

# 2024 Raspberry Pi Workshop

December 27, 2024

Thomas Trickel



# Agenda

Welcome

Raspberry Pi

Terminal

Python

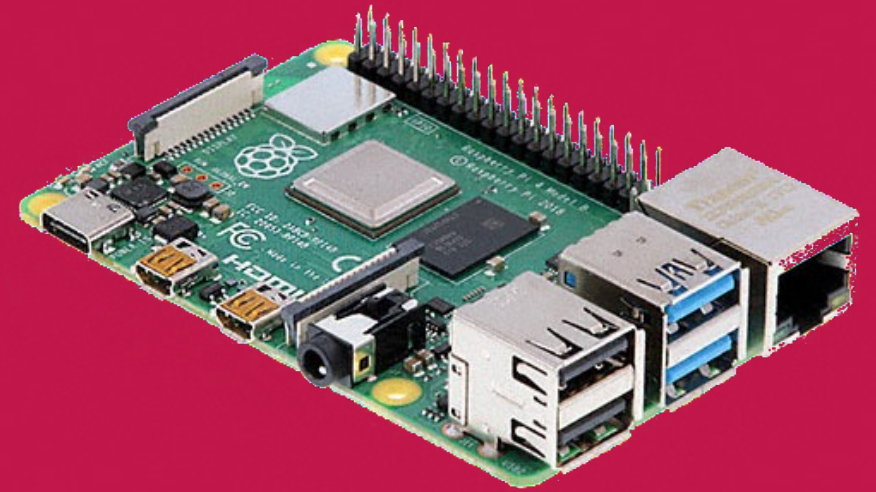
Blink LED

Break

Switch Controlled LED

Write to File on Switch Press

Cronjob Alarm





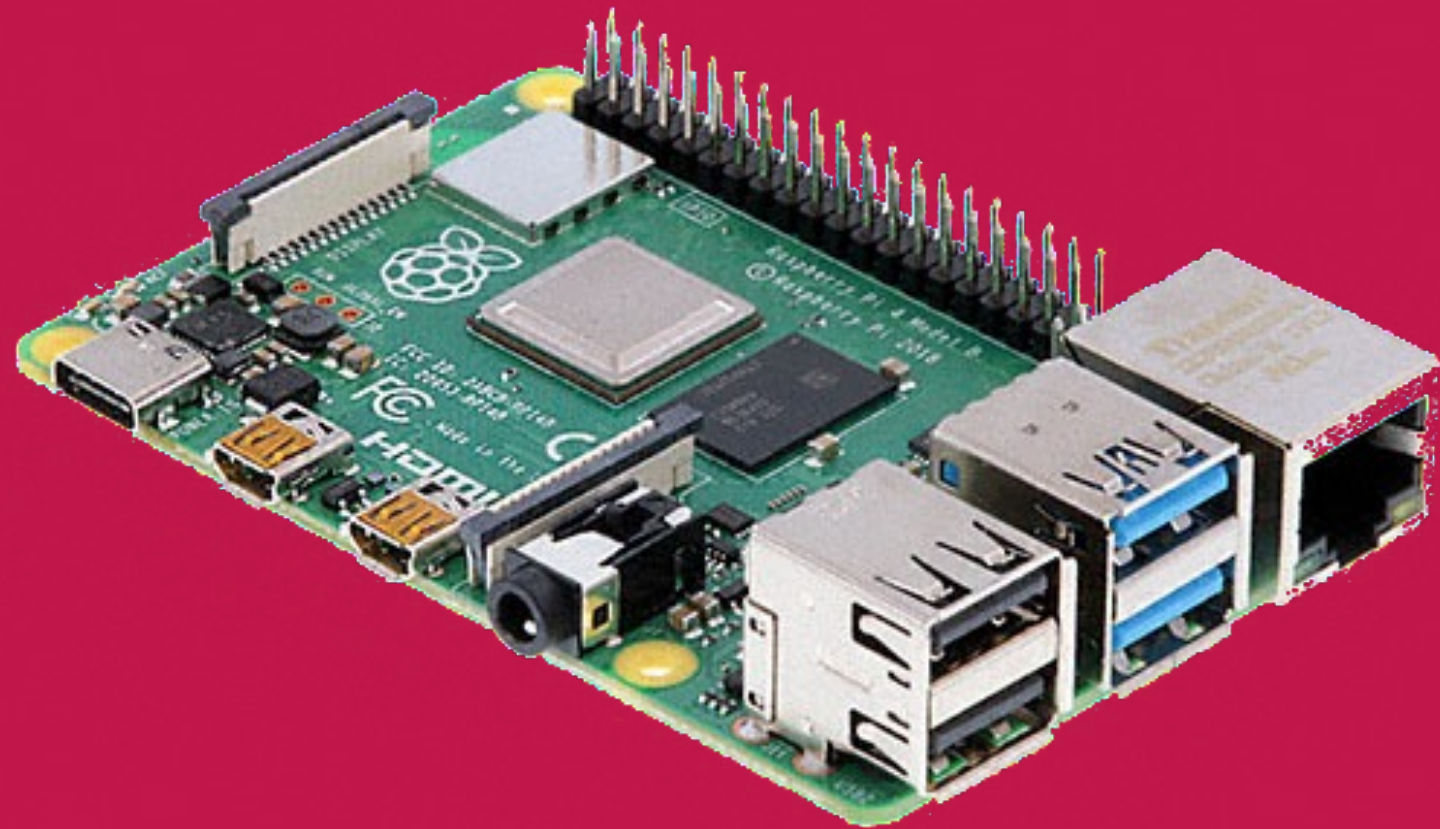
# Welcome

Name

"Coolest Tech" Used

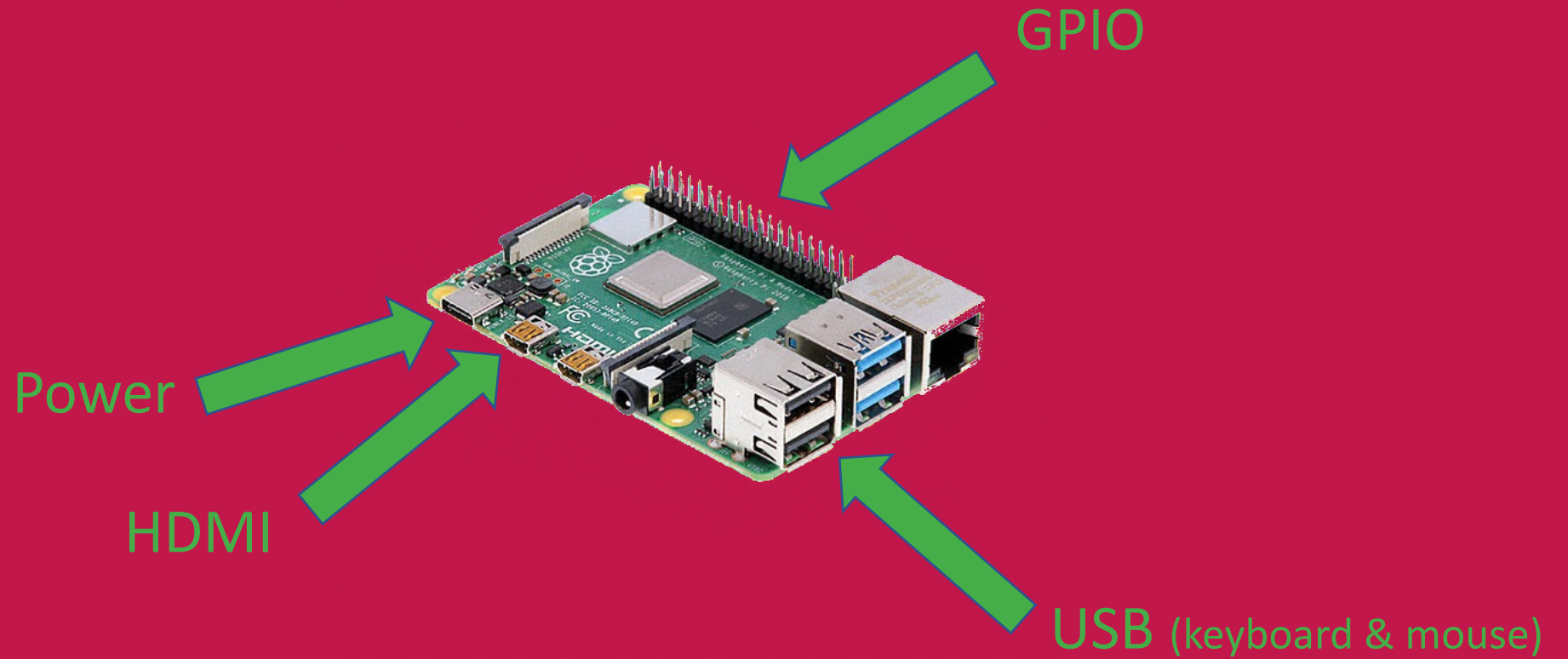


# Raspberry Pi





# Raspberry Pi

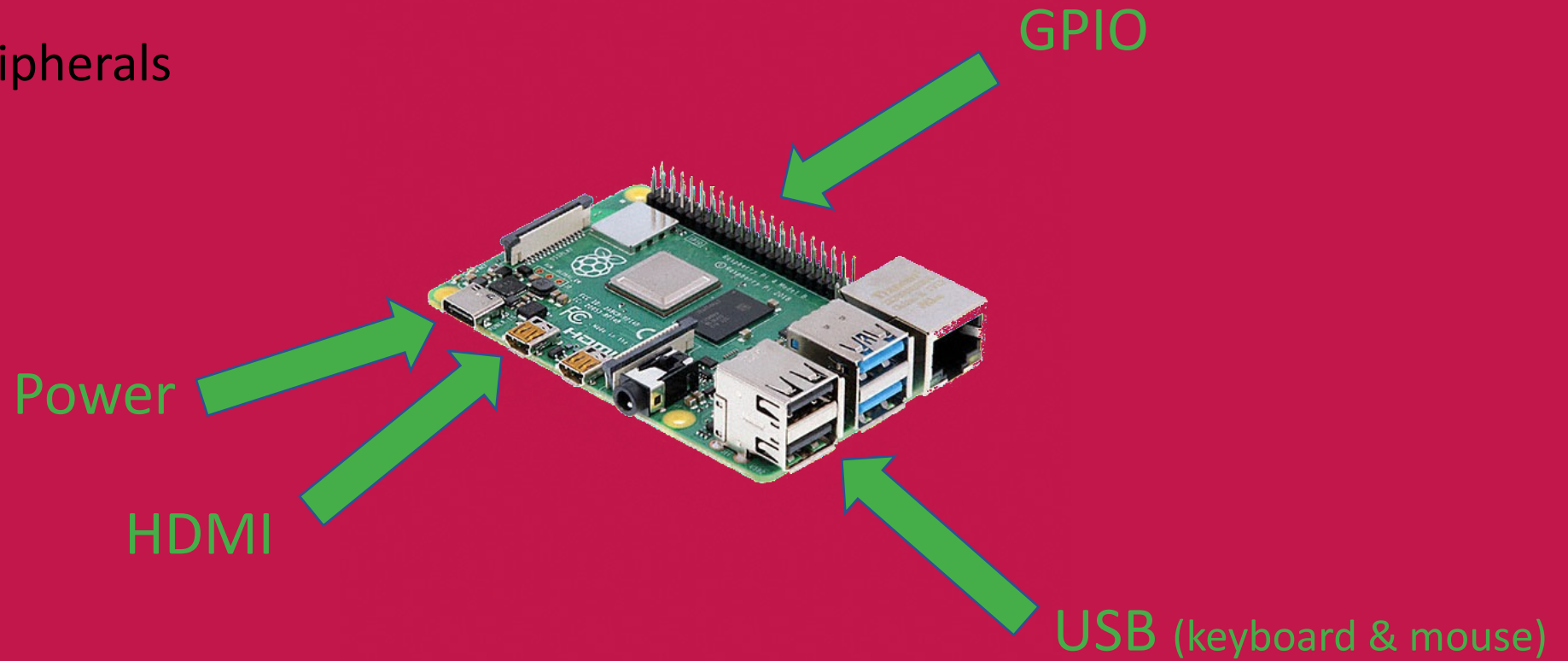




# Raspberry Pi

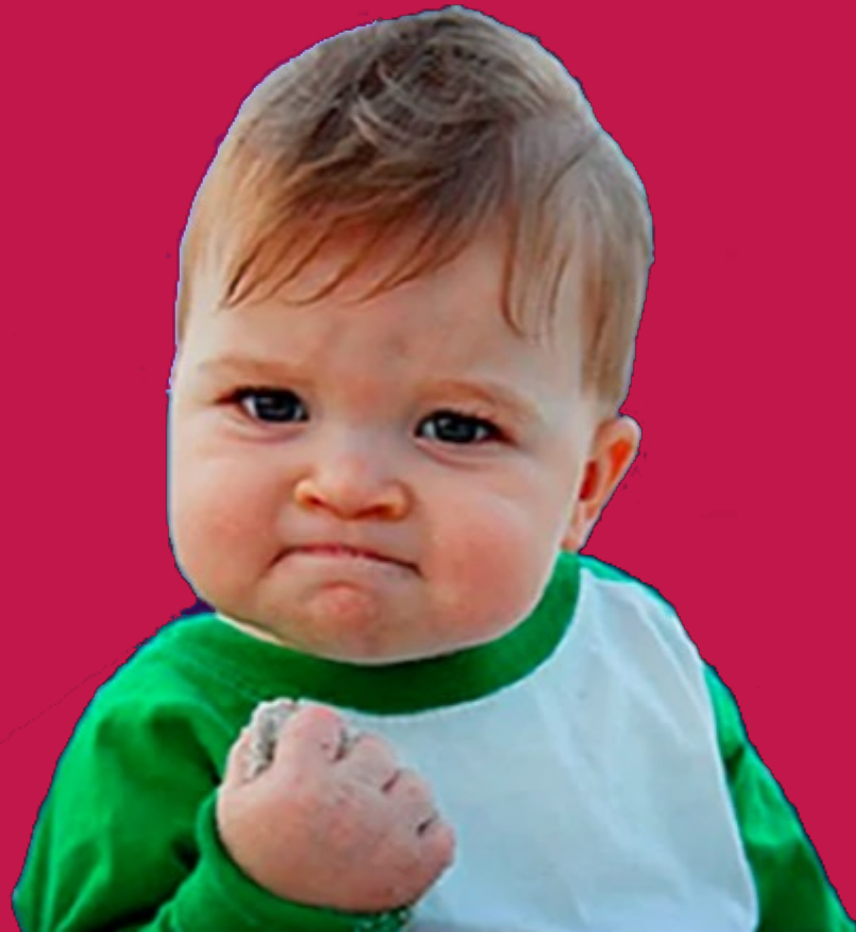
Connect Peripherals

Power On





Excellent!



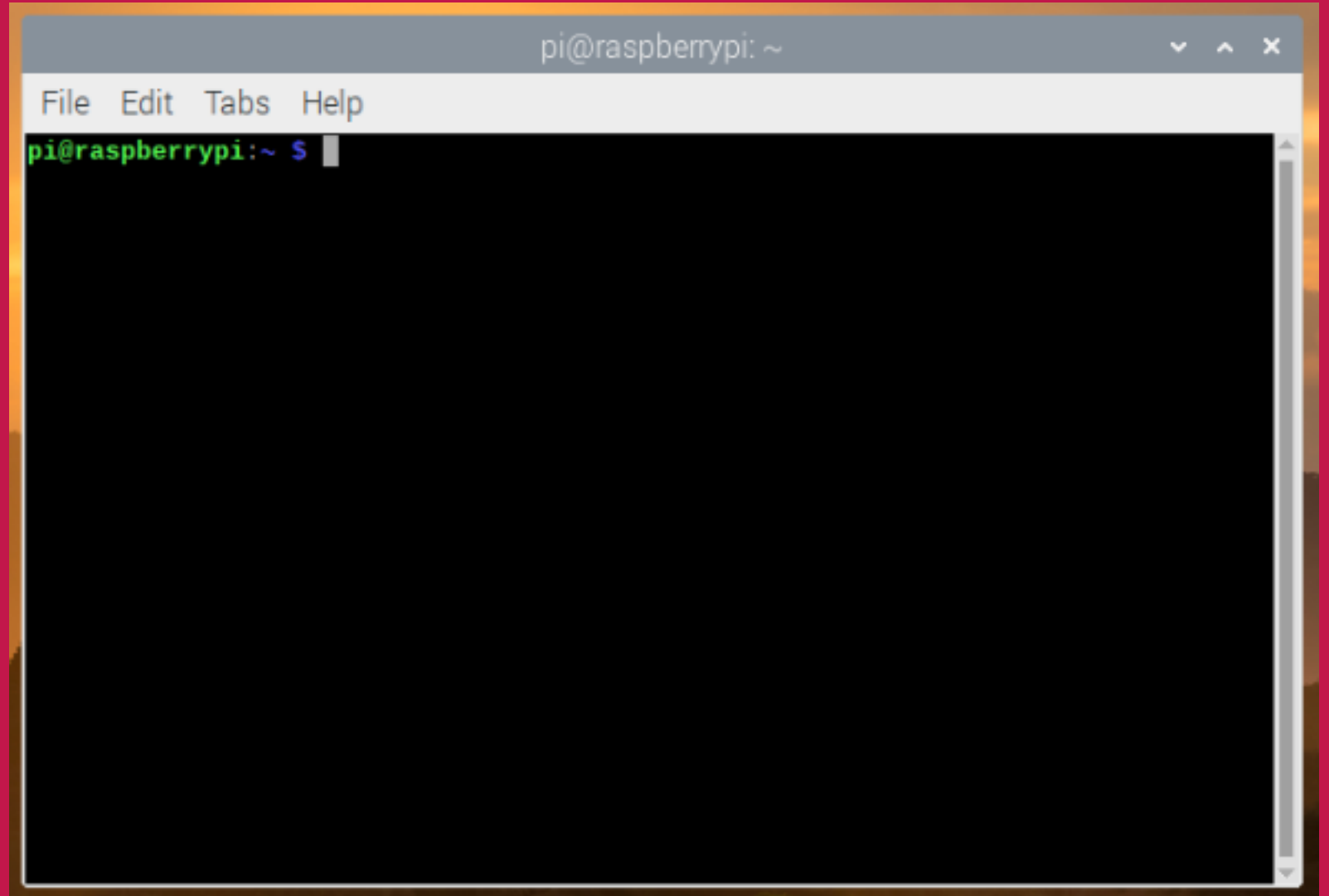


# Terminal

"Old School"

