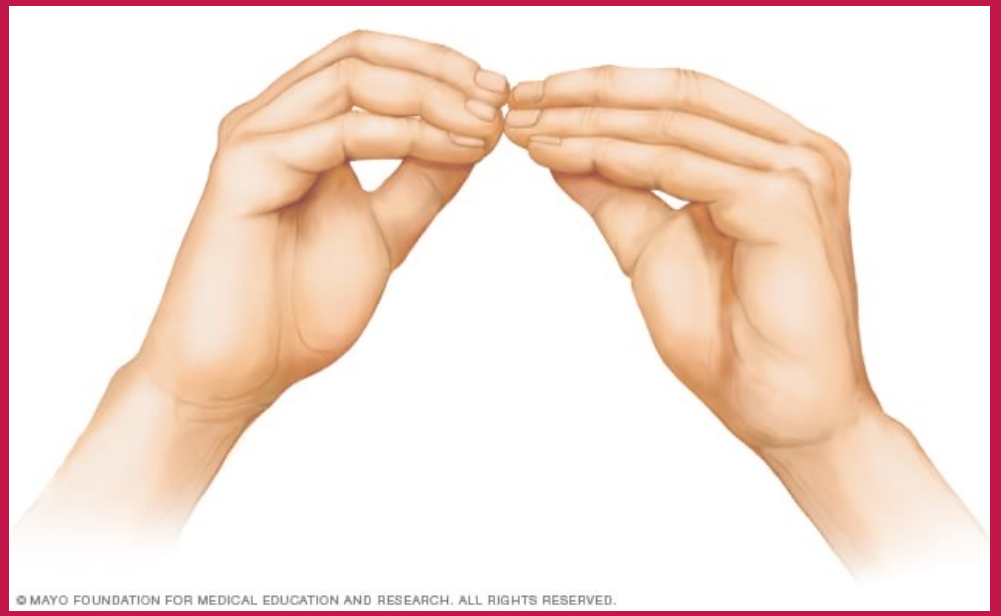
# 2024 Raspberry Pi Workshop

# 2024 Raspberry Pi Workshop





# More


© MAYO FOUNDATION FOR MEDICAL EDUCATION AND RESEARCH. ALL RIGHTS RESERVED.


Internet of things
IOT

# Adafruit IO

- Internet of Things (IoT)
- Feeds and Dashboards
- Control LED from AIO (digital output)
- Send Data to AIO (digital input)
- Send CO2 data to AIO (analog input)

# Setup AIO Account

- Sign Up
- Two factor authentication

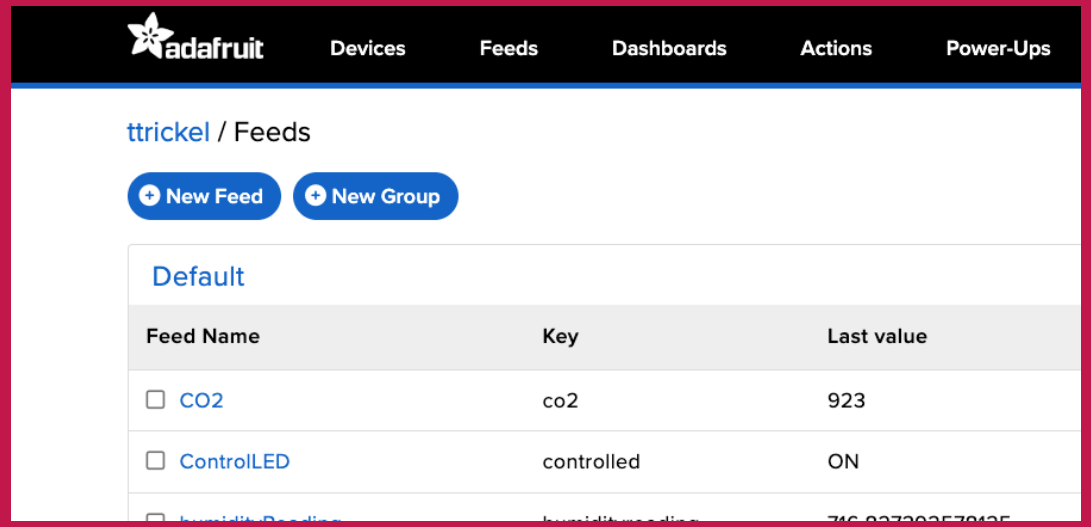
# Feeds & Dashboards

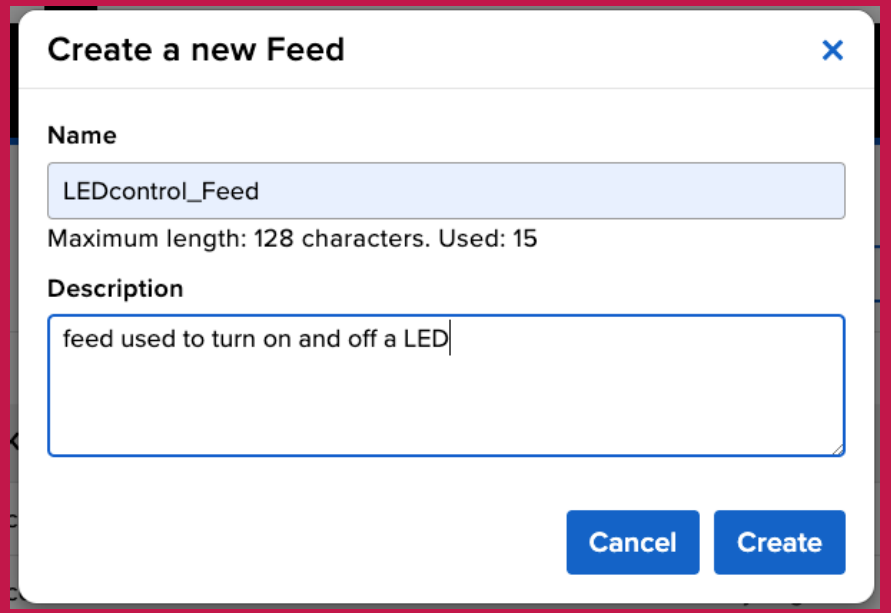
- Think of a Feed as property of an IoT device
  - Feeds are Features of an IoT device
    - Weather Station – temperature, humidity, wind speed, etc
    - Thermostat – set temperature, on/off
    - Door Lock
- Data is sent to and comes from Feeds
- Dashboards are used to
  - Display Feed data
  - Send data to a Feed

# Control LED from AIO

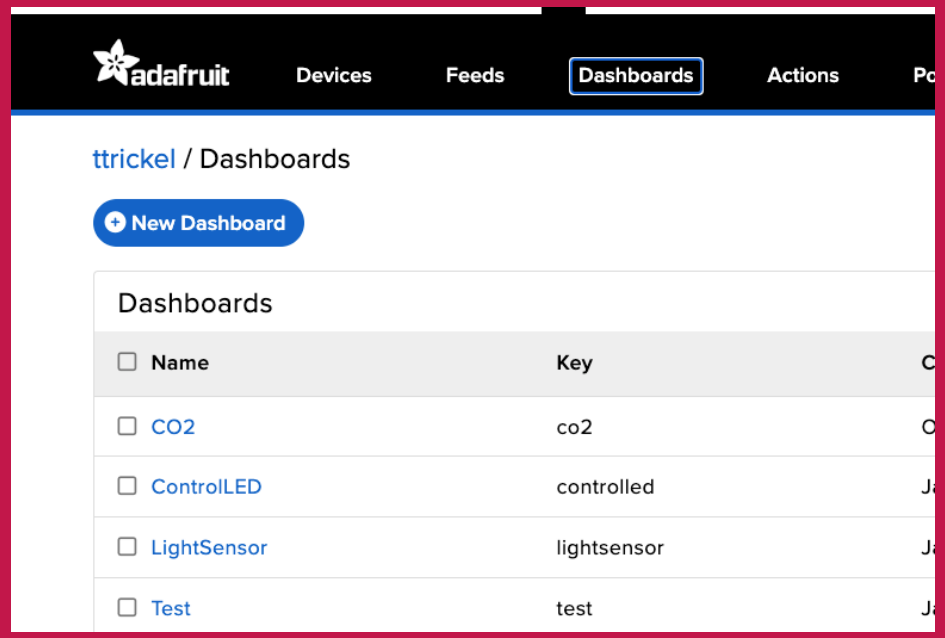
- Hardware is the same
- Create Feed
- Create Dashboard
- Write Python Code