Text based

Command Line Interface



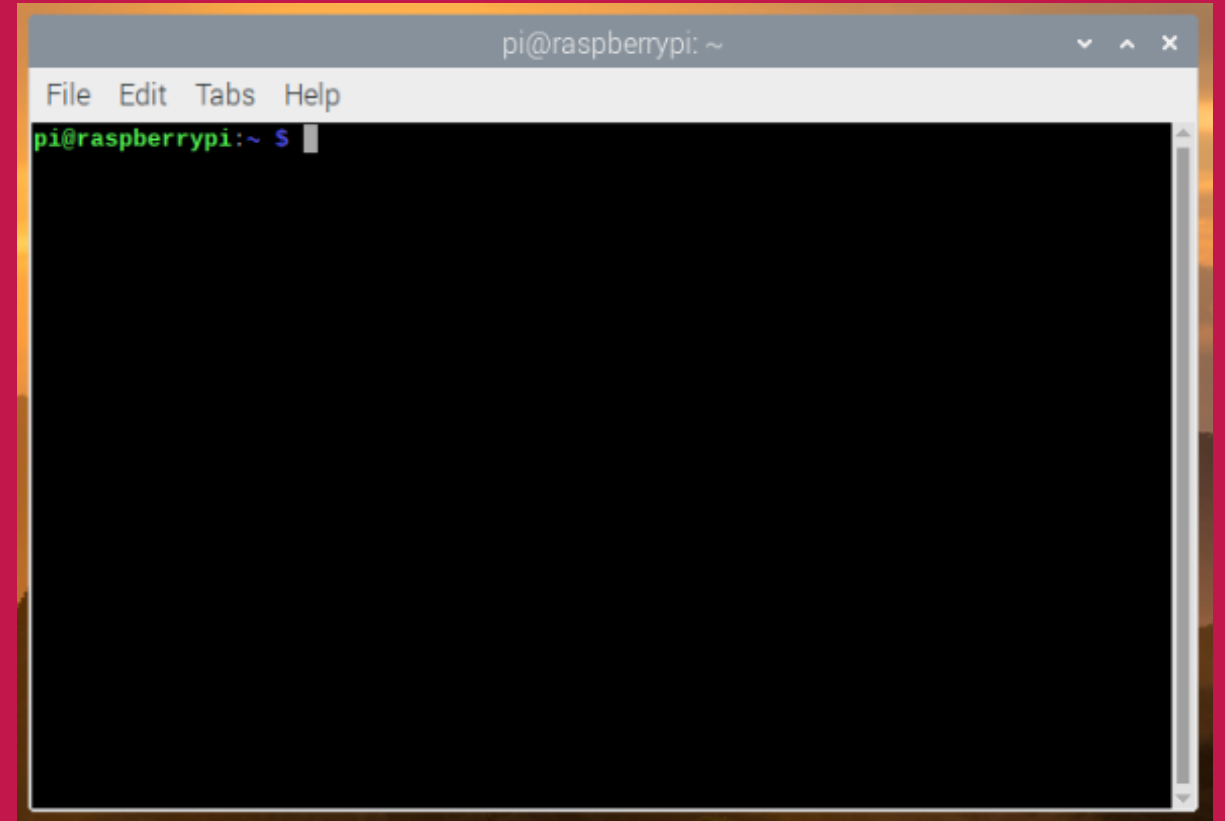


# Terminal

Often shown in tutorials

Faster system management

Direct execution of scripts (programs)





# Terminal

pwd

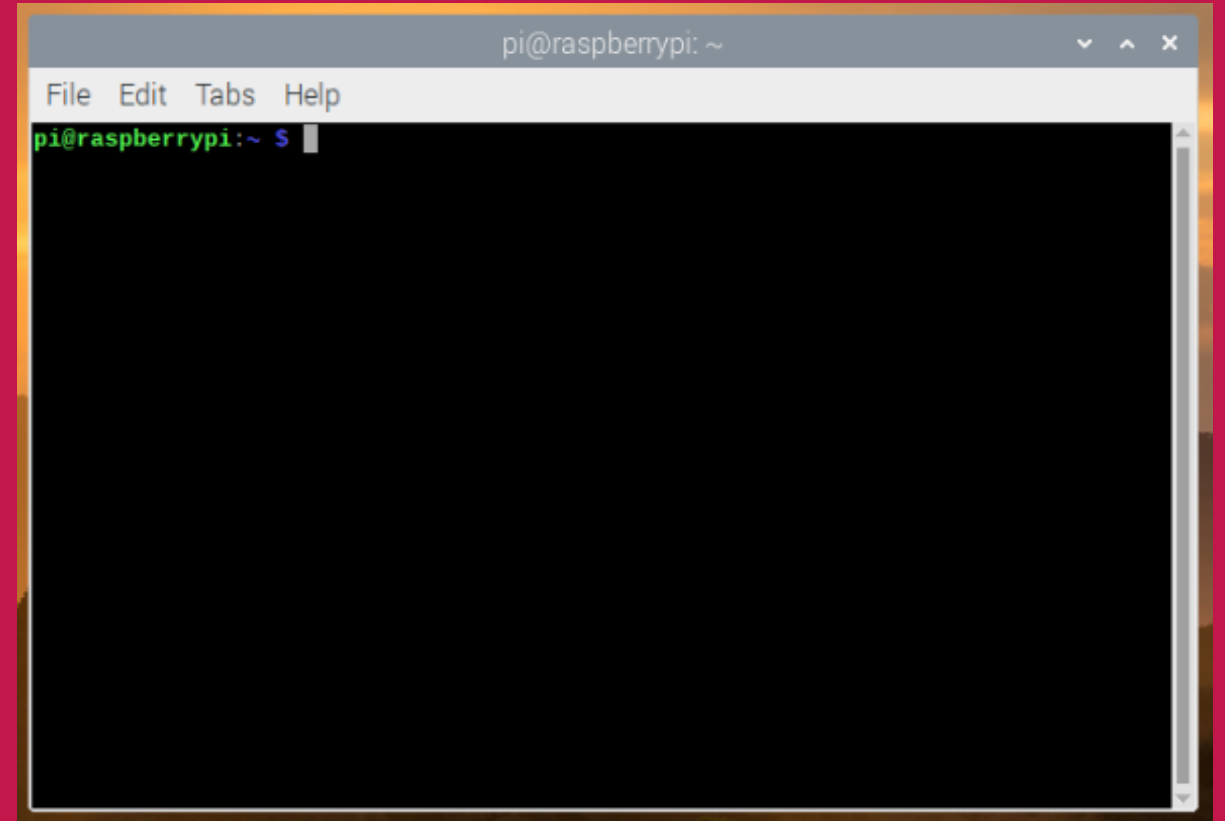
ls

cd

mkdir

nano

chmod



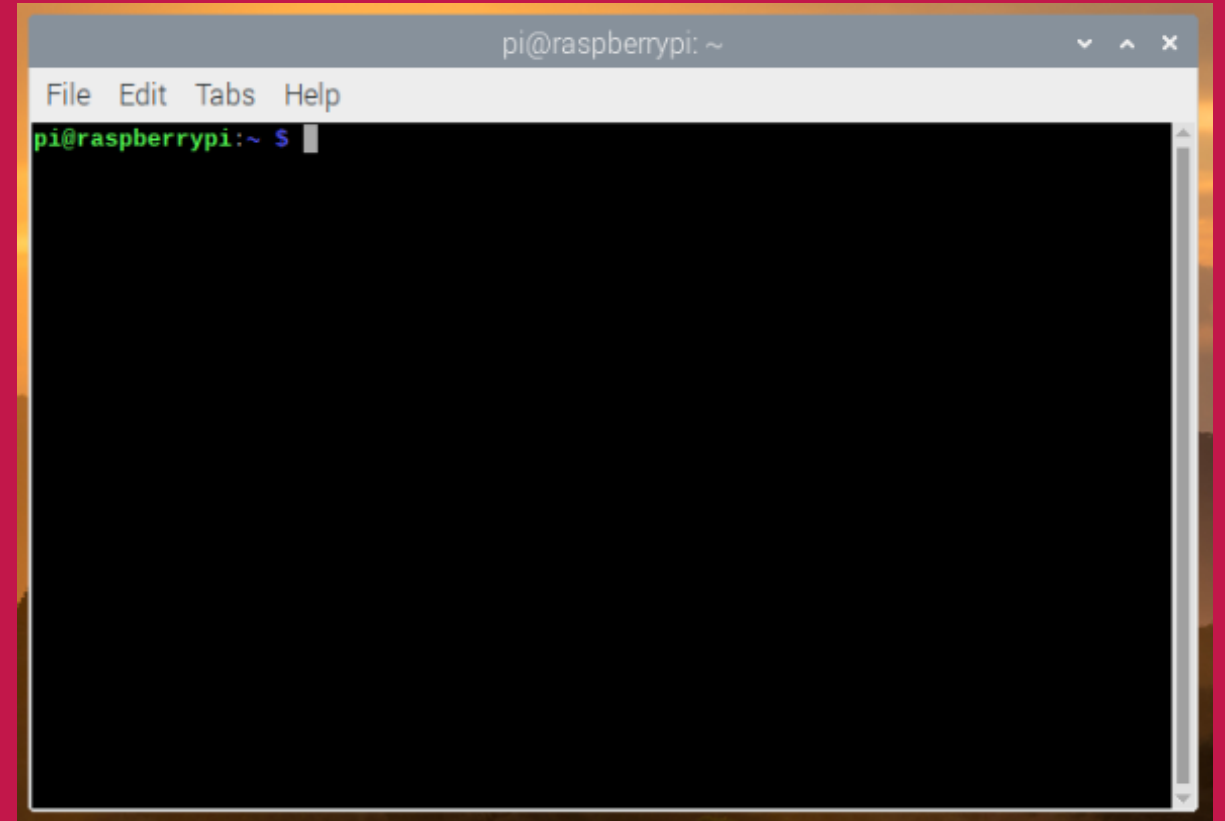


# Terminal

`mkdir <yourname>`

`cd <yourname>`

`nano helloworld.sh`





# Nano

echo "hello world"

```
GNU nano 2.0.6           File: helloworld.sh
echo "hello world"

[ Read 1 line ]
^G Get Help   ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



# Nano

echo "hello world"

<ctrl> o

<enter>

```
GNU nano 2.0.6           File: helloworld.sh

echo "hello world!"

File Name to Write: helloworld.sh
^G Get Help           ^T To Files           M-M Mac Format       M-P Prepend
^C Cancel             M-D DOS Format        M-A Append           M-B Backup File
```



# Nano

echo "hello world"

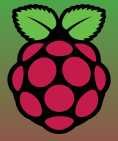
<ctrl> o

<enter>

<ctrl> x

```
GNU nano 2.0.6           File: helloworld.sh
echo "hello world"

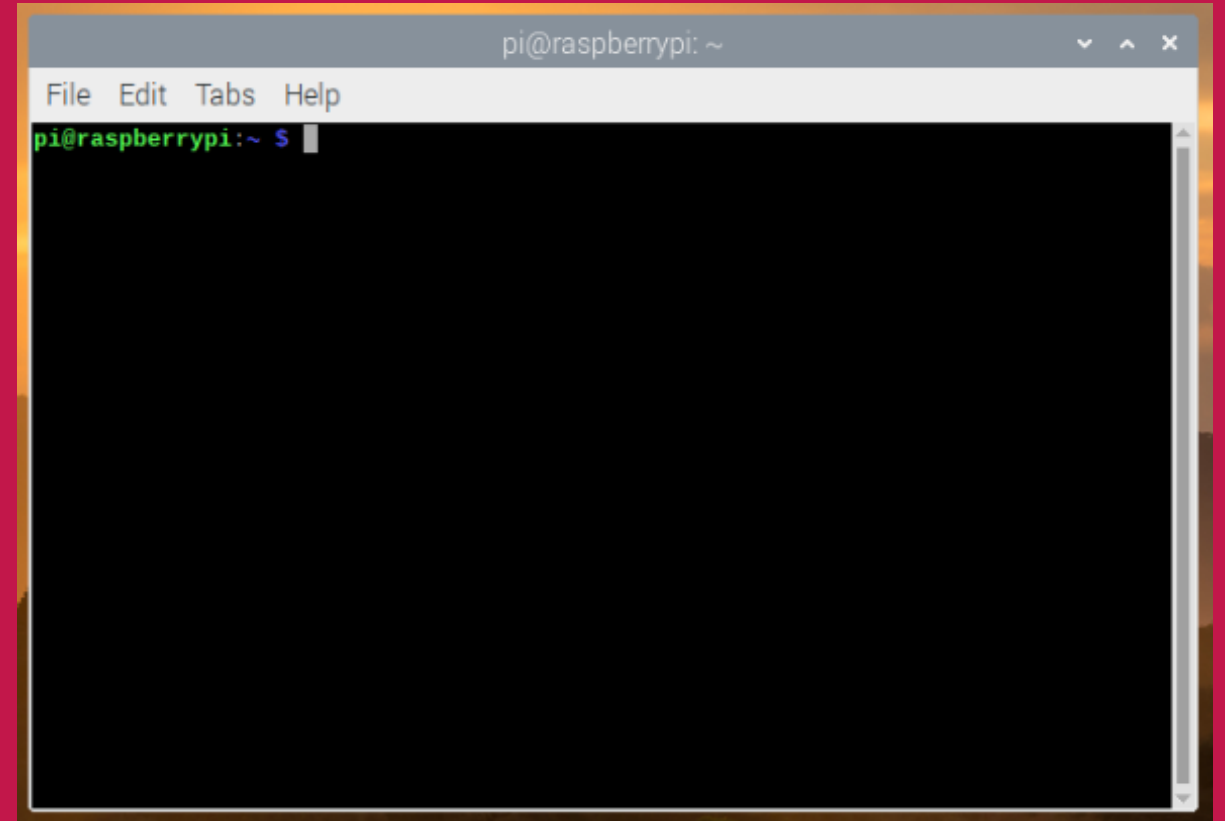
[ Read 1 line ]
^G Get Help   ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



# Terminal

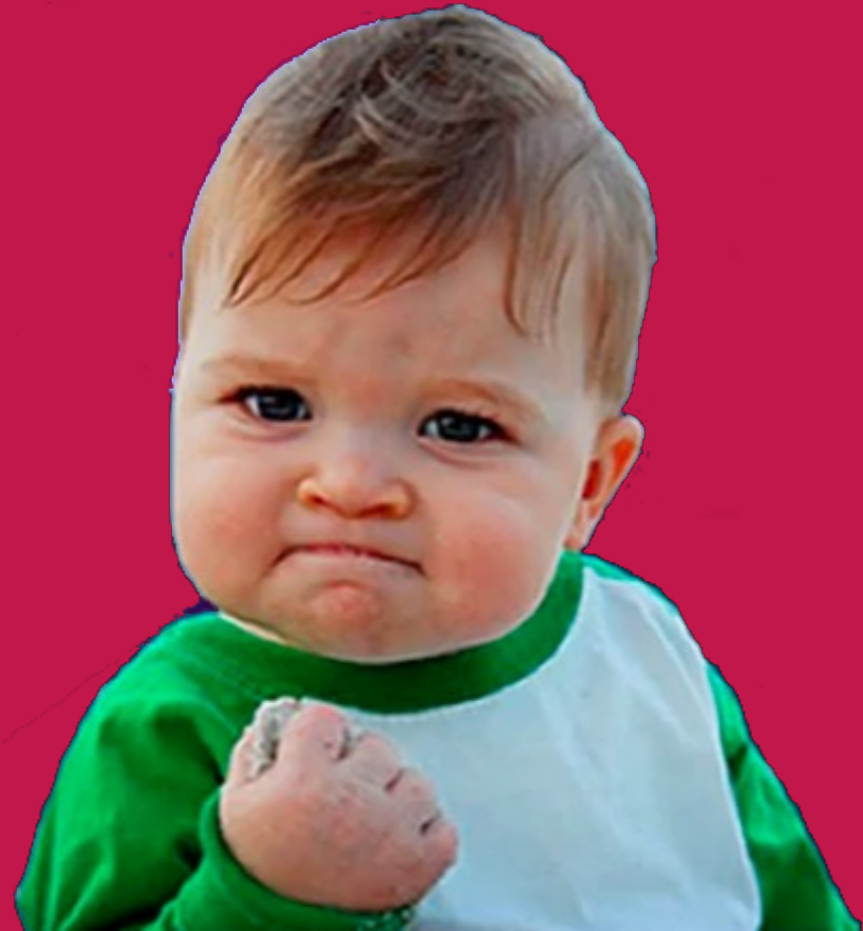
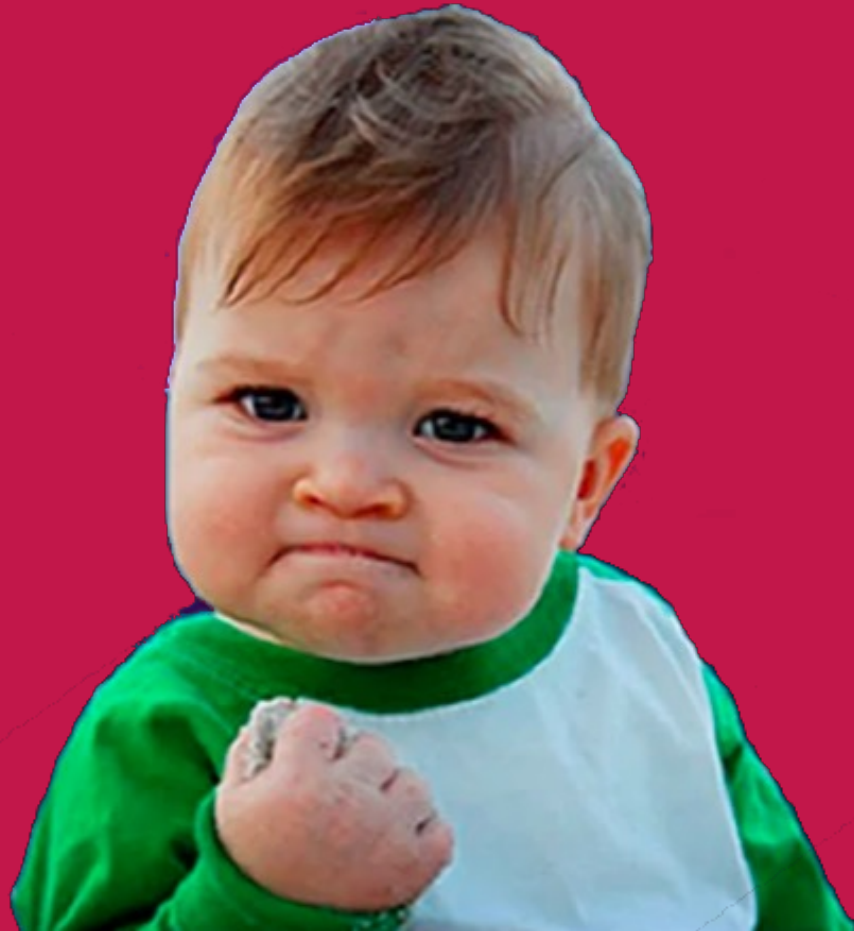
```
chmod 777 helloworld.sh
```

```
./helloworld.sh
```





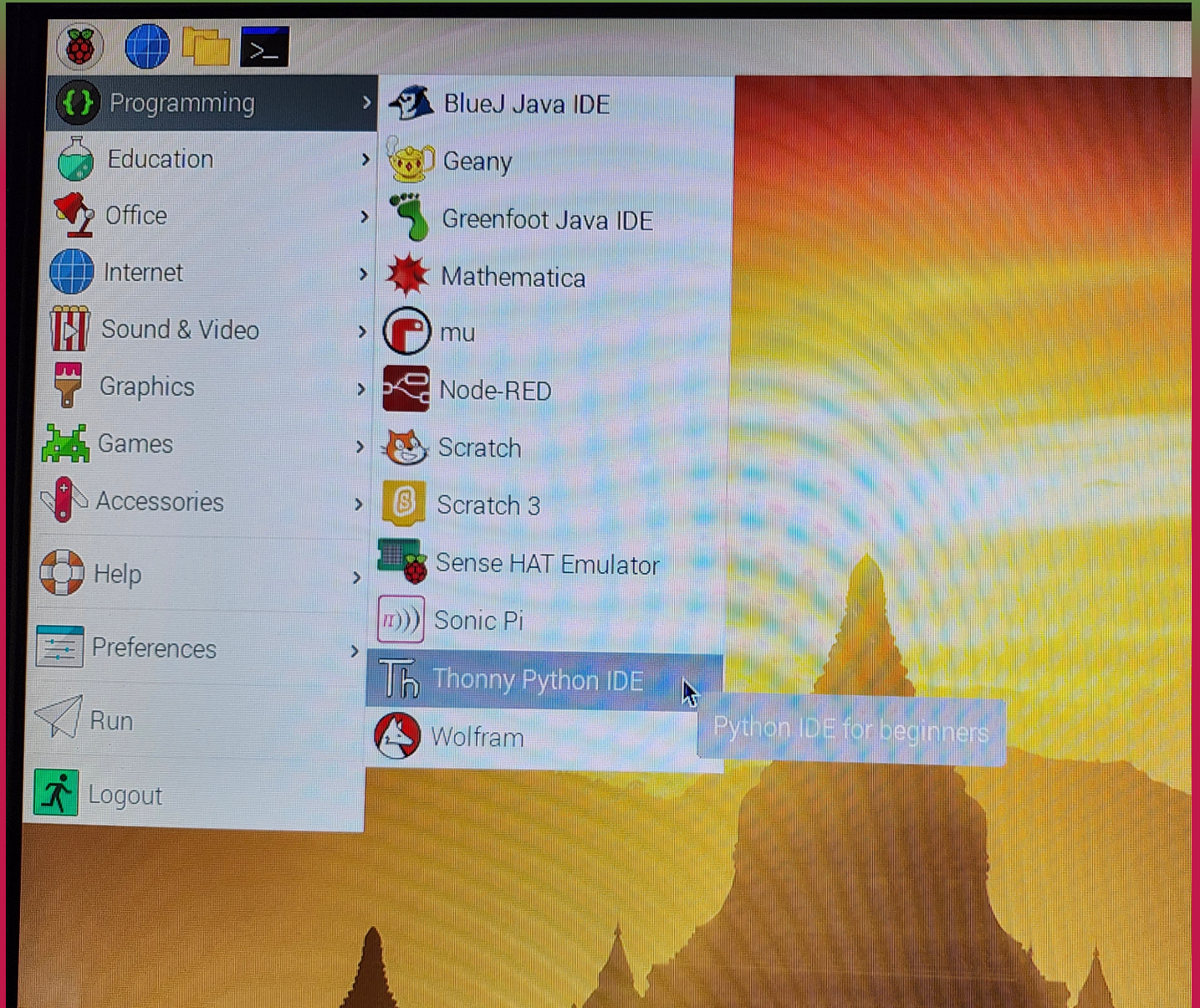
Most Excellent!





# Python

## Using Thony IDE





# Python

Type Program  
Click Run Button

Program

Run

Thonny - /home/pi/hello.py @ 1:23

File Edit View Run Tools Help

hello.py

```
print("Hello, World!")
```

Shell

>>>

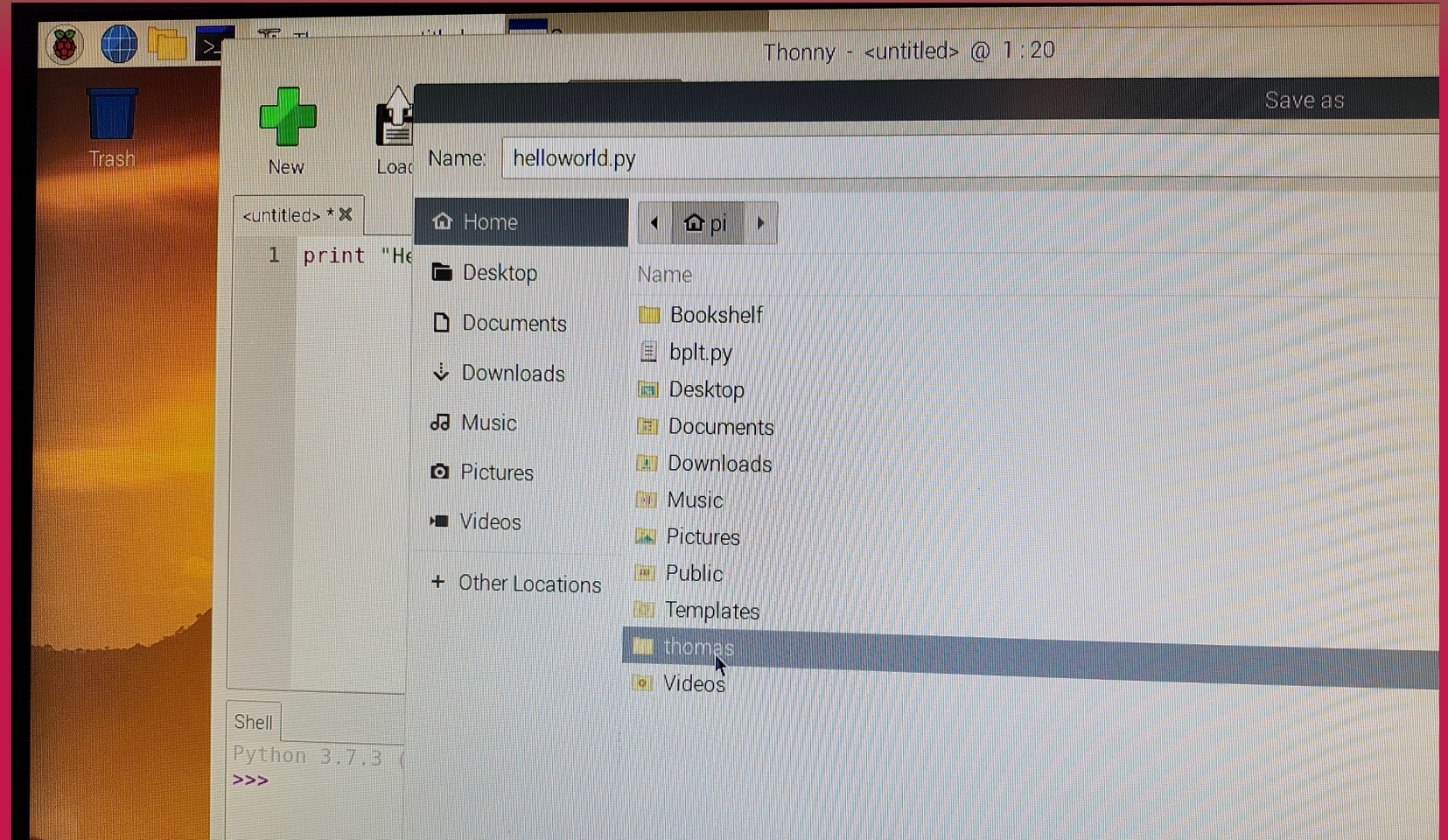


# Python

Name File

Select Folder

Click OK button





# Python

Thonny - /home/pi/hello.py @ 1:23

File Edit View Run Tools Help

hello.py

```
print("Hello, World!")
```

Shell

```
Hello, World!  
>>>
```

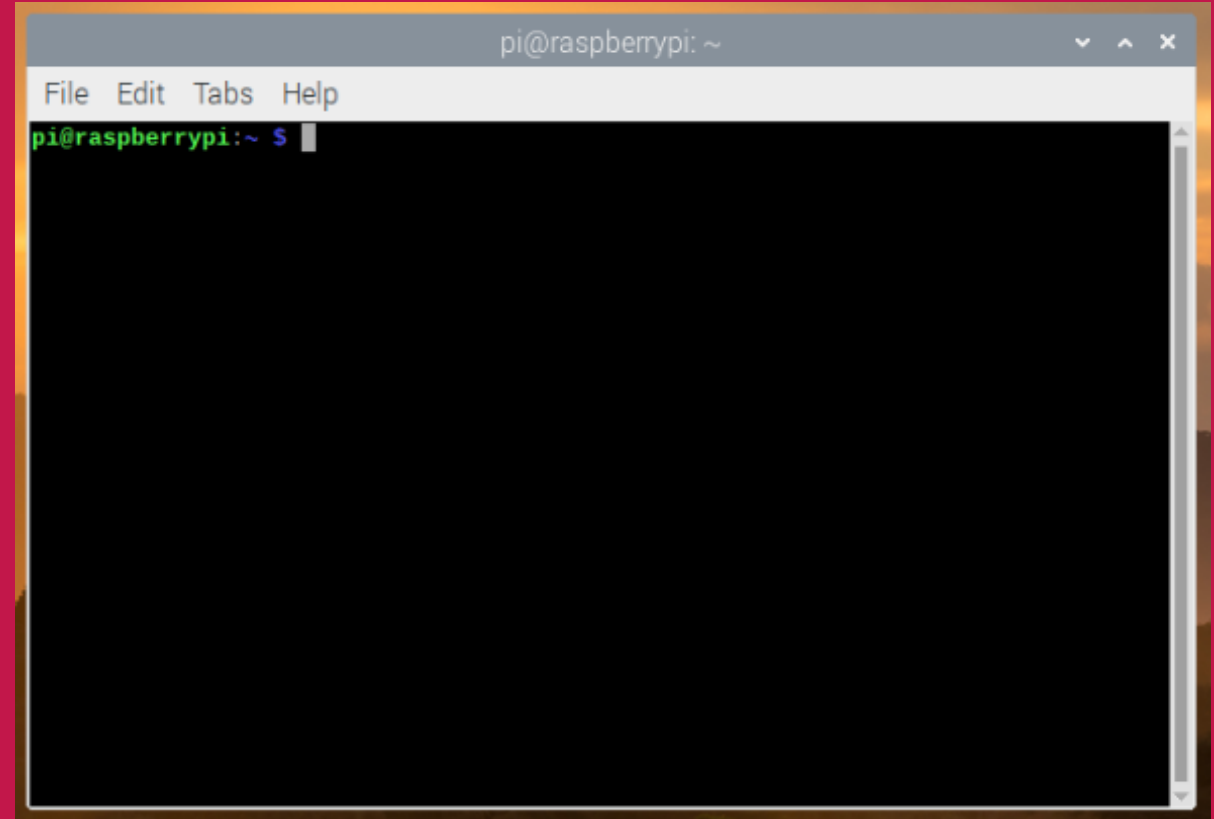
Program  
output





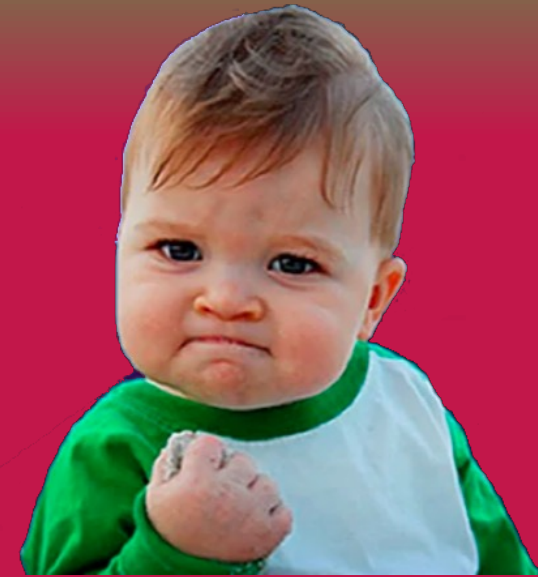
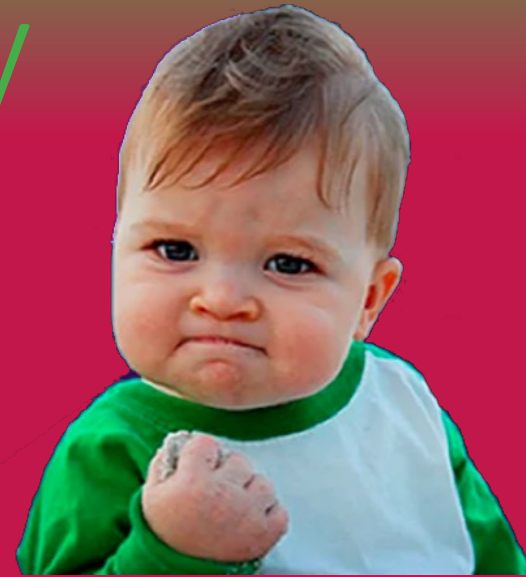
# Python via Terminal

```
python ./helloworld.py
```





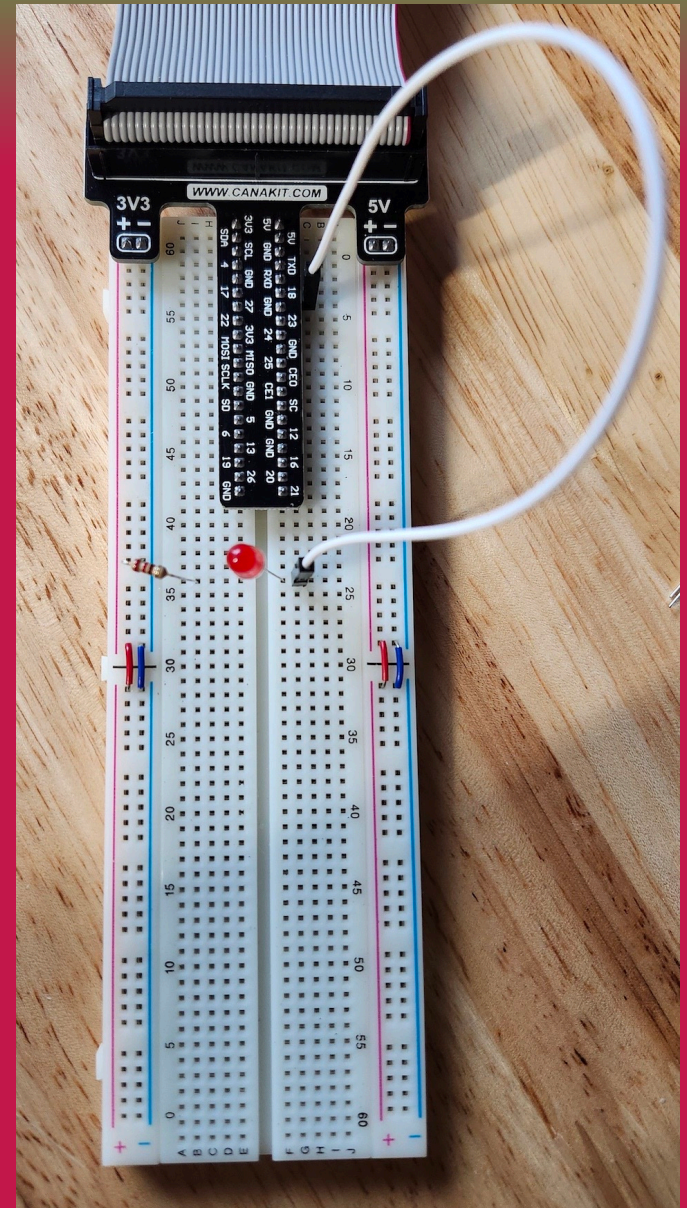
Awesomly  
Excellent!