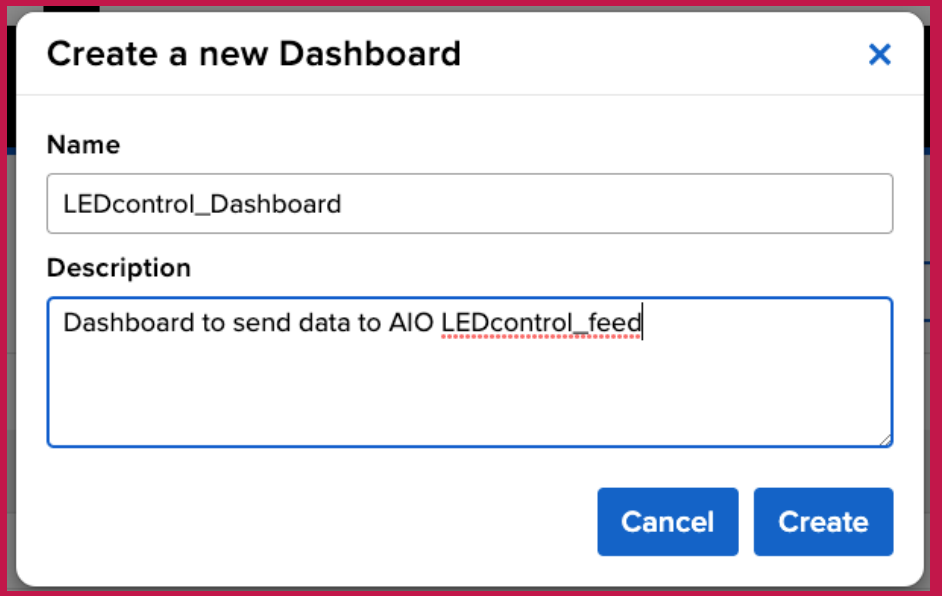
# Create AIO Feed



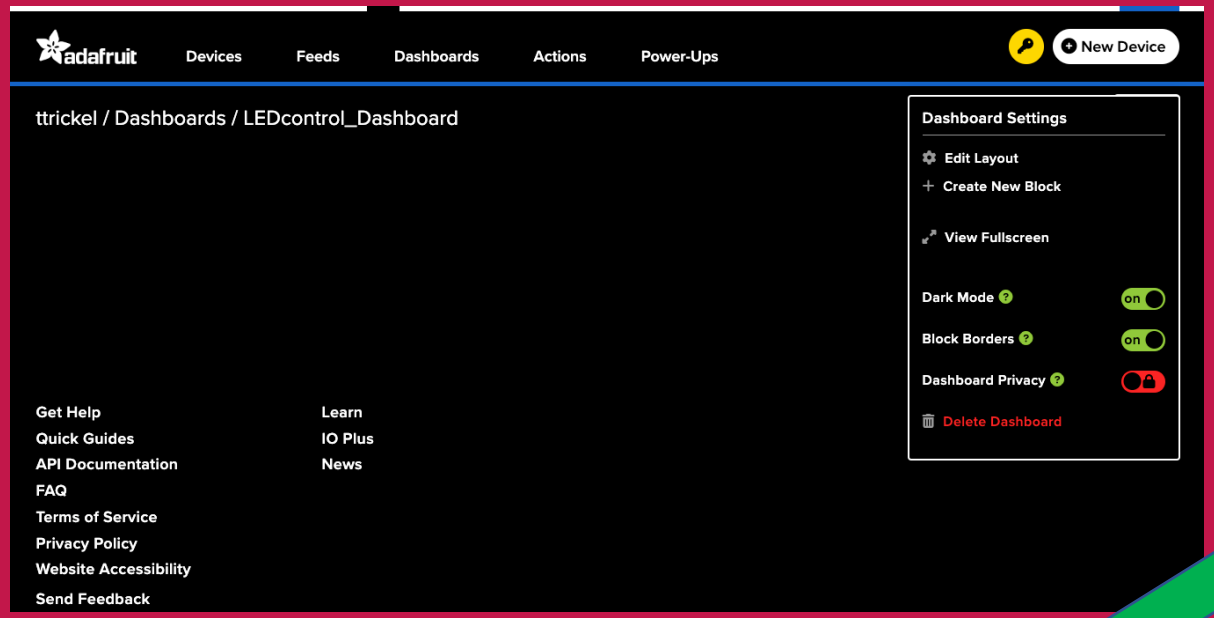



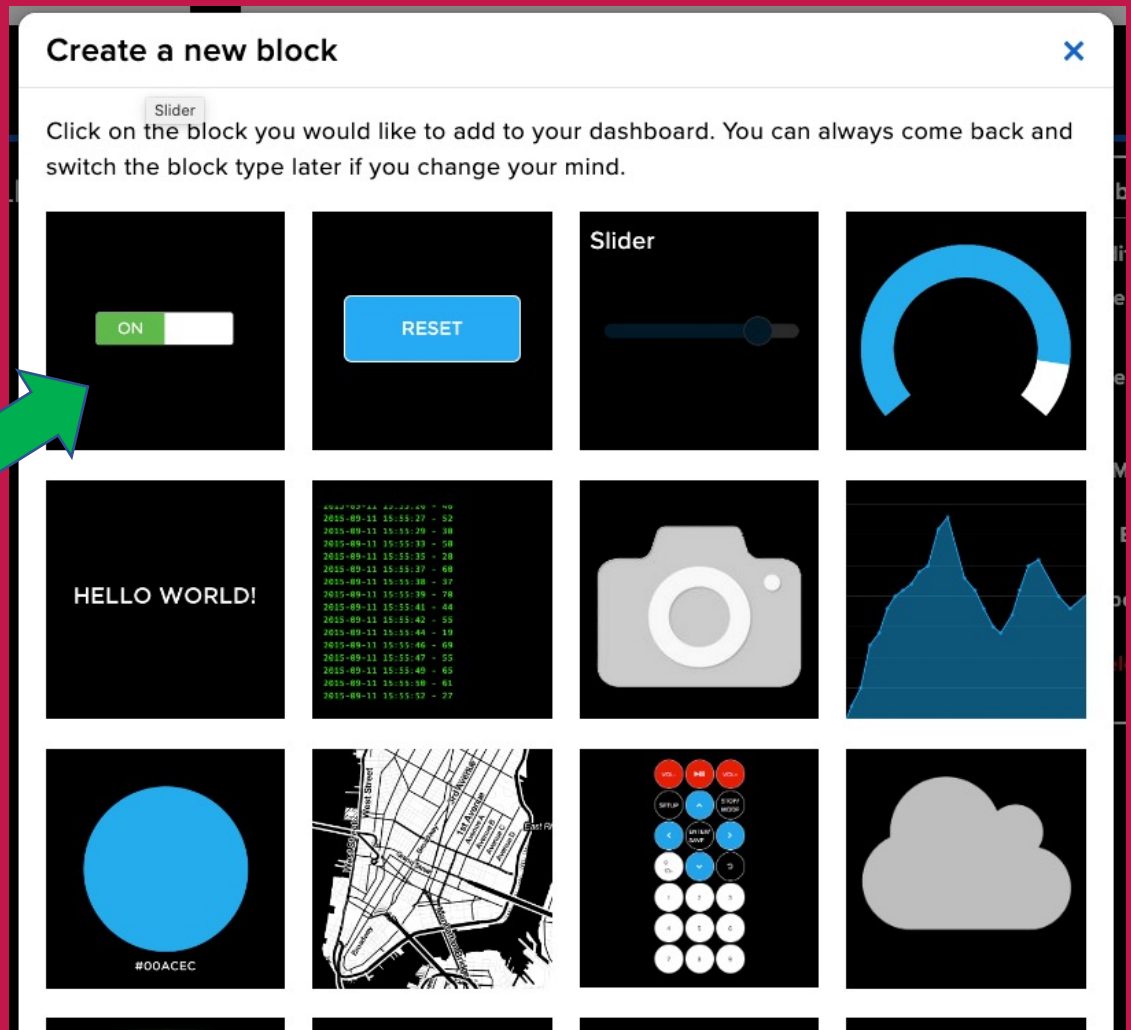
# Create AIO Dashboard



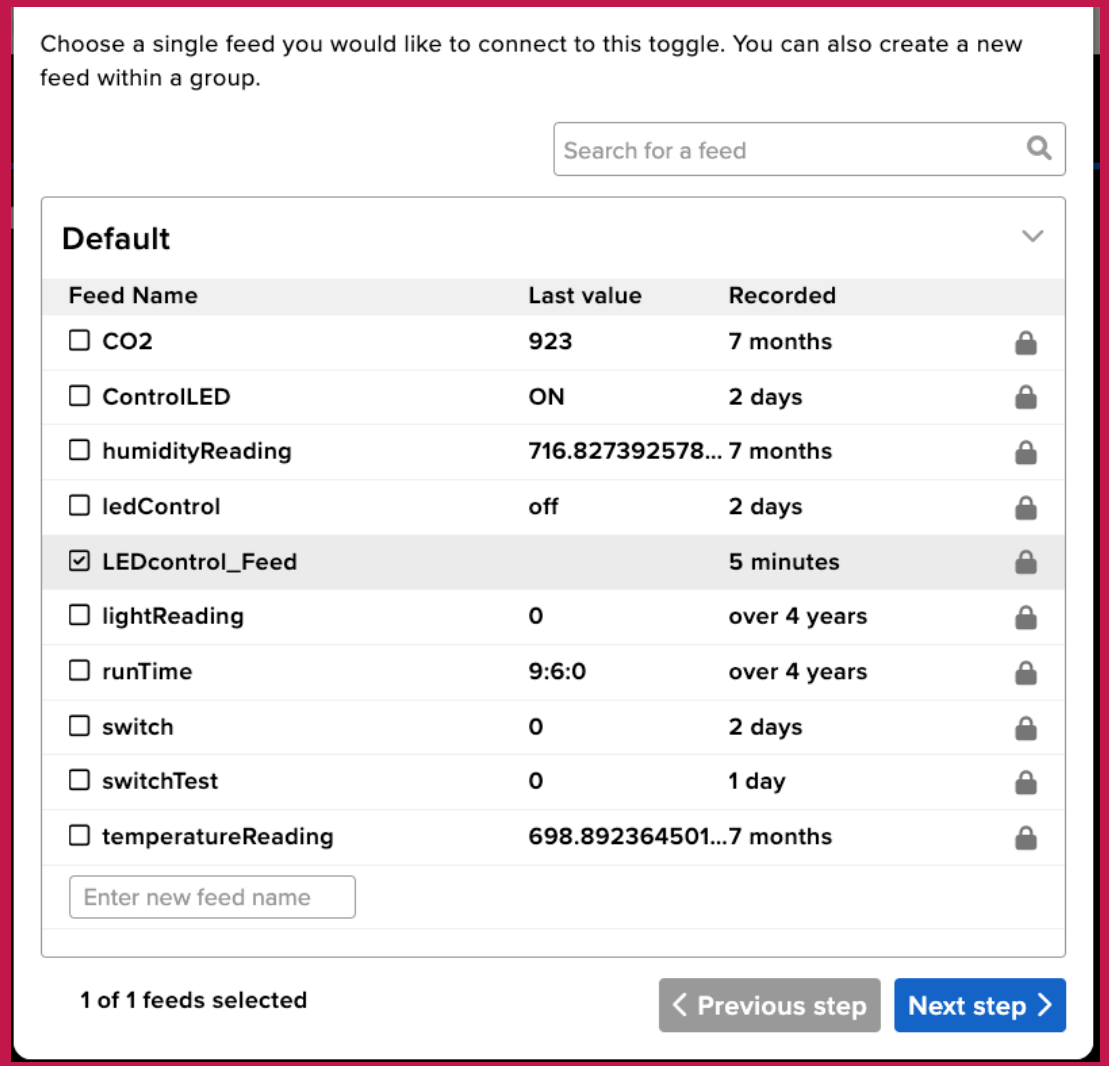



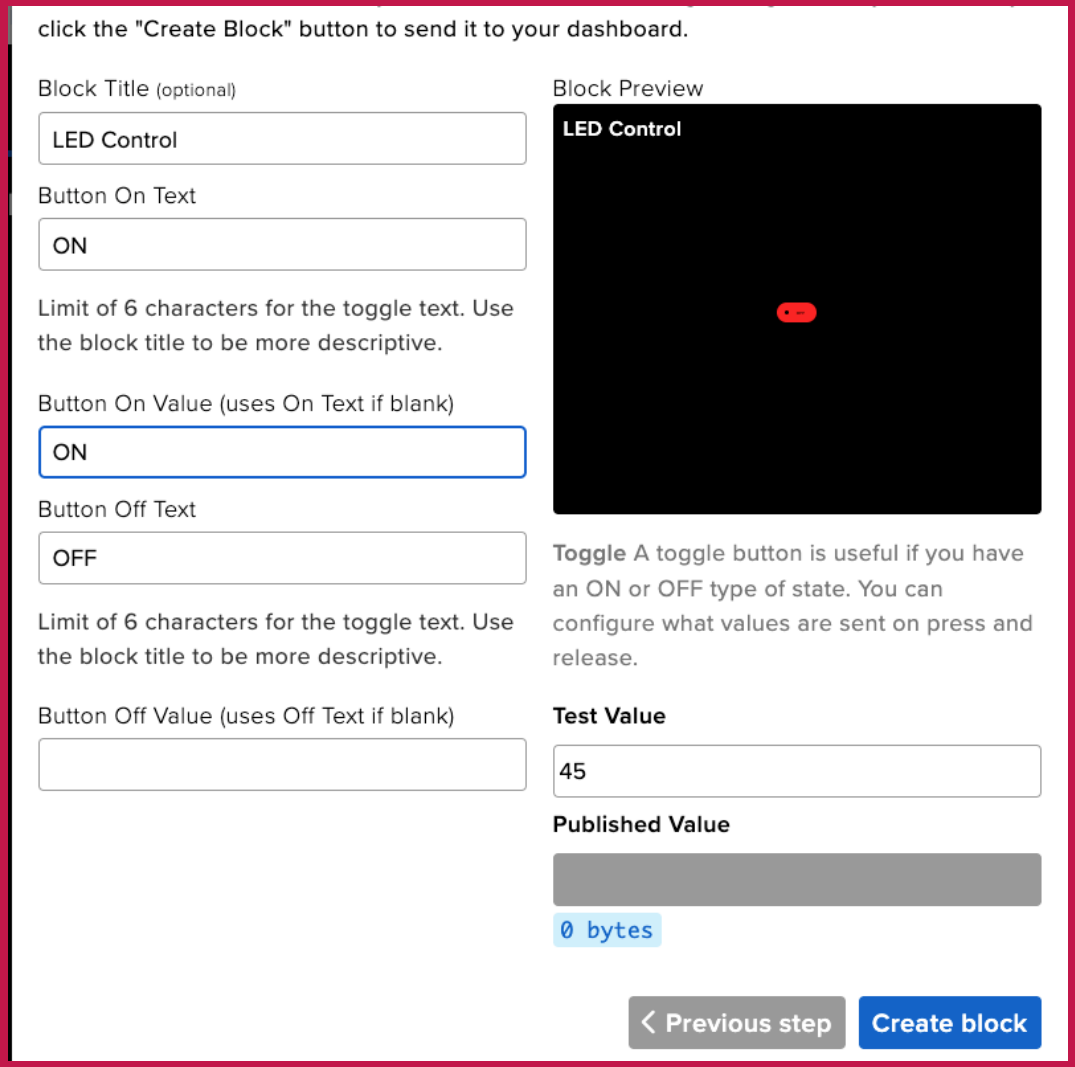
# Create AIO Dashboard continued



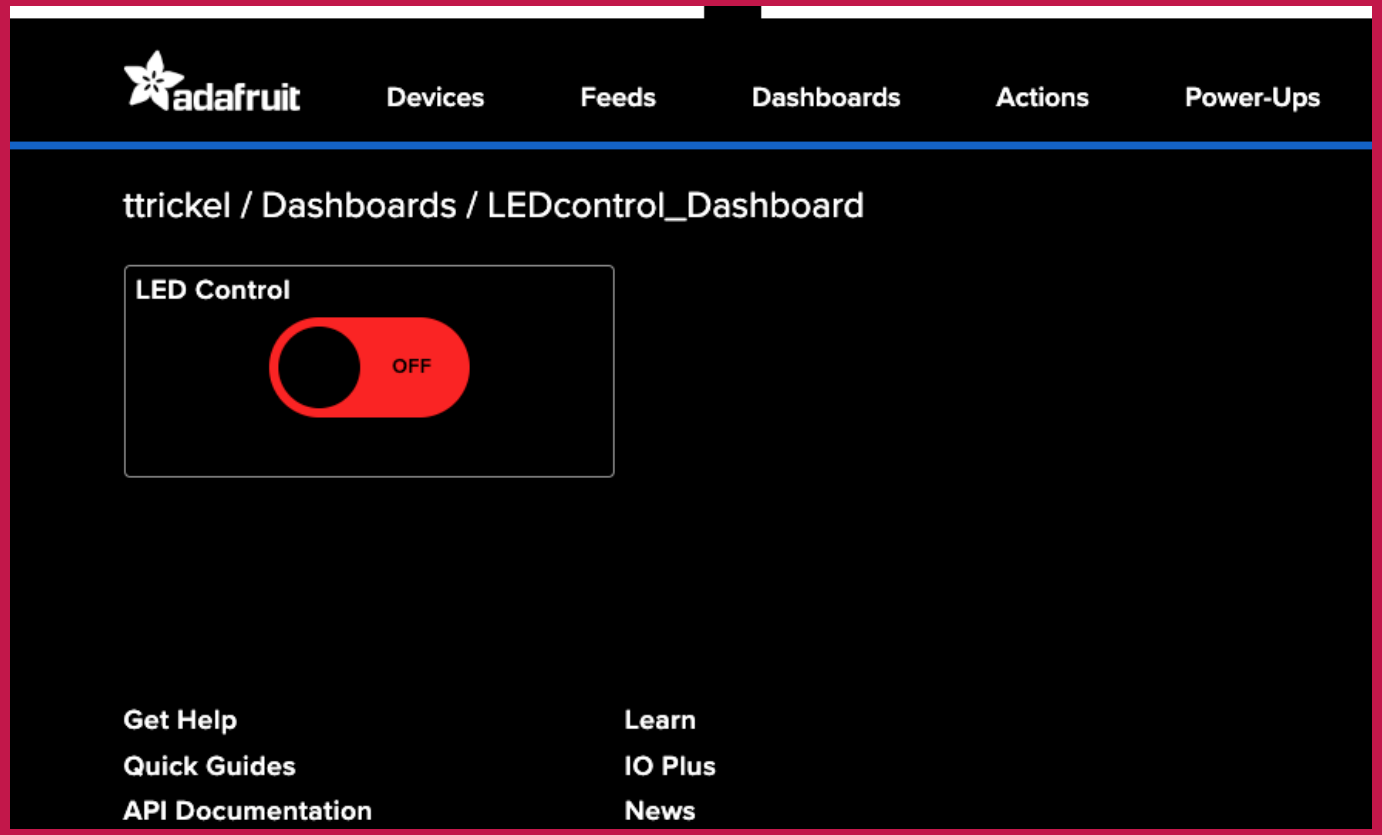



# Create AIO Dashboard continued



# Create AIO Dashboard continued



# Control LED from AIO Python Code

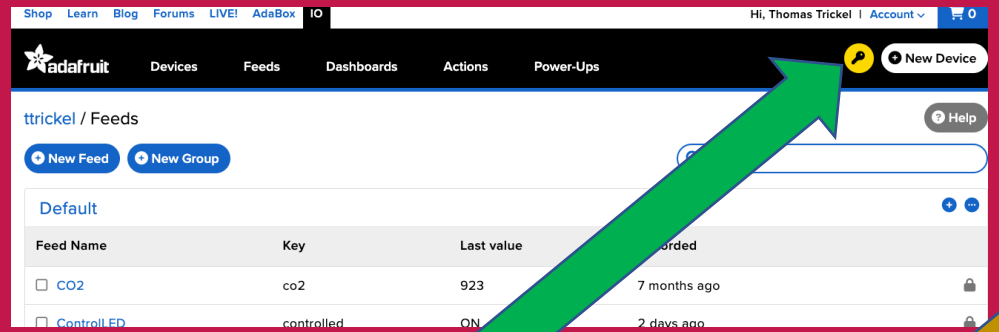
change

YOUR_AIO_KEY

YOUR_AIO_USERNAME

```
"""
'digital_out.py'
==================================
Example of turning on and off a LED
from the Adafruit IO Python Client
Author(s): Brent Rubell, Todd Treece
           with modifications by Thomas Trickel
"""
# Import standard python modules
import time
import RPi.GPIO as GPIO

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed, RequestError

# setup GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)


# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME"

# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

try: # if we have a 'digital' feed
    digital = aio.feeds('ledcontrol-feed')
except RequestError: # create a digital feed
    feed = Feed(name="ledcontrol-feed")
    digital = aio.create_feed(feed)

while True:
    data = aio.receive(digital.key)
    if data.value == "ON":
        print('received <- ON\n')
        GPIO.output(18, True)
    elif data.value == "OFF":
        print('received <- OFF\n')
        GPIO.output(18, False)

# timeout so we dont flood adafruit-io with requests
    time.sleep(0.5)