# 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

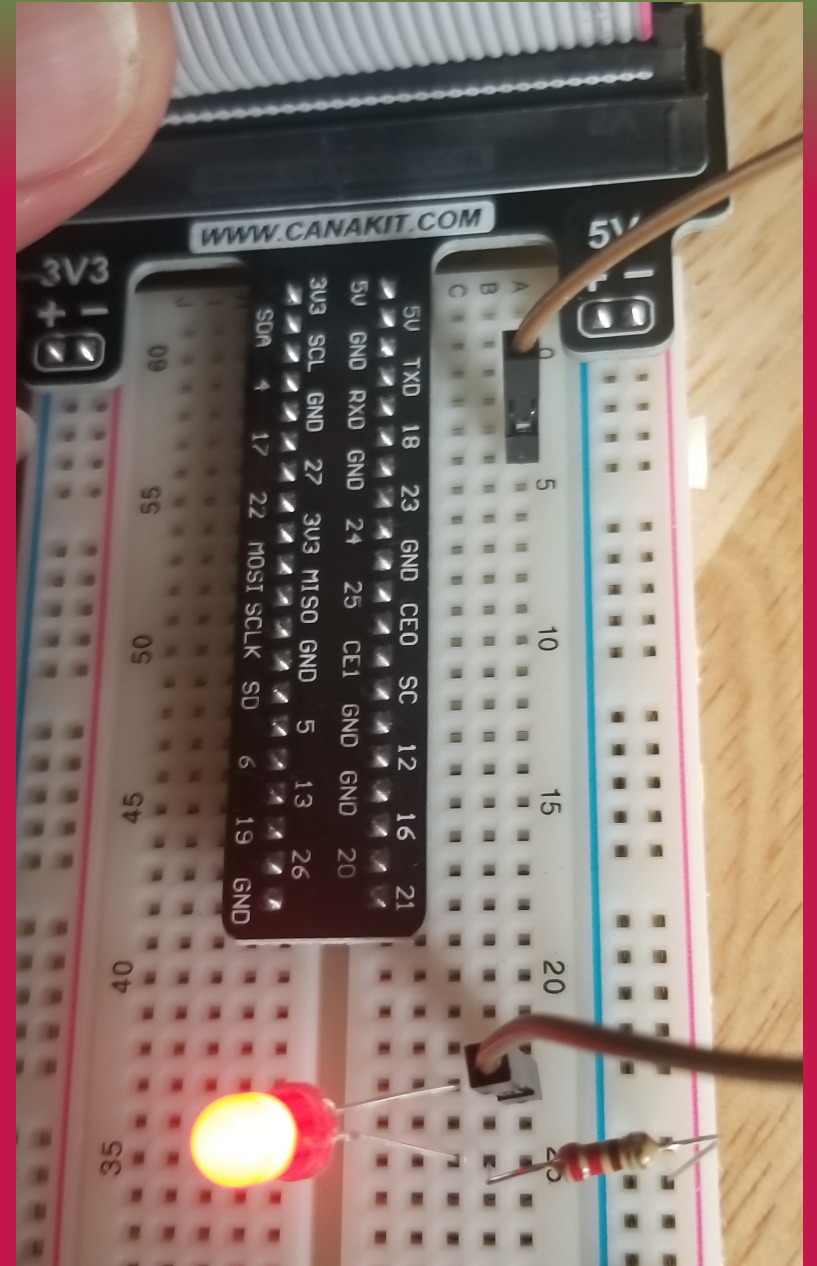
Polarity matters on LED

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

Resistor 220 $\Omega$  (Red Red Brown)

Wire

LED Anode to GPIO 18





# GPIO LED - Software

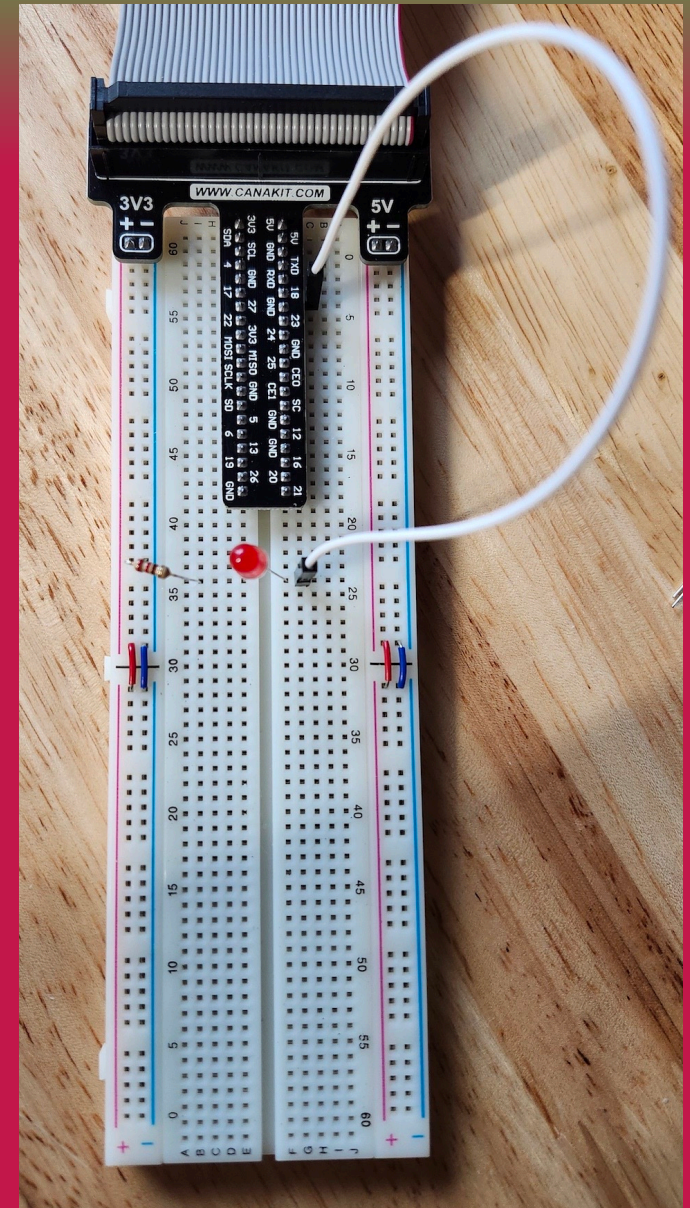
Thony

```
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

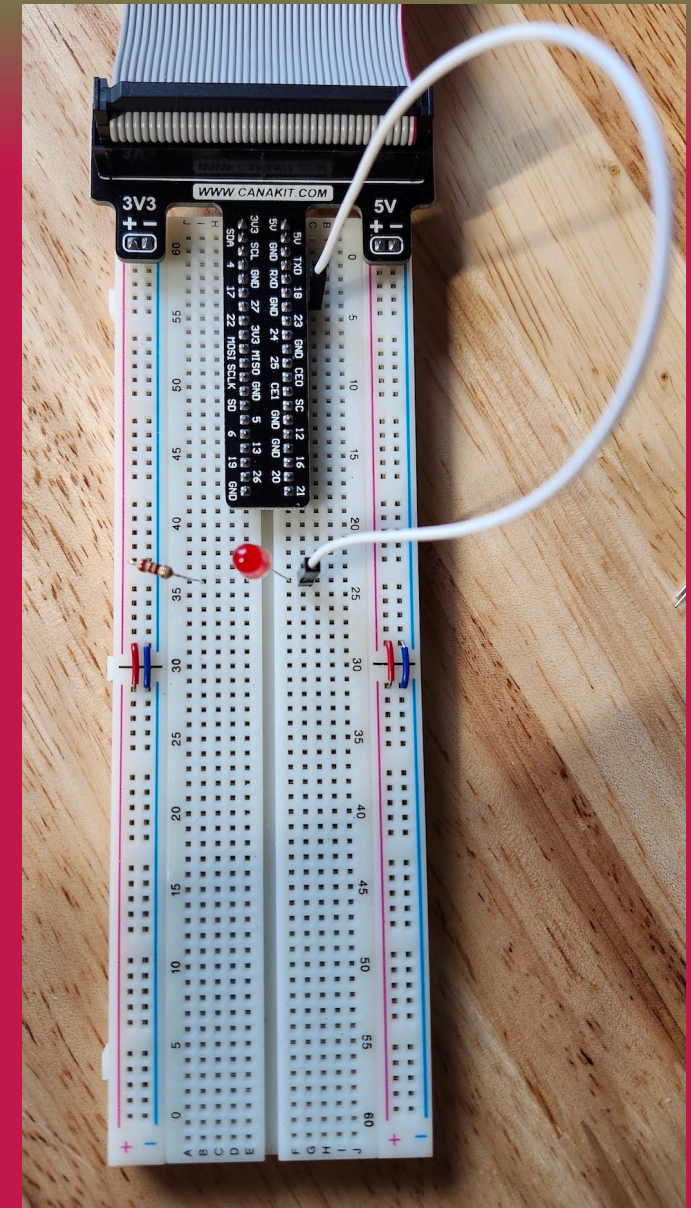
Thony

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

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

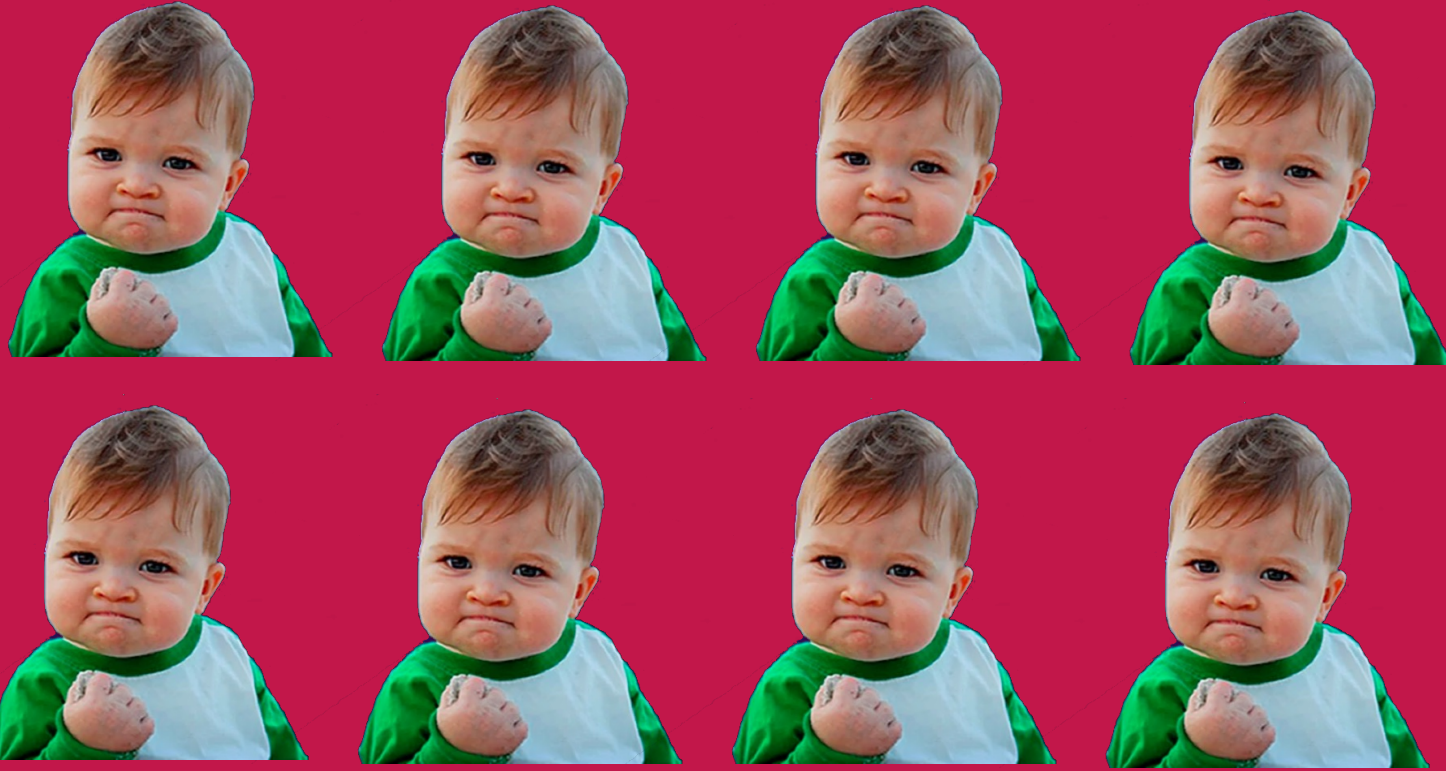
```
while True:
```

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





# Most Awesomely Excellent!





**BREAK  
TIME !!**



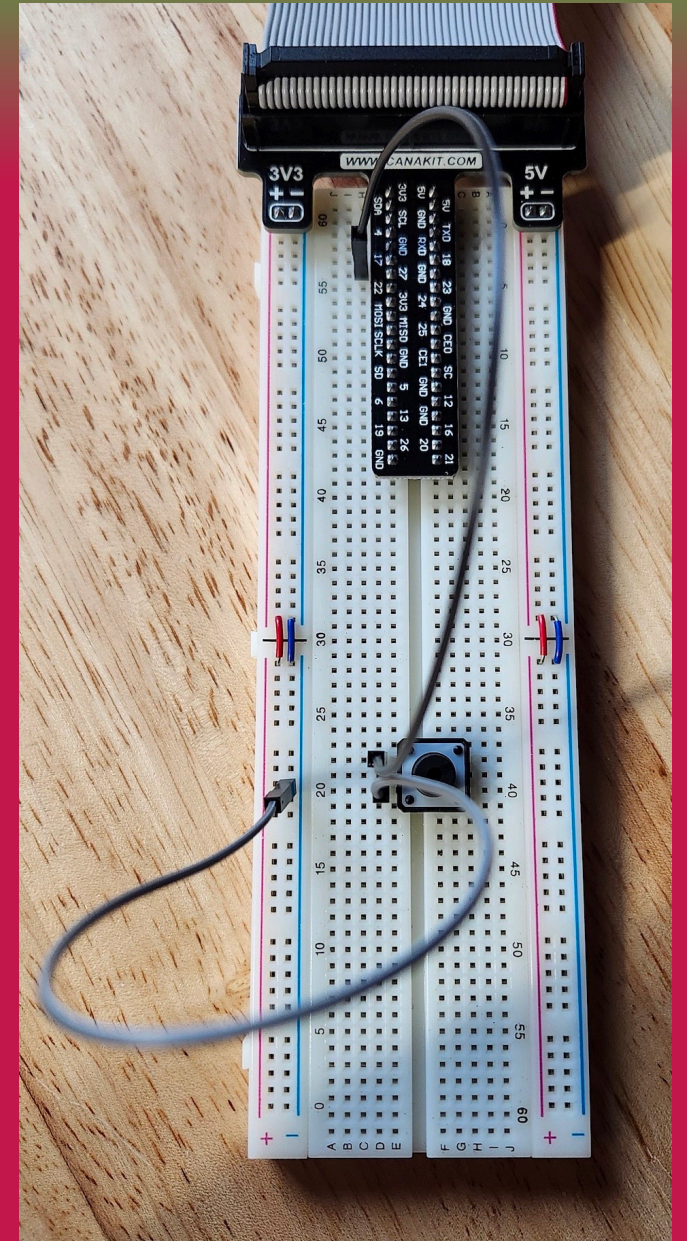
# GPIO Switch

Hardware

Software

Pressing switch will turn on LED

LED will turn off when switch is released





# GPIO Switch - Hardware

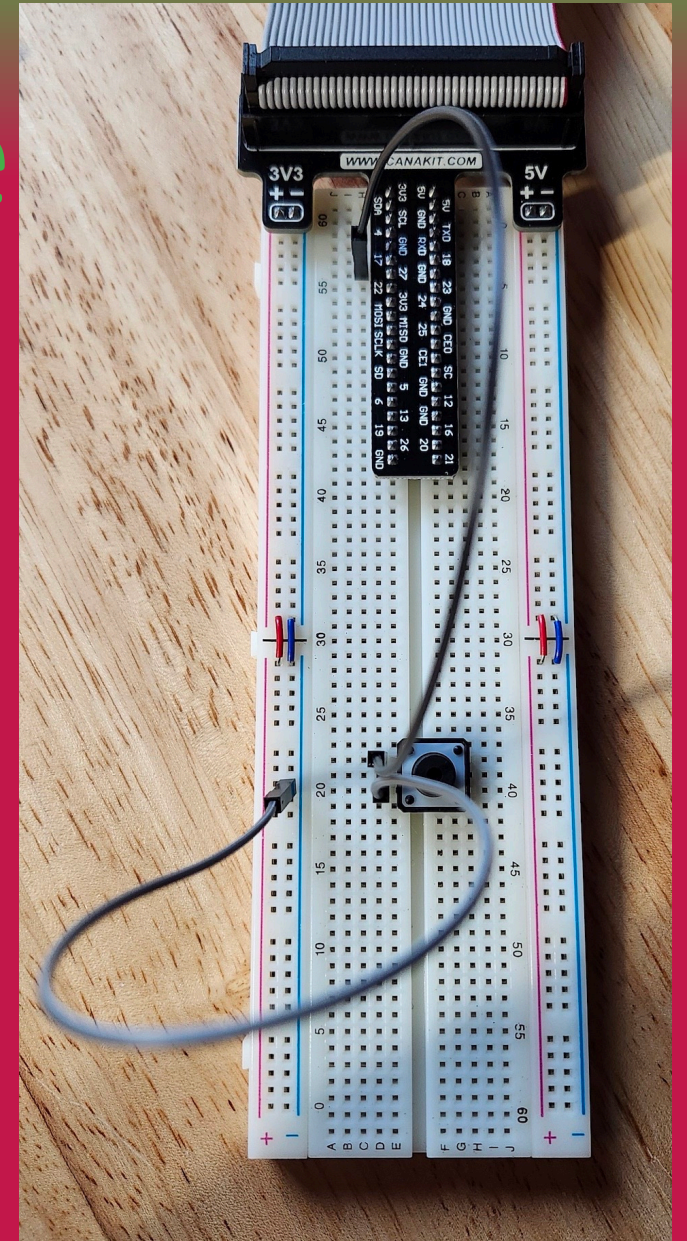
Digital Input

Polarity does NOT matter on Switch

Wires

GPIO 17

- (GND)





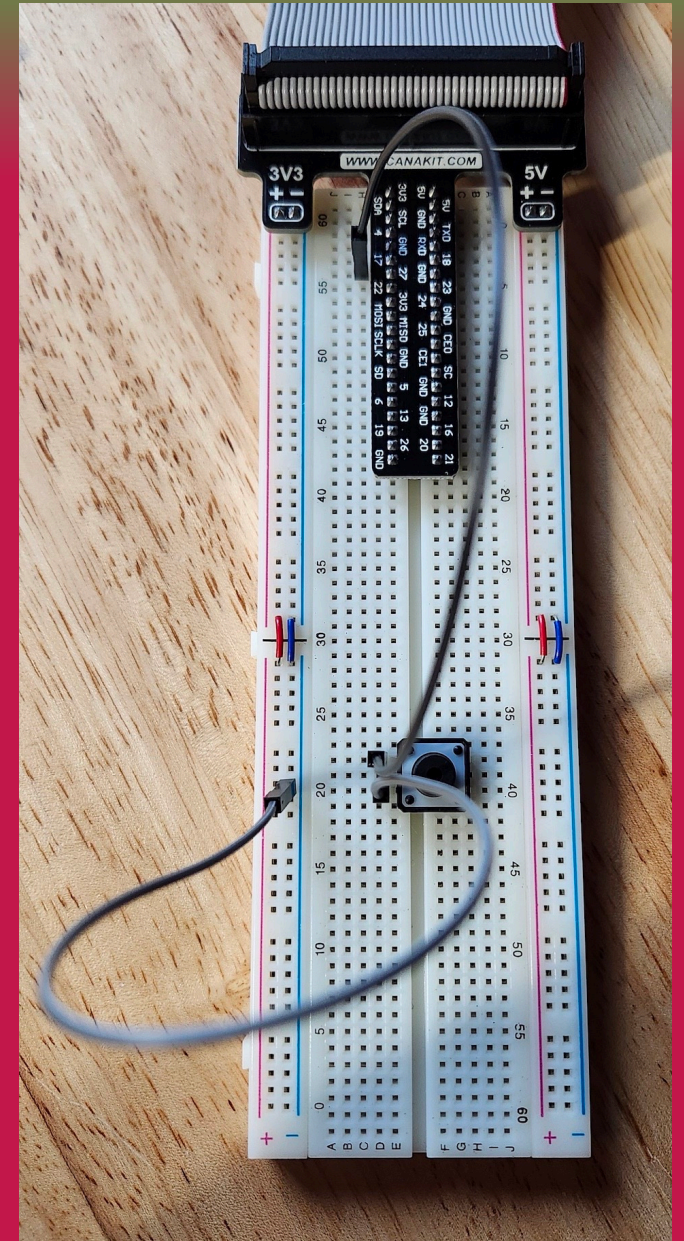
# GPIO Switch - Software

```
#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

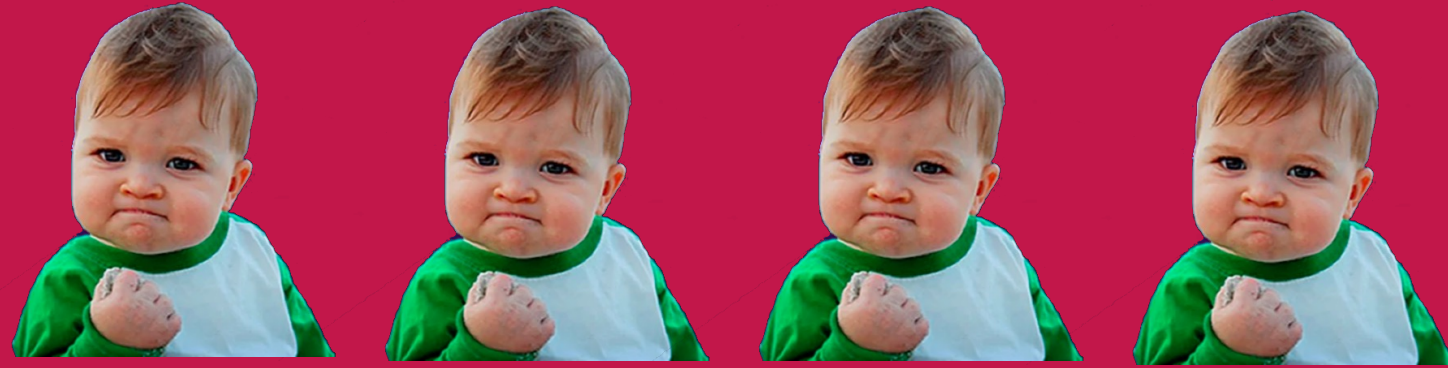
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN)
```

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





# Most Awesomely Excellent!



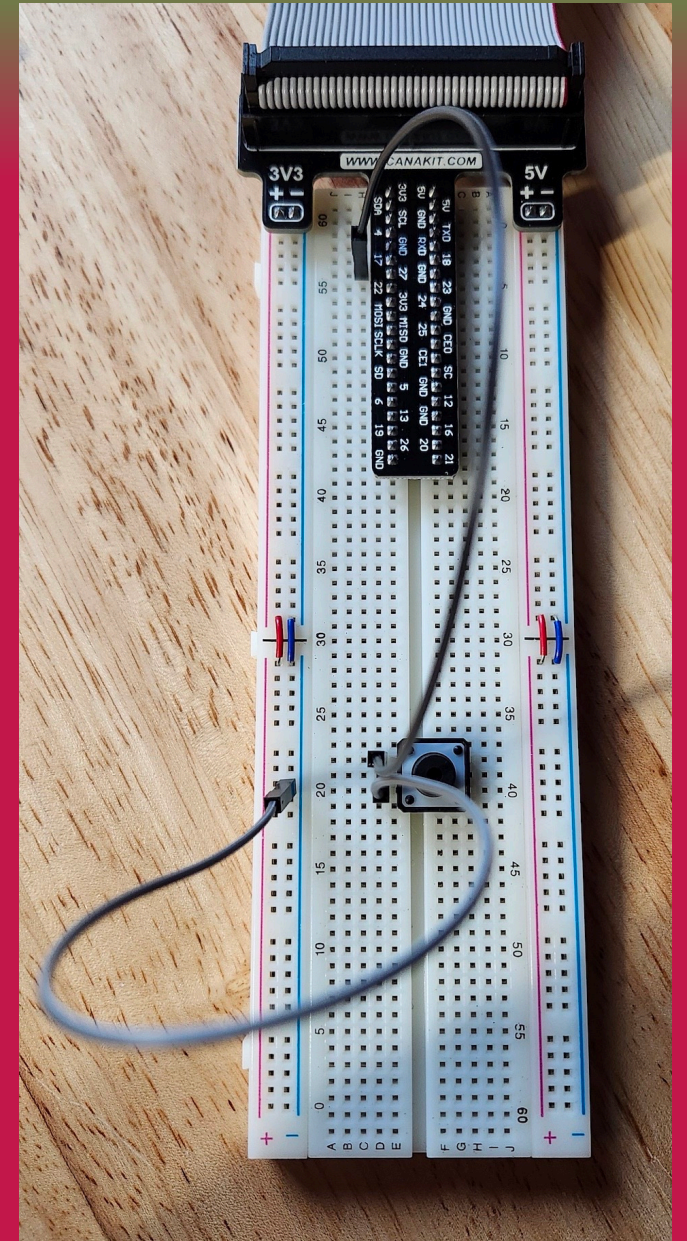


# File Write

Hardware

Software

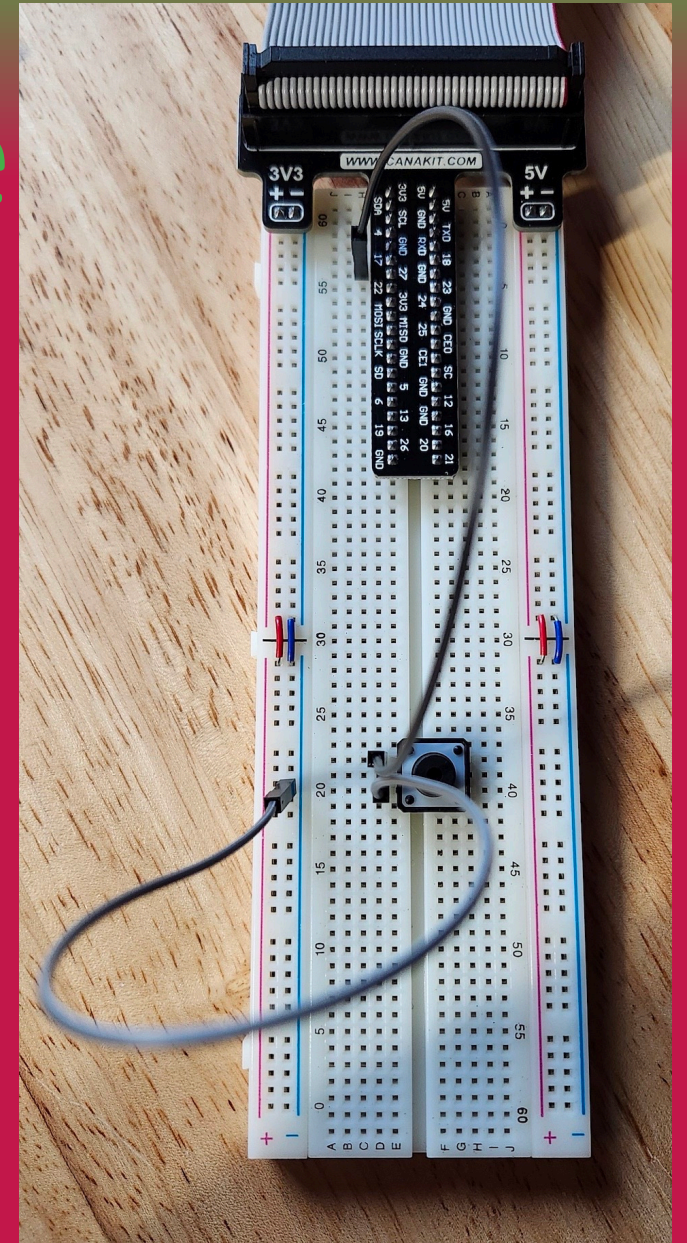
Pressing switch will  
write current time to file





# 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)
```