```

# AIO KEY & USERNAME




## YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

**Username** [redacted]

**Active Key** [redacted] REGENERATE KEY

Hide Code Samples

**Arduino**

```
#define IO_USERNAME  "ttrickel"
#define IO_KEY       "a0da523dfa16502494a70e61e1f683ba8
a0b32f4"
```

**Linux Shell**

```
export IO_USERNAME="ttrickel"
export IO_KEY="a0da523dfa16502494a70e61e1f683ba8a0b32f
4"
```

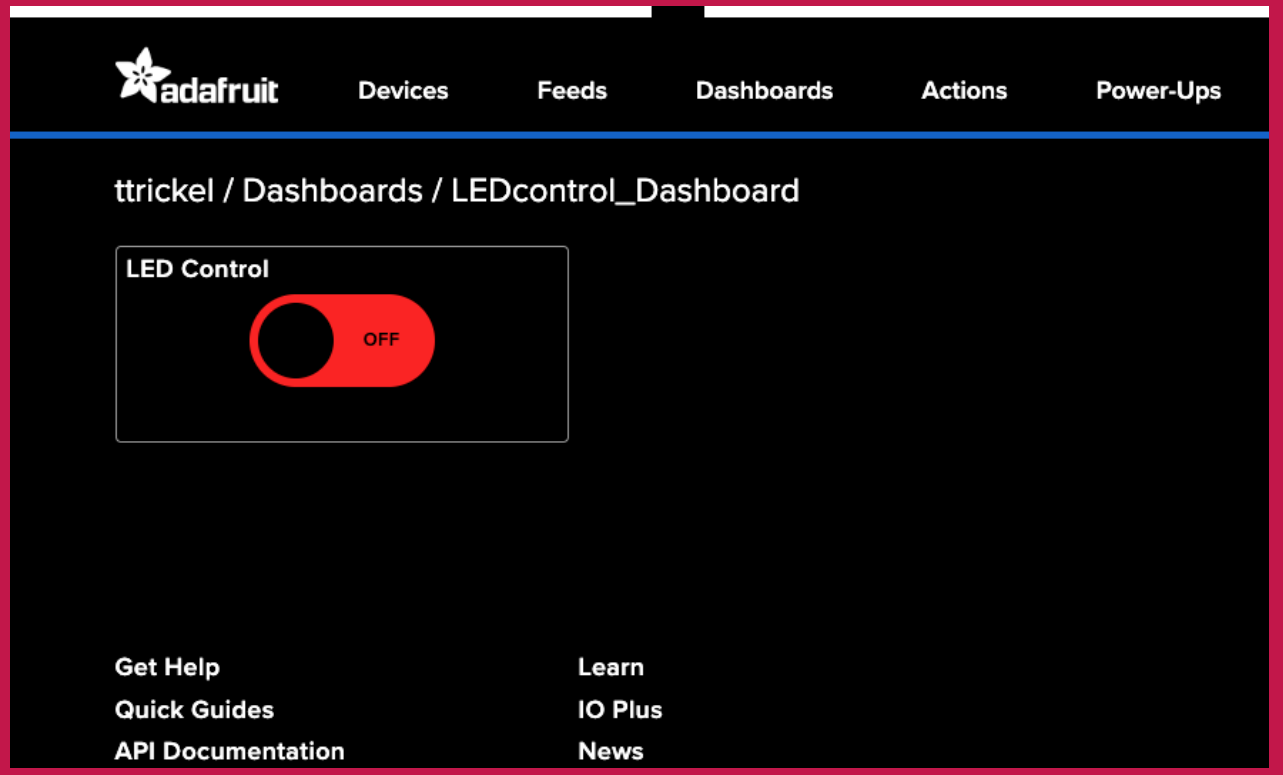
**Scripting**

```
ADAFRUIT_IO_USERNAME = "ttrickel"
ADAFRUIT_IO_KEY = "a0da523dfa16502494a70e61e1f683ba8a0b
32f4"
```

# Test It

## Run Python Code

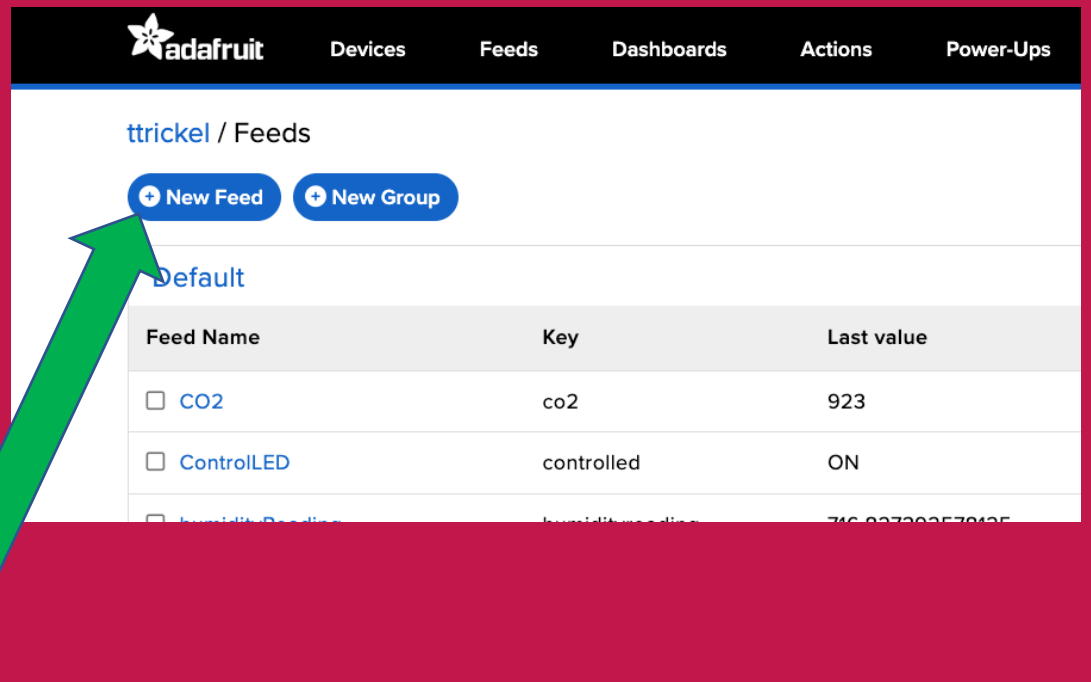
## Click on LED Control icon

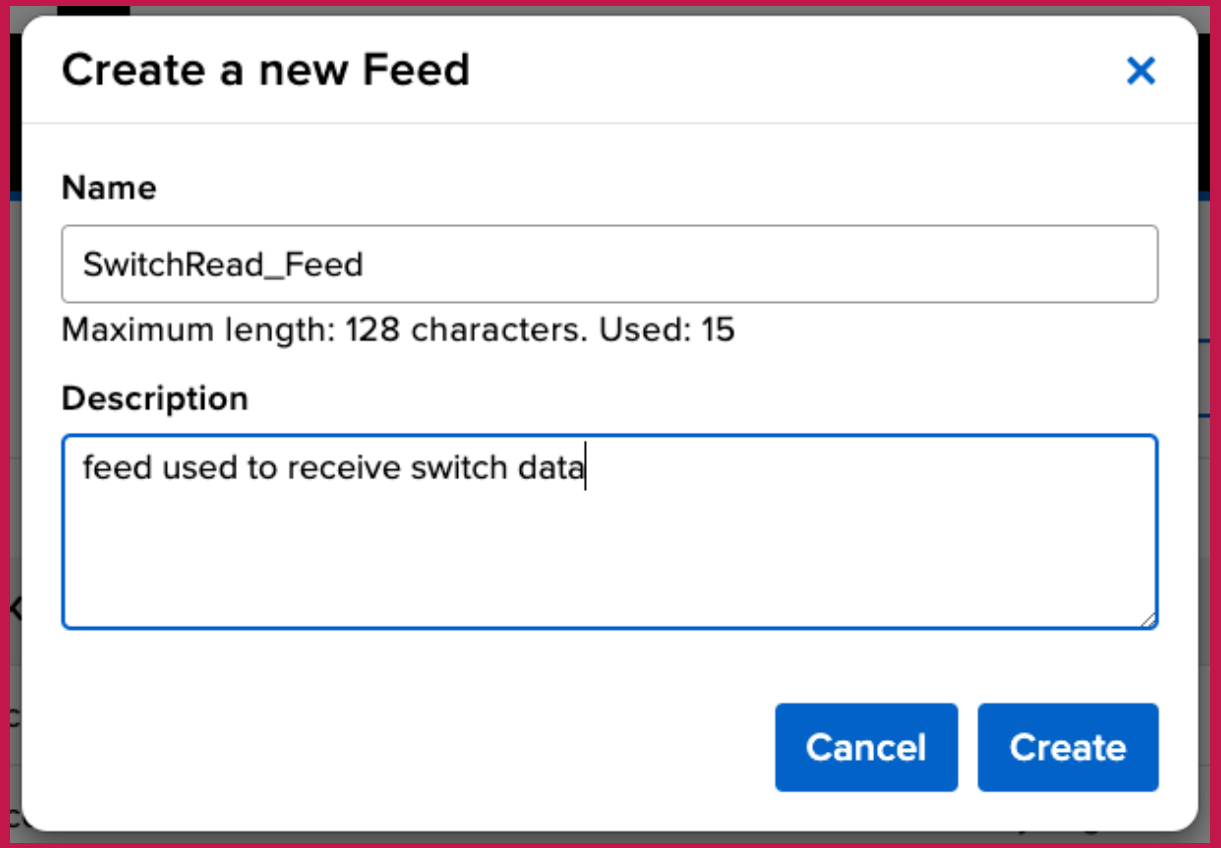


# Send Switch Data to AIO

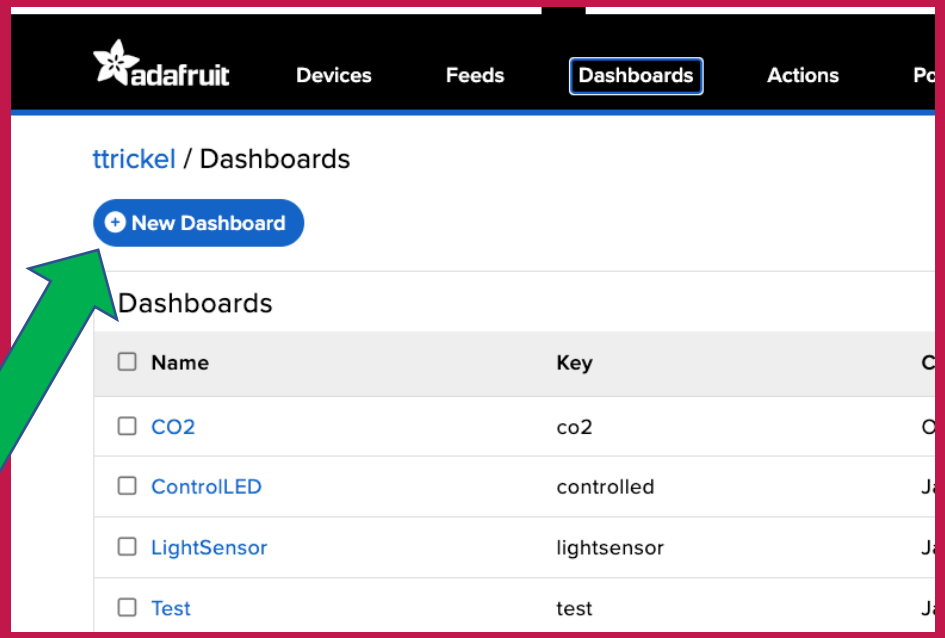
- Hardware is the same
- Create Feed
- Create Dashboard
- Write Python Code