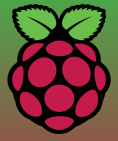
```
    # Wait for button to be released
```

```
        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!



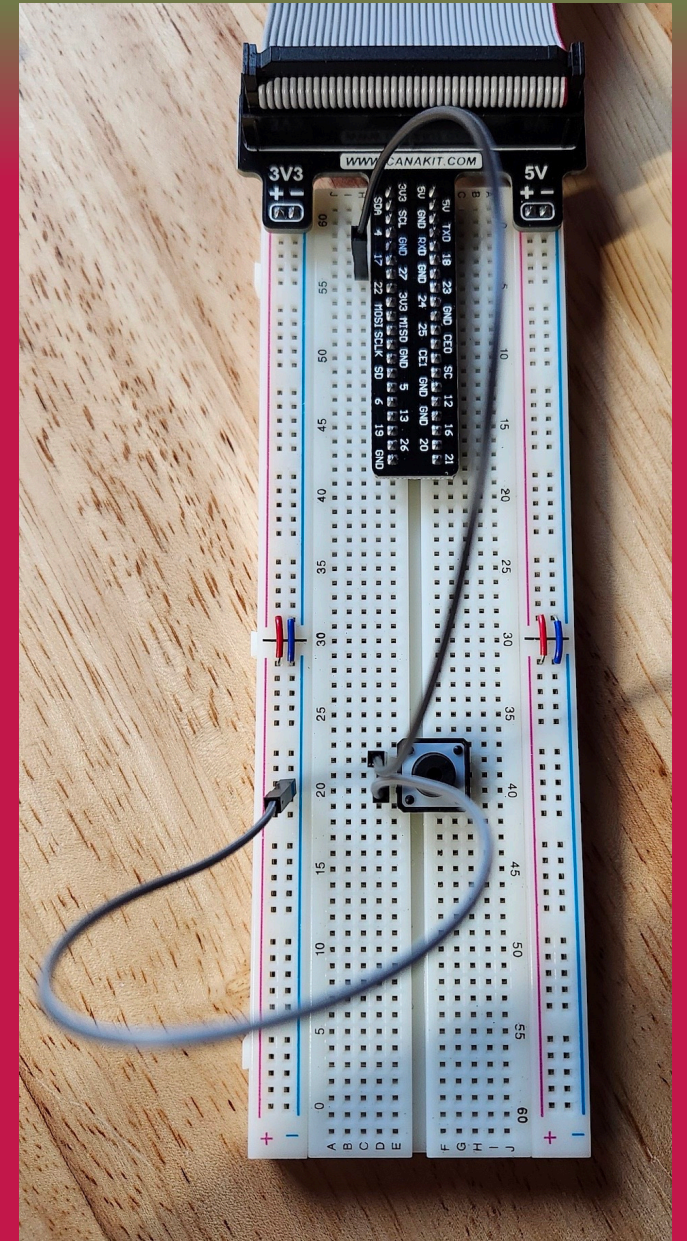


# Cronjob Alarm

Hardware

Software

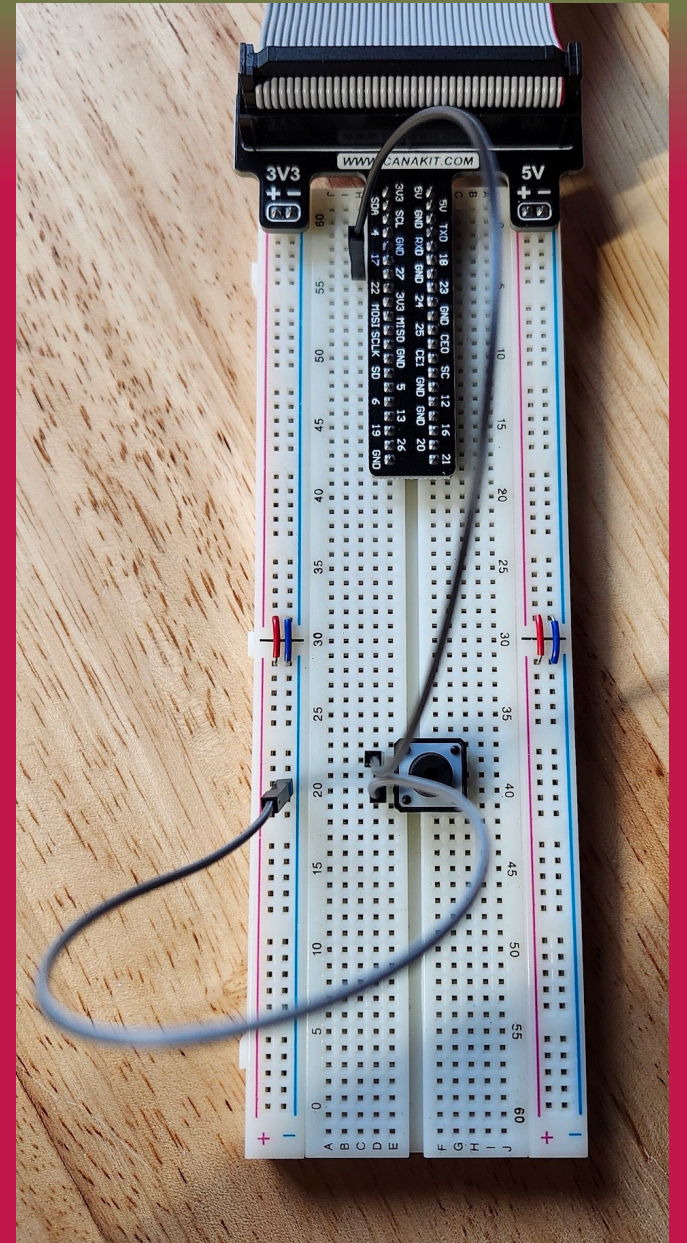
At a specific time blink an LED until a switch is pressed





# Cronjob - Hardware

Same



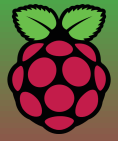


# Cronjob - Software

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



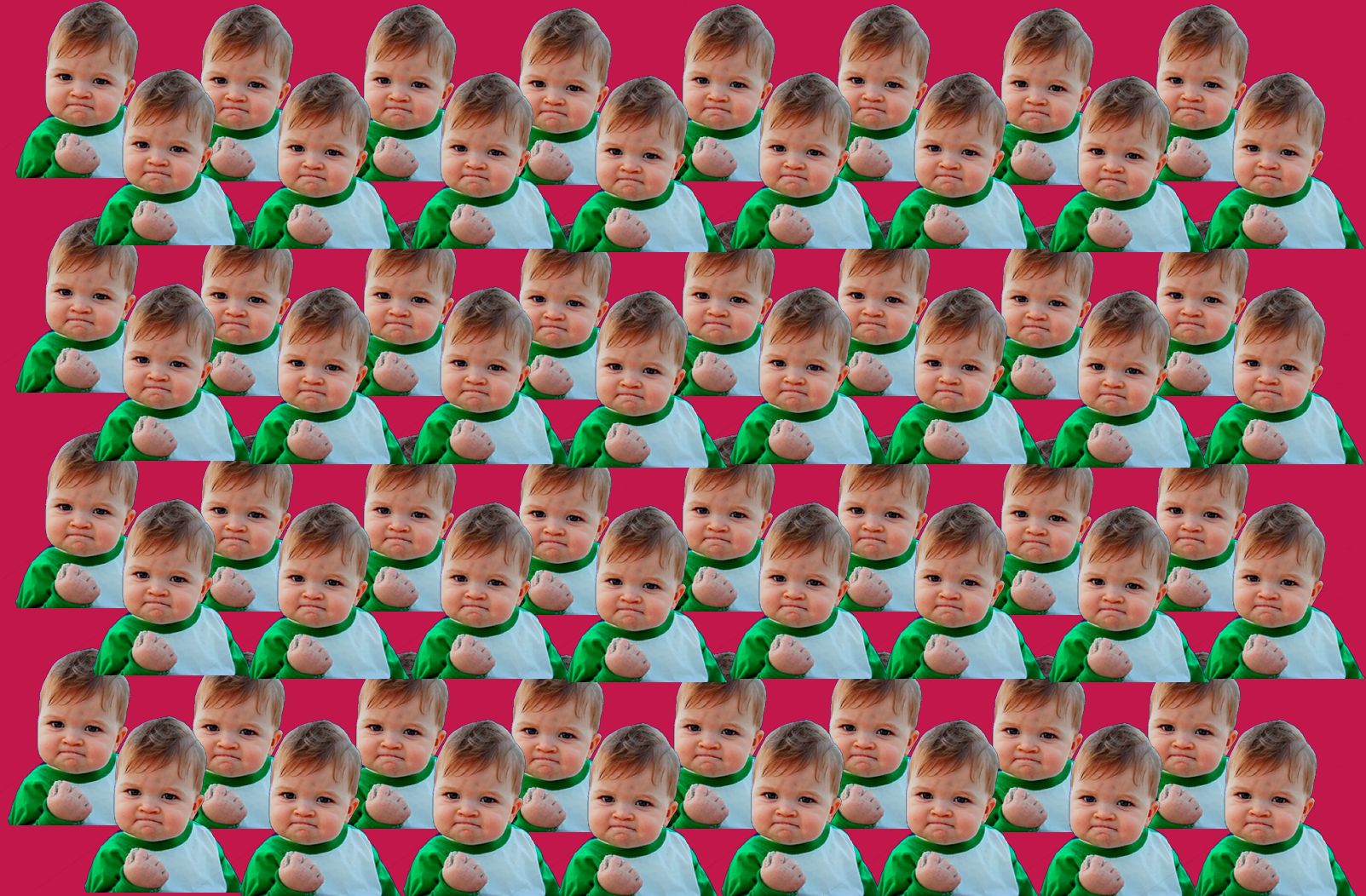
```
15 13 * * * /home/pi/<yourname>/alarm.sh
```

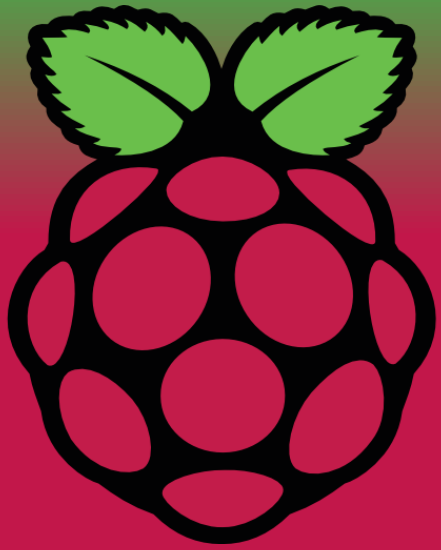
# Cronjob





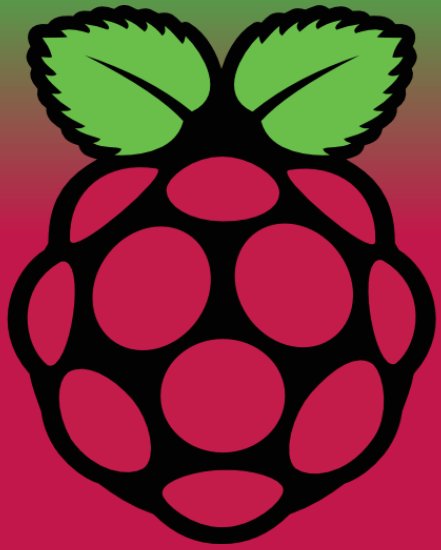
# Unbelievably Excellent!





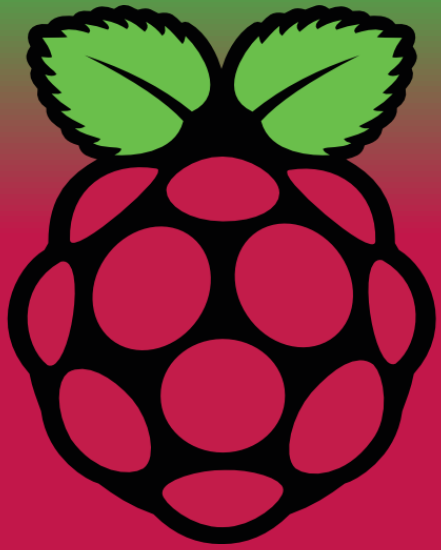
# 2024 Raspberry Pi Workshop





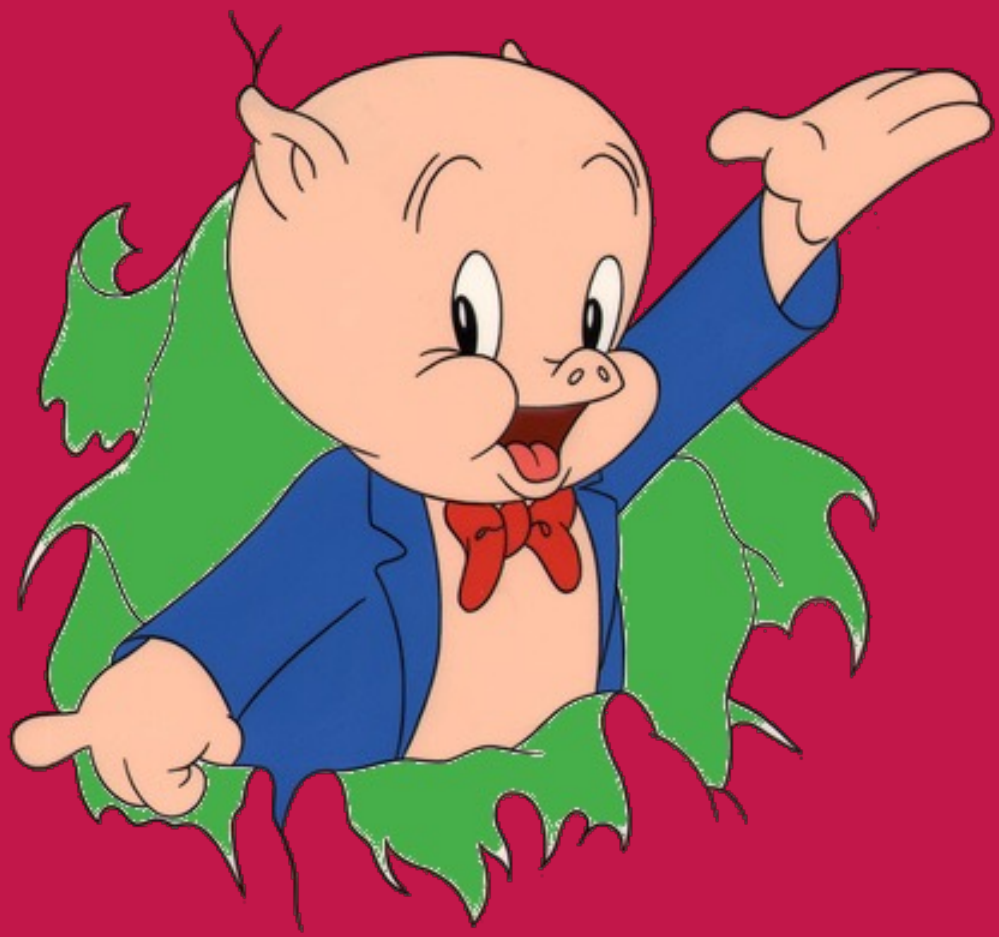
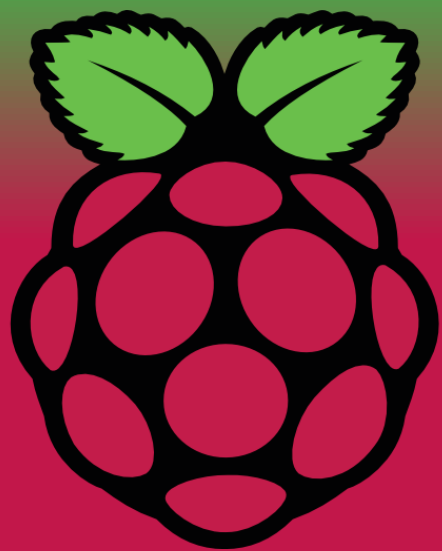
# 2024 Raspberry Pi Workshop





# 2024 Raspberry Pi Workshop







# More



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

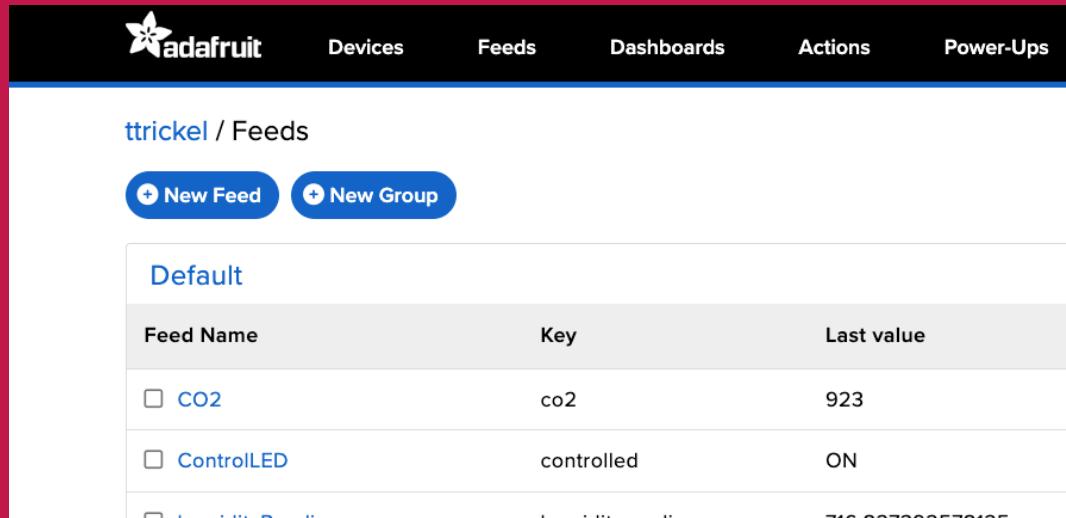
# Feeds & Dashboards

- 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

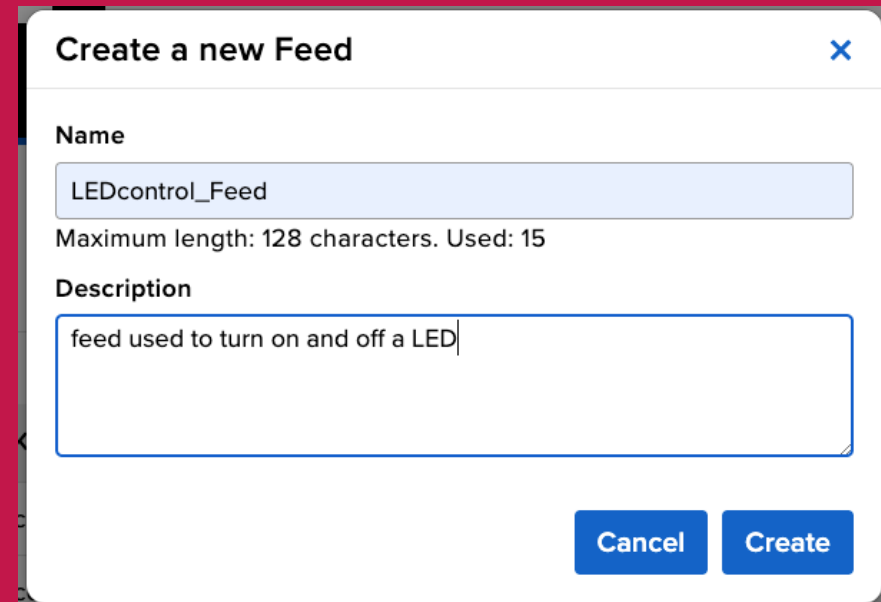
- Hardware is the same
- Create Feed
- Create Dashboard
- Write Python Code

# Create AIO Feed



The screenshot shows the Adafruit dashboard with the 'Feeds' tab selected. The breadcrumb 'ttrickel / Feeds' is visible. There are two buttons: '+ New Feed' and '+ New Group'. Below is a table with the following data:

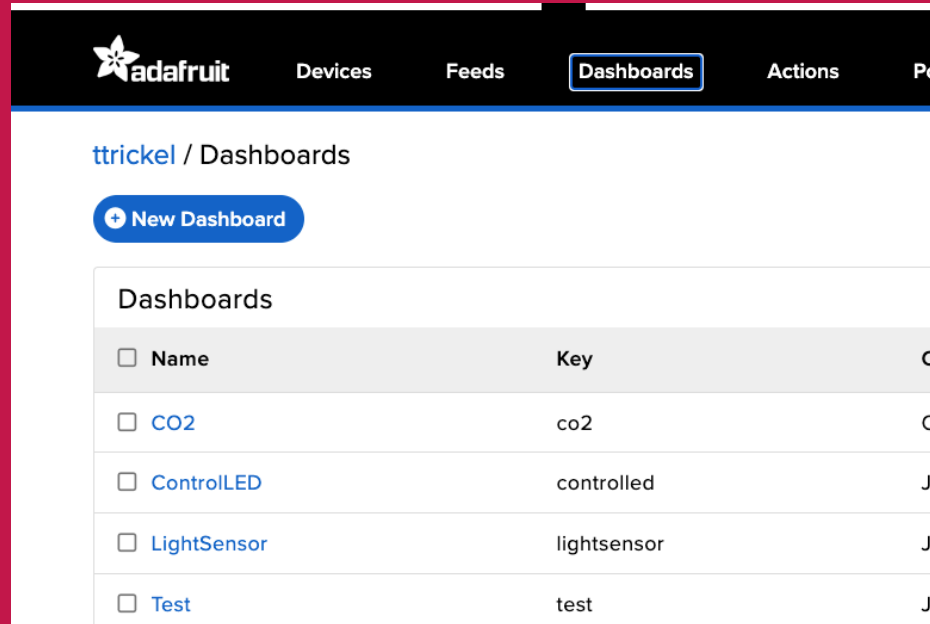
Feed Name	Key	Last value
<input type="checkbox"/> CO2	co2	923
<input type="checkbox"/> ControlLED	controlled	ON
<input type="checkbox"/> Humidity Reading	humidityreading	716.827202570125



The dialog box is titled 'Create a new Feed' and contains the following fields and buttons:

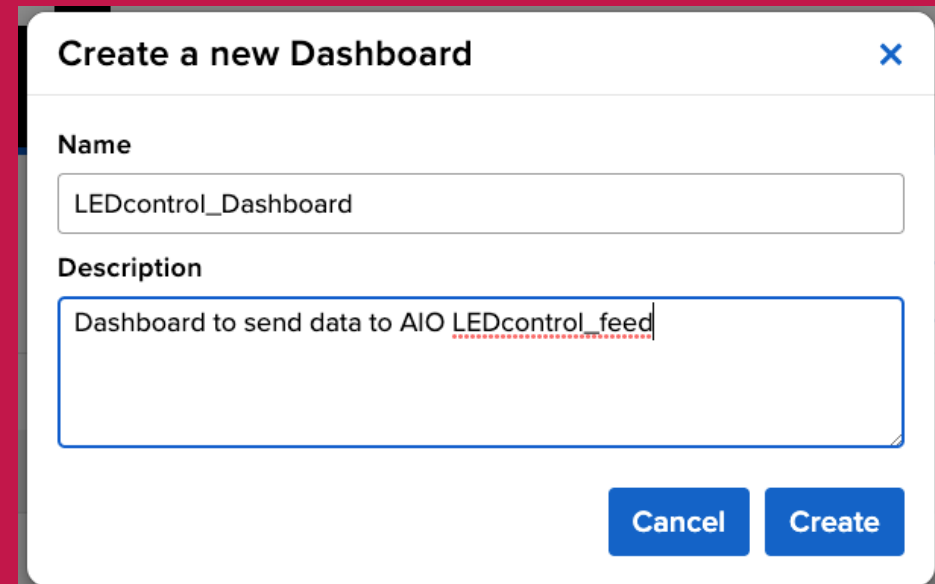
- Name:** A text input field containing 'LEDcontrol\_Feed'. Below it, a message reads 'Maximum length: 128 characters. Used: 15'.
- Description:** A text area containing 'feed used to turn on and off a LED'.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

# Create AIO Dashboard



The screenshot shows the Adafruit AIO Dashboards interface. At the top, there is a navigation bar with the Adafruit logo and menu items: Devices, Feeds, Dashboards (highlighted), Actions, and Profile. Below the navigation bar, the user's name 'ttrickel' and the page title 'Dashboards' are displayed. A blue button with a plus sign and the text 'New Dashboard' is located below the title. The main content area is a table titled 'Dashboards' with the following data:

<input type="checkbox"/>	Name	Key	
<input type="checkbox"/>	CO2	co2	C
<input type="checkbox"/>	ControlLED	controlled	J
<input type="checkbox"/>	LightSensor	lightsensor	J
<input type="checkbox"/>	Test	test	J

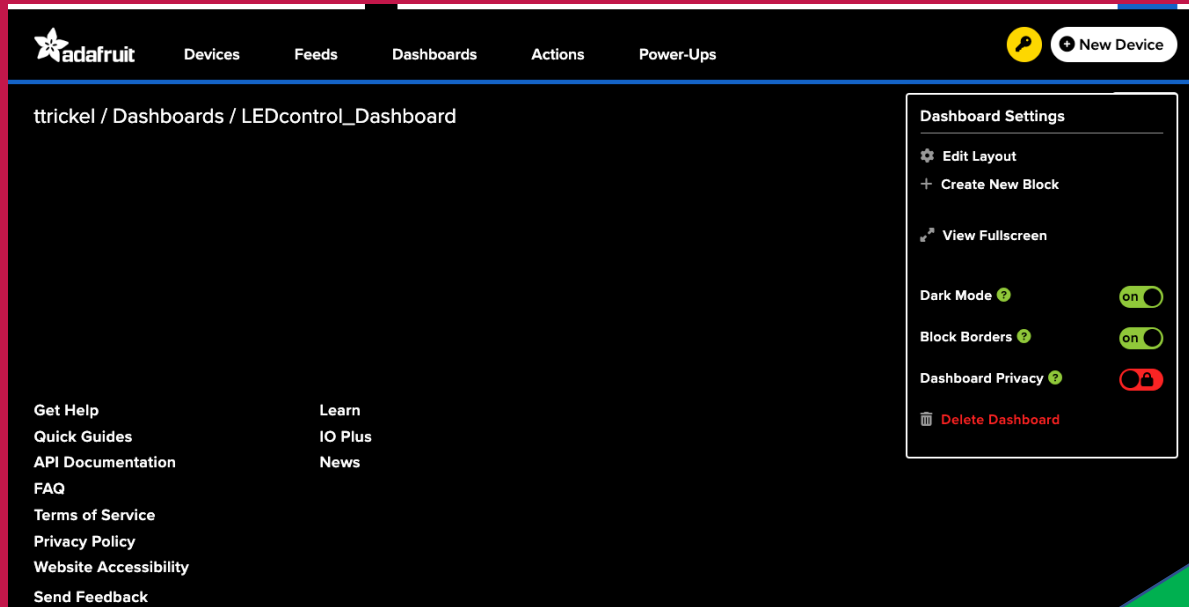


The screenshot shows a modal window titled 'Create a new Dashboard' with a close button (X) in the top right corner. The form contains the following fields:

- Name:** A text input field containing the text 'LEDcontrol\_Dashboard'.
- Description:** A text area containing the text 'Dashboard to send data to AIO LEDcontrol\_feed'.

At the bottom right of the modal, there are two buttons: 'Cancel' and 'Create'.

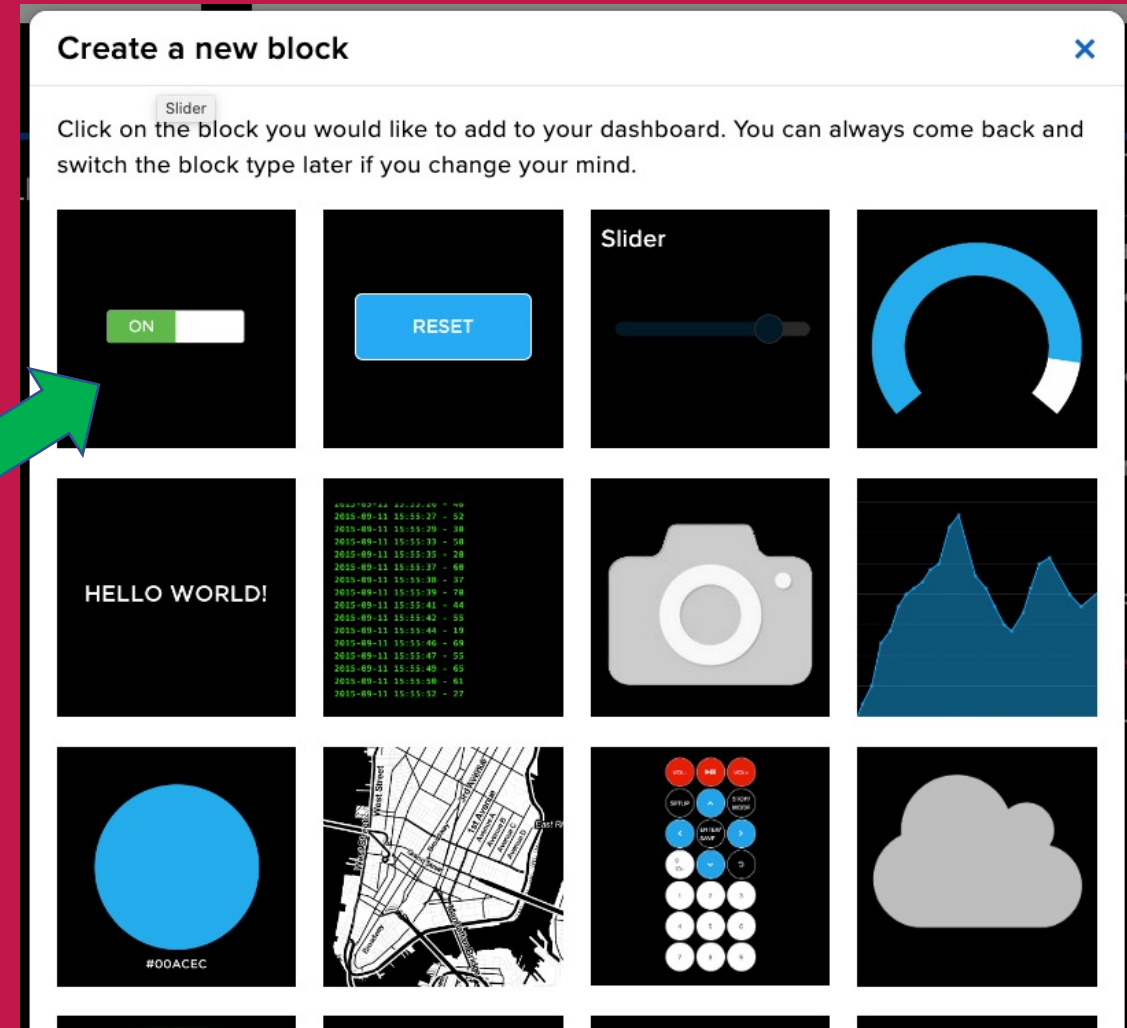
# Create AIO Dashboard continued



The screenshot shows the Adafruit dashboard interface. At the top, there is a navigation bar with the Adafruit logo and links for Devices, Feeds, Dashboards, Actions, and Power-Ups. A 'New Device' button is visible in the top right. The main content area shows the path 'ttrickel / Dashboards / LEDcontrol\_Dashboard'. On the right side, there is a 'Dashboard Settings' panel with the following options:

- Edit Layout
- Create New Block
- View Fullscreen
- Dark Mode: on
- Block Borders: on
- Dashboard Privacy: off
- Delete Dashboard

On the left side, there is a sidebar with links for Get Help, Quick Guides, API Documentation, FAQ, Terms of Service, Privacy Policy, Website Accessibility, and Send Feedback. There are also links for Learn, IO Plus, and News.



The screenshot shows a 'Create a new block' dialog box. The title is 'Create a new block' and there is a close button (X) in the top right corner. Below the title, there is a 'Slider' label and a paragraph of text: 'Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.'

The dialog box displays a grid of block options:

- ON (toggle switch)
- RESET (button)
- Slider (slider control)
- Progress indicator (circular gauge)
- HELLO WORLD! (text)
- Terminal (code editor)
- Camera (camera icon)
- Line graph (line chart)
- Blue circle (circle)
- Map (map icon)
- Keypad (numeric keypad)
- Cloud (cloud icon)

A green arrow points from the 'ON' block in the dialog box to the 'ON' block in the dashboard settings panel.

# Create AIO Dashboard continued

Choose a single feed you would like to connect to this toggle. You can also create a new feed within a group.

## Default

Feed Name	Last value	Recorded	
<input type="checkbox"/> CO2	923	7 months	🔒
<input type="checkbox"/> ControlLED	ON	2 days	🔒
<input type="checkbox"/> humidityReading	716.827392578...	7 months	🔒
<input type="checkbox"/> ledControl	off	2 days	🔒
<input checked="" type="checkbox"/> LEDcontrol_Feed		5 minutes	🔒
<input type="checkbox"/> lightReading	0	over 4 years	🔒
<input type="checkbox"/> runTime	9:6:0	over 4 years	🔒
<input type="checkbox"/> switch	0	2 days	🔒
<input type="checkbox"/> switchTest	0	1 day	🔒
<input type="checkbox"/> temperatureReading	698.892364501...	7 months	🔒

1 of 1 feeds selected

< Previous step

Next step >

click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Button On Text

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

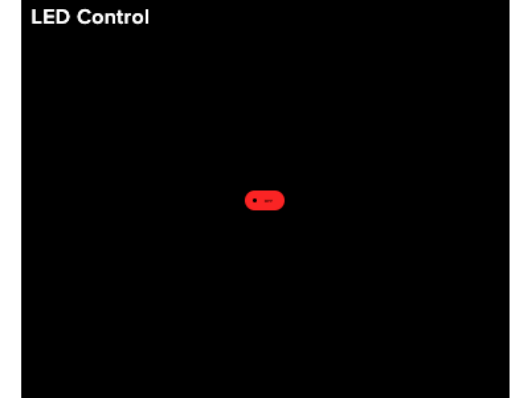
Button On Value (uses On Text if blank)

Button Off Text

Limit of 6 characters for the toggle text. Use the block title to be more descriptive.

Button Off Value (uses Off Text if blank)

Block Preview



Toggle A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.

Test Value

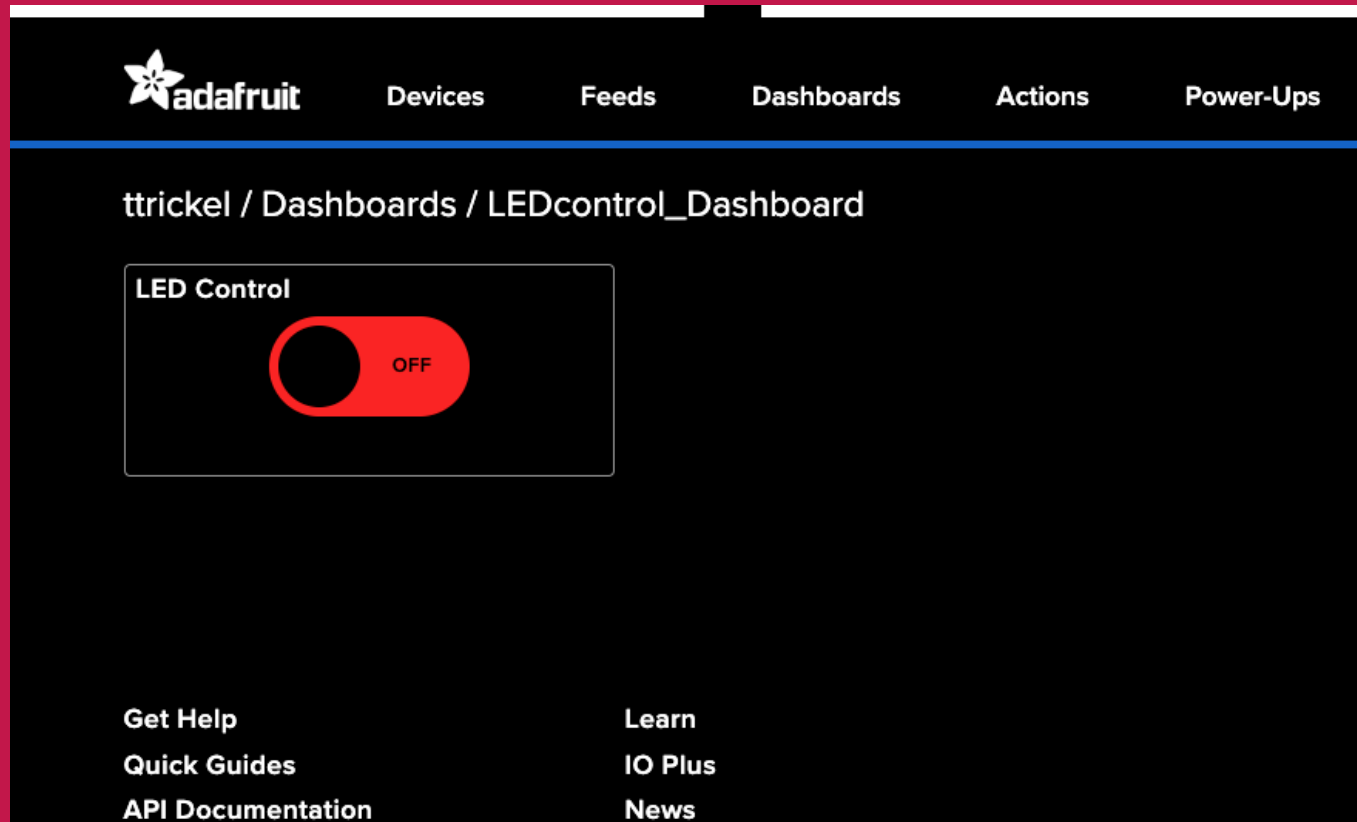
Published Value

0 bytes

< Previous step

Create block

# Create AIO Dashboard continued



# Control LED from AIO Python Code

change

YOUR\_AIO\_KEY

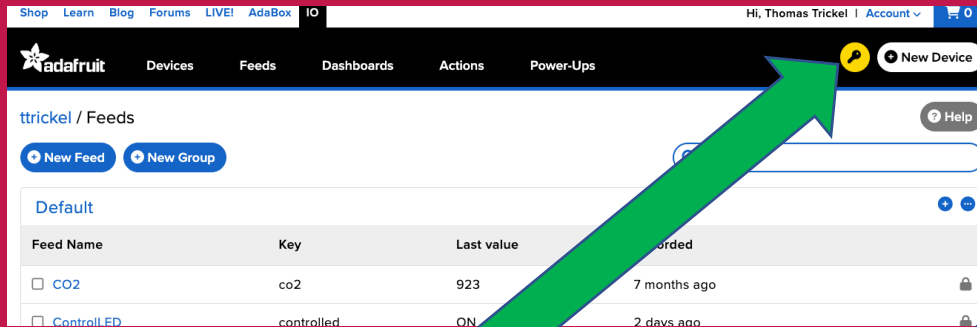


YOUR\_AIO\_USERNAME



```
1  """
2  'digital_out.py'
3  =====
4  Example of turning on and off a LED
5  from the Adafruit IO Python Client
6  Author(s): Brent Rubell, Todd Treece
7             with modifications by Thomas Trickel
8  """
9  # Import standard python modules
10 import time
11 import RPi.GPIO as GPIO
12
13 # import Adafruit IO REST client.
14 from Adafruit_IO import Client, Feed, RequestError
15
16 # setup GPIO
17 GPIO.setwarnings(False)
18 GPIO.setmode(GPIO.BCM)
19 GPIO.setup(18, GPIO.OUT)
20
21
22 # Set to your Adafruit IO key.
23 # Remember, your key is a secret,
24 # so make sure not to publish it when you publish this code!
25 ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'
26
27 # Set to your Adafruit IO username.
28 # (go to https://accounts.adafruit.com to find your username)
29 ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'
30
31 # Create an instance of the REST client.
32 aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
33
34 try: # if we have a 'digital' feed
35     digital = aio.feeds('ledcontrol-feed')
36 except RequestError: # create a digital feed
37     feed = Feed(name="ledcontrol-feed")
38     digital = aio.create_feed(feed)
39
40 while True:
41     data = aio.receive(digital.key)
42     if data.value == "ON":
43         print('received <- ON\n')
44         GPIO.output(18, True)
45     elif data.value == "OFF":
46         print('received <- OFF\n')
47         GPIO.output(18, False)
48
49 # timeout so we dont flood adafruit-io with requests
50 time.sleep(0.5)
51
```

# AIO KEY & USERNAME



Shop Learn Blog Forums LIVE! AdaBox IO Hi, Thomas Trickel | Account

adafruit Devices Feeds Dashboards Actions Power-Ups

ttrickel / Feeds

New Feed New Group

Feed Name	Key	Last value	Recorded
CO2	co2	923	7 months ago
ControlLED	controlled	ON	2 days ago

## 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

Active Key

REGENERATE KEY

[Hide Code Samples](#)

Arduino

```
#define IO_USERNAME "ttrickel"
#define IO_KEY      "a0da523dfa16502494a70e61e1f683ba8a0b32f4"
```

Linux Shell