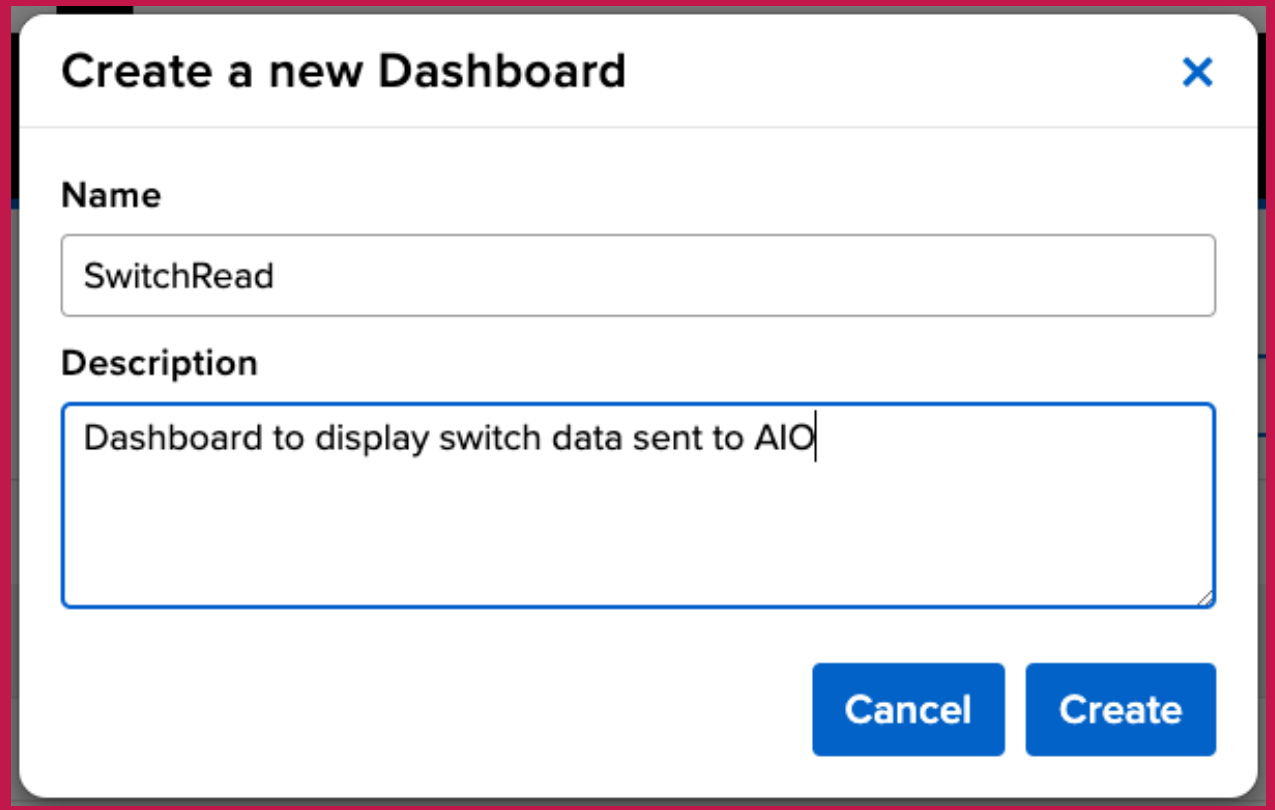
# Create AIO Feed



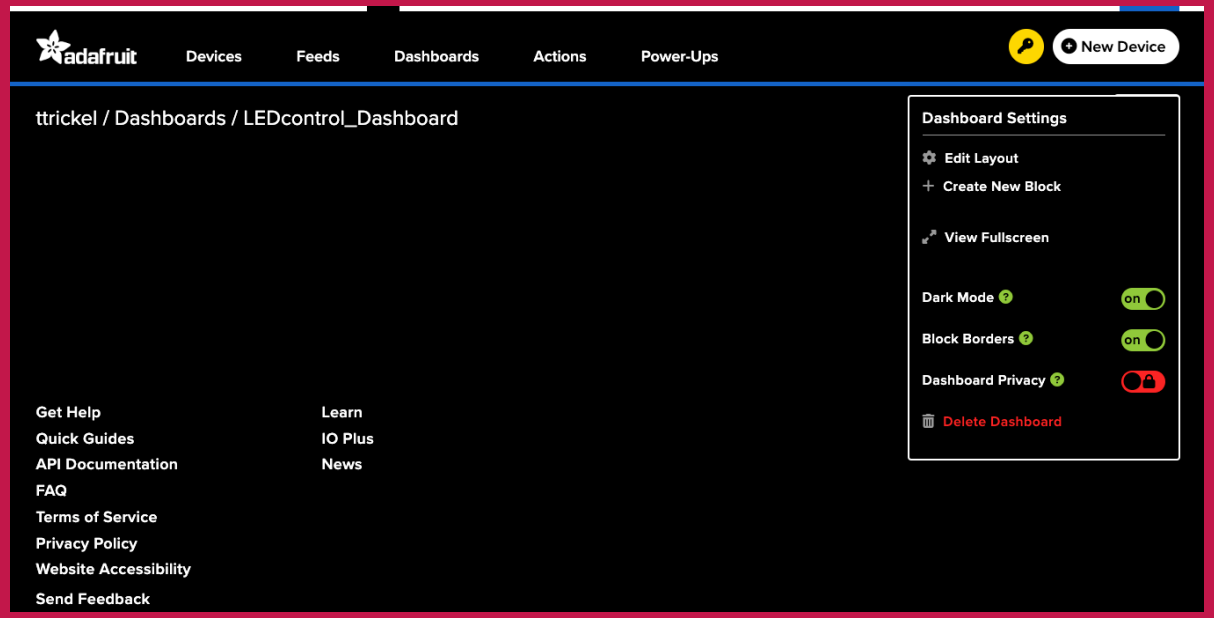



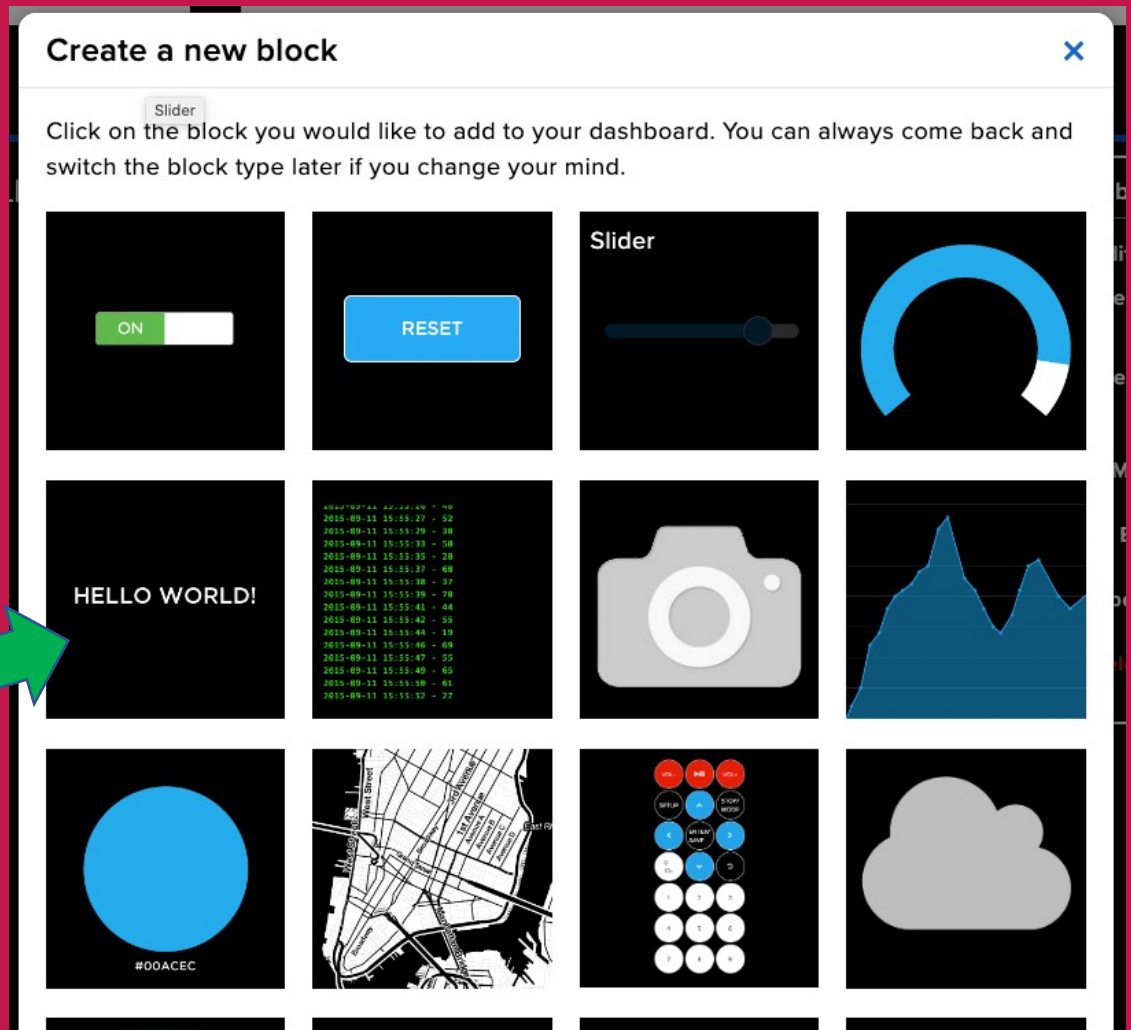
# Create AIO Dashboard



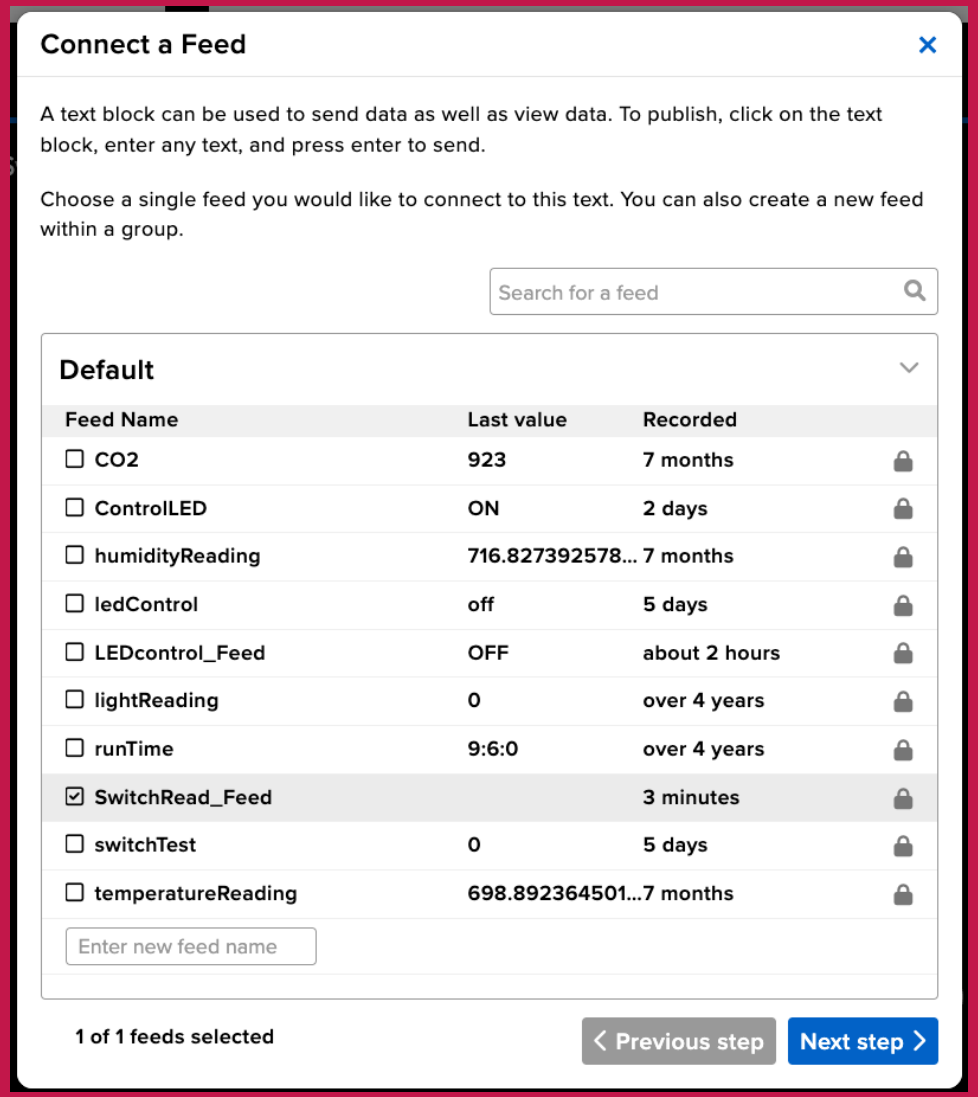



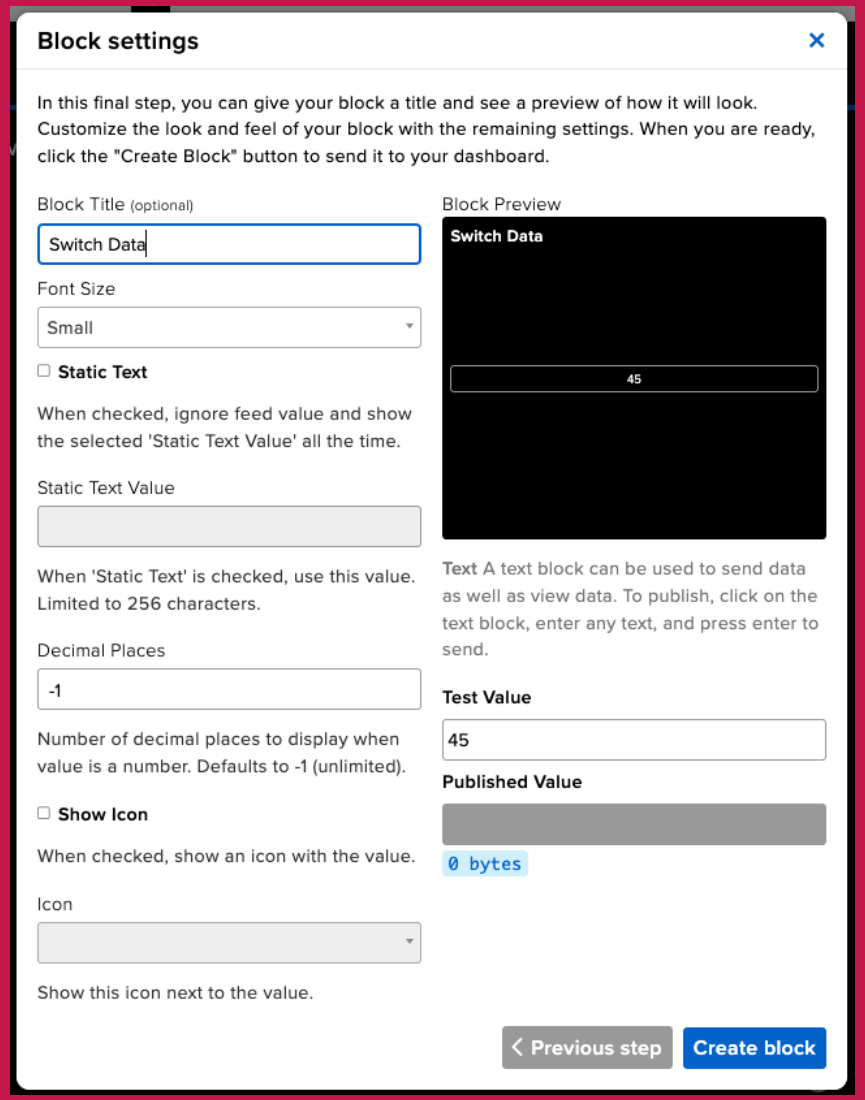
# Create AIO Dashboard continued



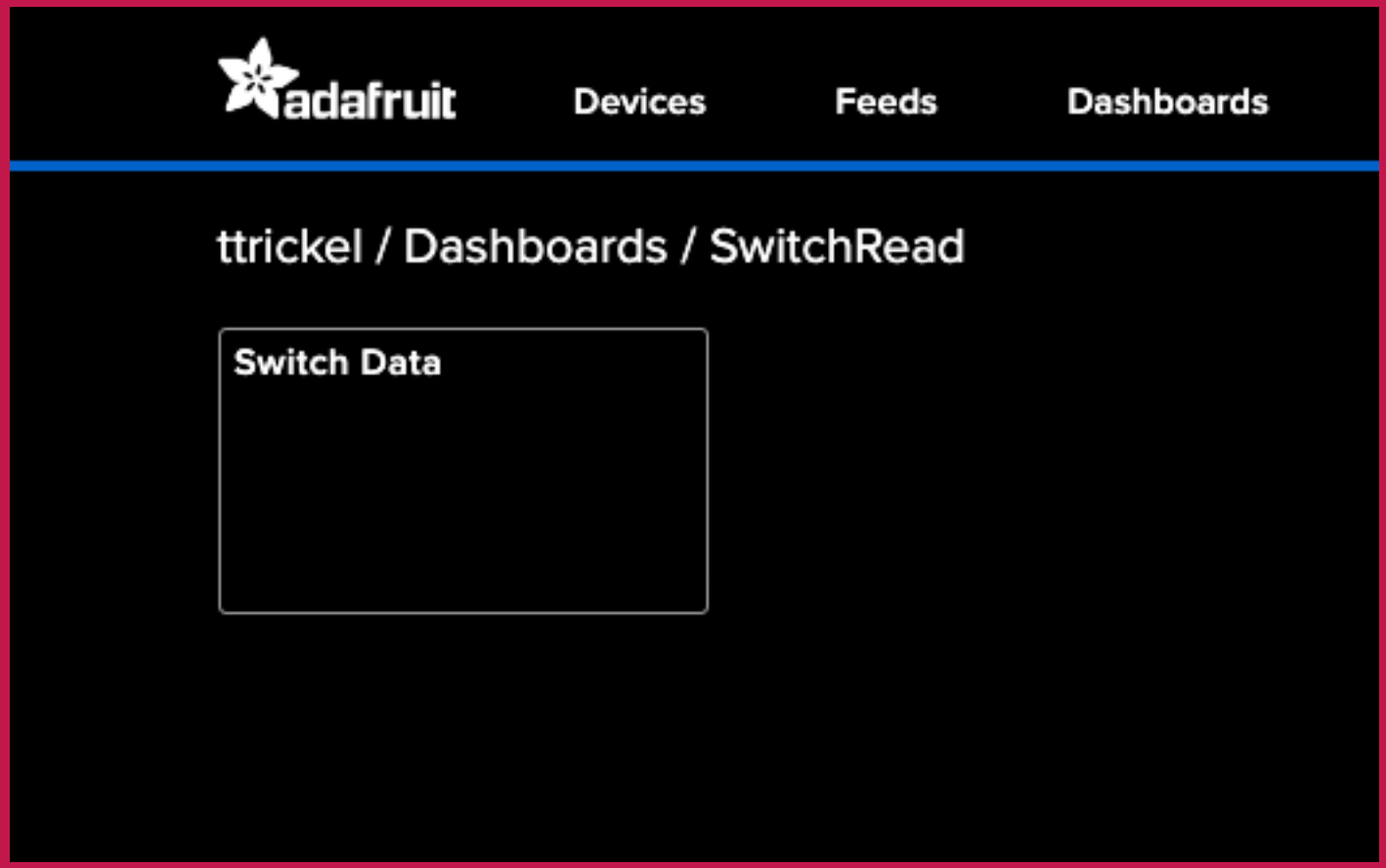



# Create AIO Dashboard continued



# Create AIO Dashboard continued



# Send Switch Data to AIO Python Code

change

YOUR_AIO_KEY

YOUR_AIO_USERNAME

```
"""
'digitalIn.py'
==================================
Example of sending the status of a
switch to the Adafruit IO Python Client
"""
# Import standard python modules
import time
import RPi.GPIO as GPIO

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed, RequestError

# setup GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.IN)

# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'

# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

try: # if we have a 'digital' feed
    digital = aio.feeds('switchread-feed')
except RequestError: # create a digital feed
    feed = Feed(name="switchread-feed")
    digital = aio.create_feed(feed)

while True:
    if GPIO.input(25):
        aio.send(digital.key, 0)
    else:
        aio.send(digital.key, 1)

# timeout so we dont flood adafruit-io with requests
    time.sleep(0.5)

```

# Test It

## Run Python Code

## Press Switch on Breadboard