```
export IO_USERNAME="ttrickel"
export IO_KEY="a0da523dfa16502494a70e61e1f683ba8a0b32f4"
```

Scripting

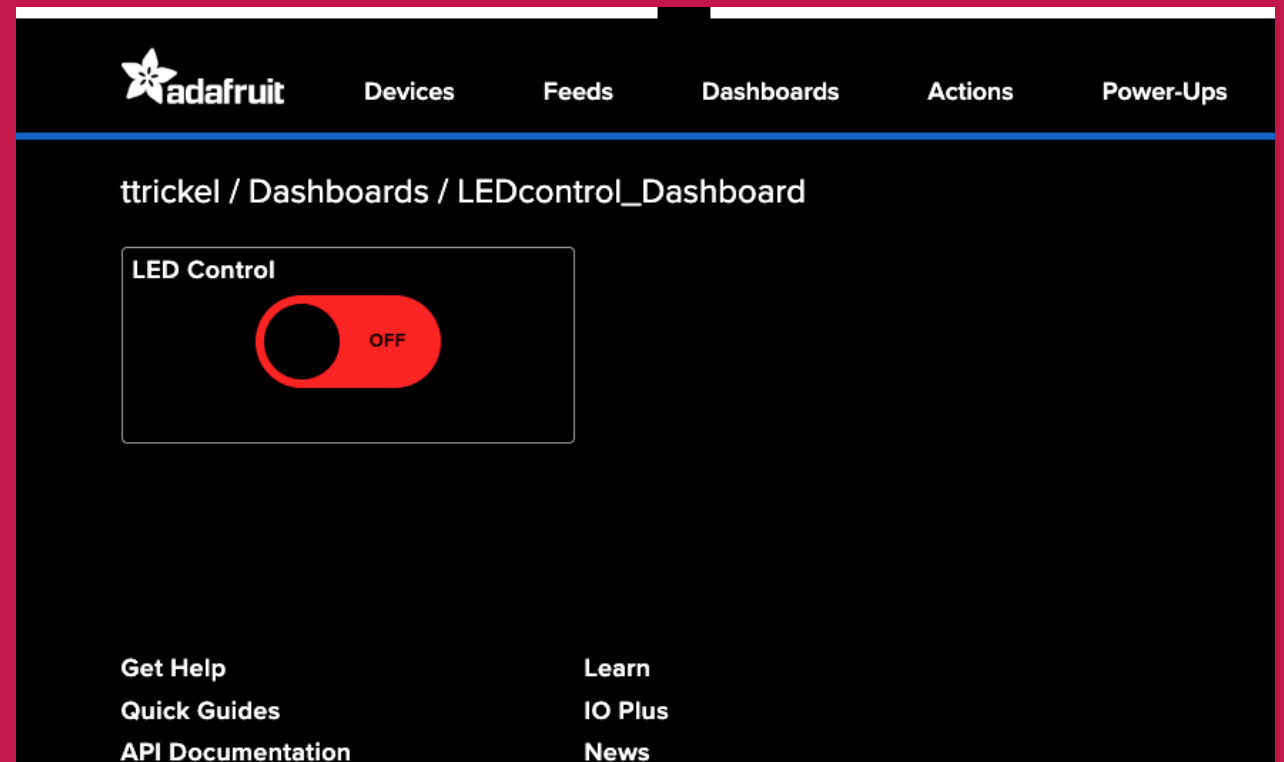
```
ADAFRUIT_IO_USERNAME = "ttrickel"
ADAFRUIT_IO_KEY = "a0da523dfa16502494a70e61e1f683ba8a0b32f4"
```

# Test It

## Run Python Code



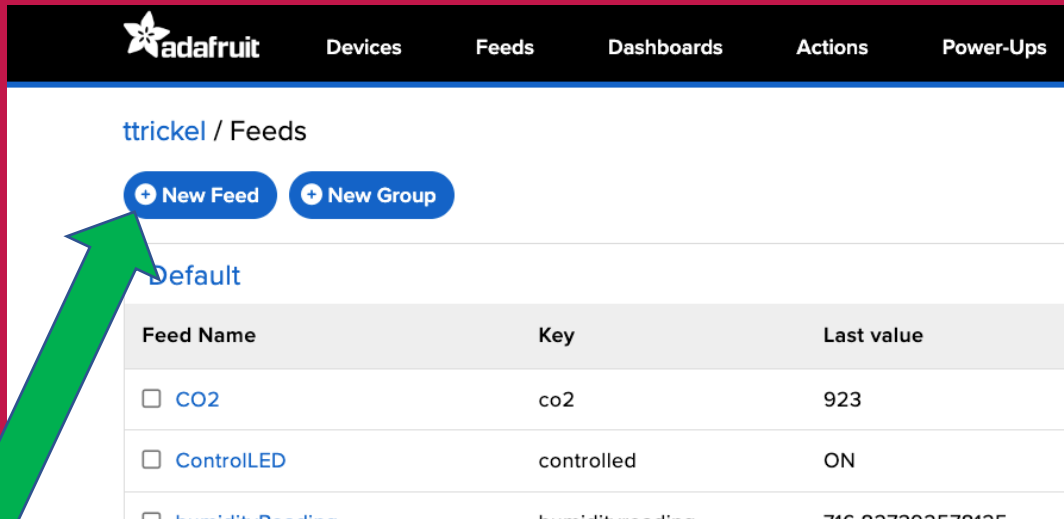
## Click on LED Control icon



# Send Switch Data to AIO

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

# Create AIO Feed



The screenshot shows the Adafruit IoT dashboard with the 'Feeds' section selected. The breadcrumb is 'ttrickel / Feeds'. There are two buttons: '+ New Feed' and '+ New Group'. A green arrow points to the '+ New Feed' button. Below the buttons is a 'Default' dropdown and a table of feeds.

Feed Name	Key	Last value
<input type="checkbox"/> CO2	co2	923
<input type="checkbox"/> ControlLED	controlled	ON
<input type="checkbox"/> Humidity Reading	humidityreading	716.827202570125

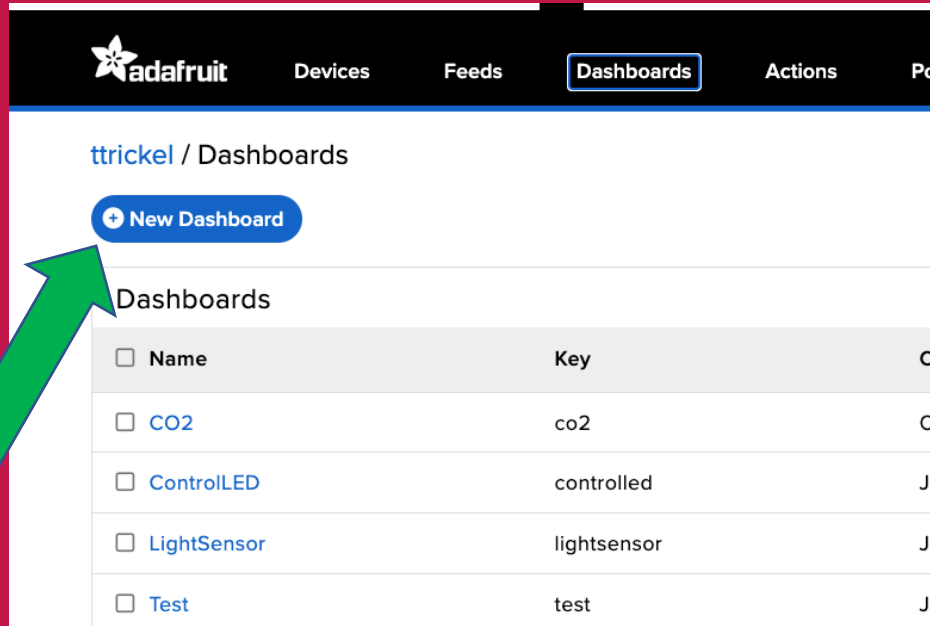
### Create a new Feed ✕

**Name**

Maximum length: 128 characters. Used: 15

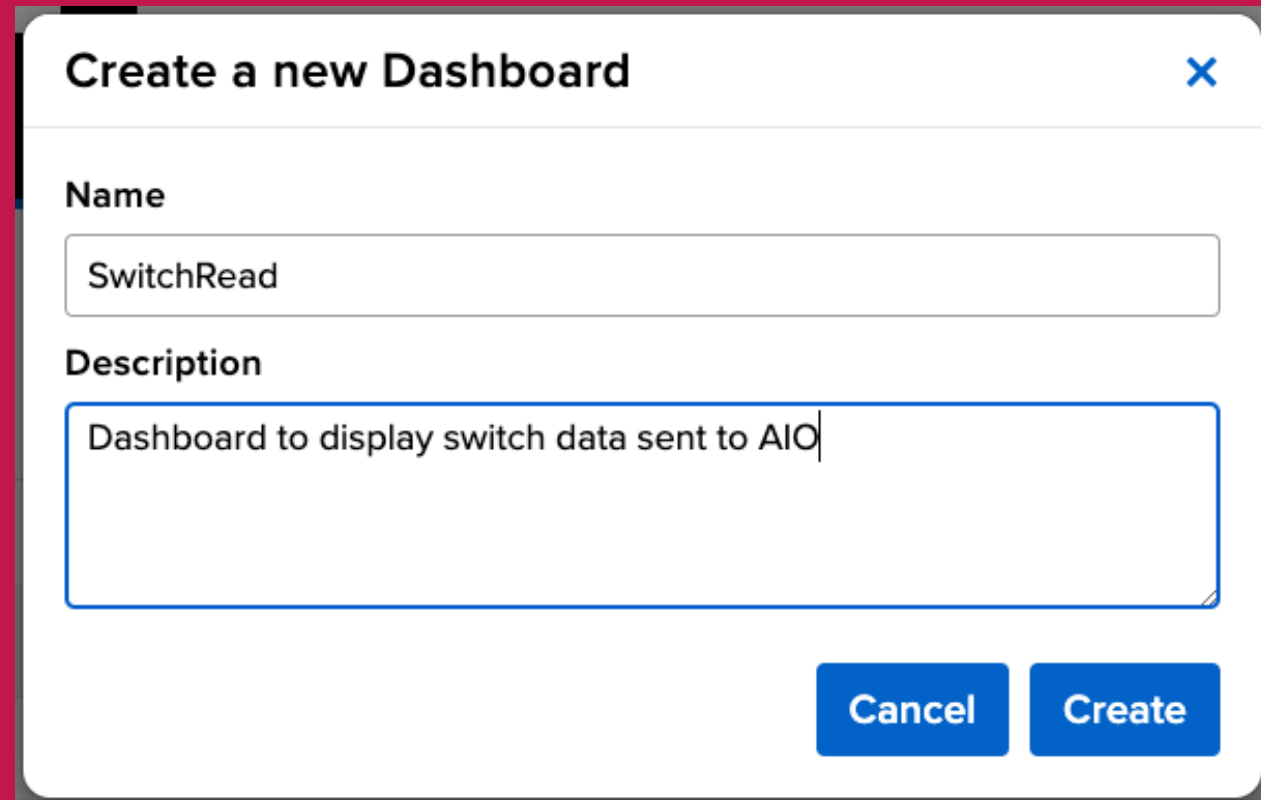
**Description**

# Create AIO Dashboard



The screenshot shows the Adafruit web interface with the 'Dashboards' menu selected. A blue button labeled '+ New Dashboard' is visible. Below it, a table lists existing dashboards:

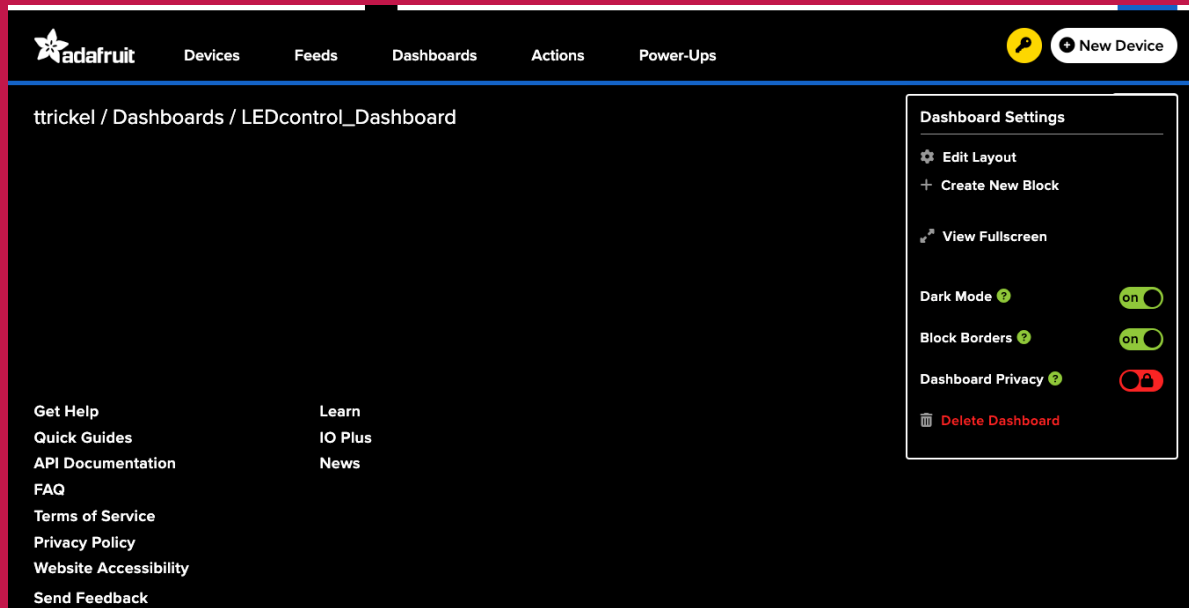
<input type="checkbox"/> Name	Key	
<input type="checkbox"/> CO2	co2	
<input type="checkbox"/> ControlLED	controlled	
<input type="checkbox"/> LightSensor	lightsensor	
<input type="checkbox"/> Test	test	



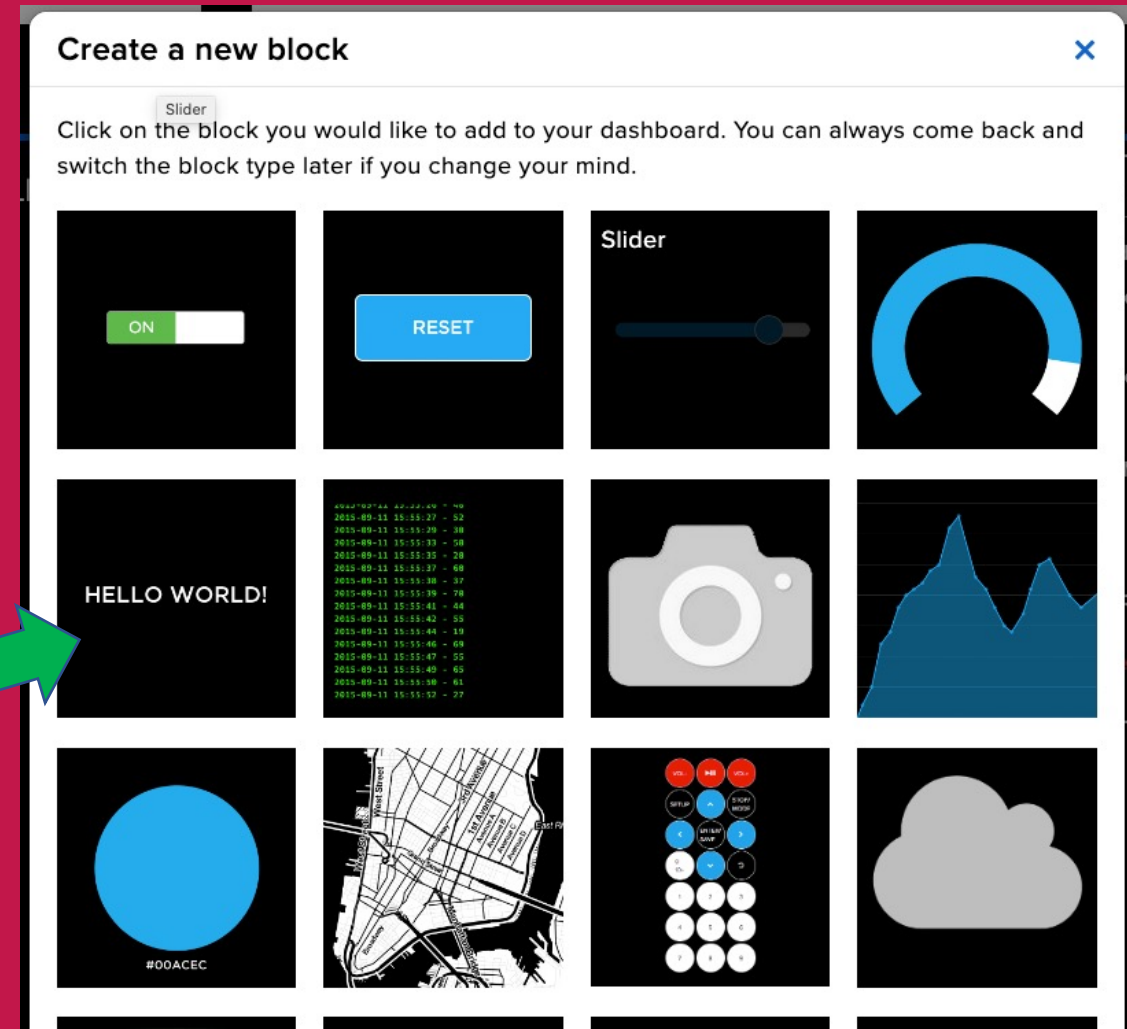
The 'Create a new Dashboard' modal form contains the following fields and buttons:

- Name:** A text input field containing the text 'SwitchRead'.
- Description:** A text area containing the text 'Dashboard to display switch data sent to AIO'.
- Buttons:** 'Cancel' and 'Create' buttons at the bottom right.

# Create AIO Dashboard continued



The screenshot shows the Adafruit dashboard settings page. The top navigation bar includes the Adafruit logo, links for Devices, Feeds, Dashboards, Actions, and Power-Ups, and a 'New Device' button. The main content area is titled 'ttrickel / Dashboards / LEDcontrol\_Dashboard'. On the right, the 'Dashboard Settings' panel includes options for 'Edit Layout', 'Create New Block', and 'View Fullscreen'. Below these are three toggle switches: 'Dark Mode' (on), 'Block Borders' (on), and 'Dashboard Privacy' (off). A 'Delete Dashboard' button is at the bottom of the settings panel. On the left, there is a sidebar with links for 'Get Help', 'Quick Guides', 'API Documentation', 'FAQ', 'Terms of Service', 'Privacy Policy', 'Website Accessibility', and 'Send Feedback'. A second sidebar contains links for 'Learn', 'IO Plus', and 'News'.



The screenshot shows the 'Create a new block' dialog box. It features a title bar with a close button (X) and a 'Slider' tab. The main text reads: 'Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.' Below the text is a grid of block preview cards. A green arrow points from the 'HELLO WORLD!' block in the second row, first column to the 'Create a new block' dialog box. The grid includes various block types such as a toggle switch, a 'RESET' button, a slider, a circular progress indicator, a text block with 'HELLO WORLD!', a terminal window with green text on a black background, a camera icon, a line graph, a blue circle with the hashtag '#00ACEC', a map, a numeric keypad, and a cloud icon.

# Create AIO Dashboard continued

## Connect a Feed ✕

A text block can be used to send data as well as view data. To publish, click on the text block, enter any text, and press enter to send.

Choose a single feed you would like to connect to this text. You can also create a new feed within a group.

Search for a feed

Default			
Feed Name	Last value	Recorded	
<input type="checkbox"/> CO2	923	7 months	🔒
<input type="checkbox"/> ControlLED	ON	2 days	🔒
<input type="checkbox"/> humidityReading	716.827392578...	7 months	🔒
<input type="checkbox"/> ledControl	off	5 days	🔒
<input type="checkbox"/> LEDcontrol_Feed	OFF	about 2 hours	🔒
<input type="checkbox"/> lightReading	0	over 4 years	🔒
<input type="checkbox"/> runTime	9:6:0	over 4 years	🔒
<input checked="" type="checkbox"/> SwitchRead_Feed		3 minutes	🔒
<input type="checkbox"/> switchTest	0	5 days	🔒
<input type="checkbox"/> temperatureReading	698.892364501...	7 months	🔒

1 of 1 feeds selected

< Previous step Next step >

## Block settings ✕

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

Block Preview

Font Size

**Static Text**

When checked, ignore feed value and show the selected 'Static Text Value' all the time.

Static Text Value

When 'Static Text' is checked, use this value. Limited to 256 characters.

Decimal Places

Number of decimal places to display when value is a number. Defaults to -1 (unlimited).

**Show Icon**

When checked, show an icon with the value.

Icon

Show this icon next to the value.

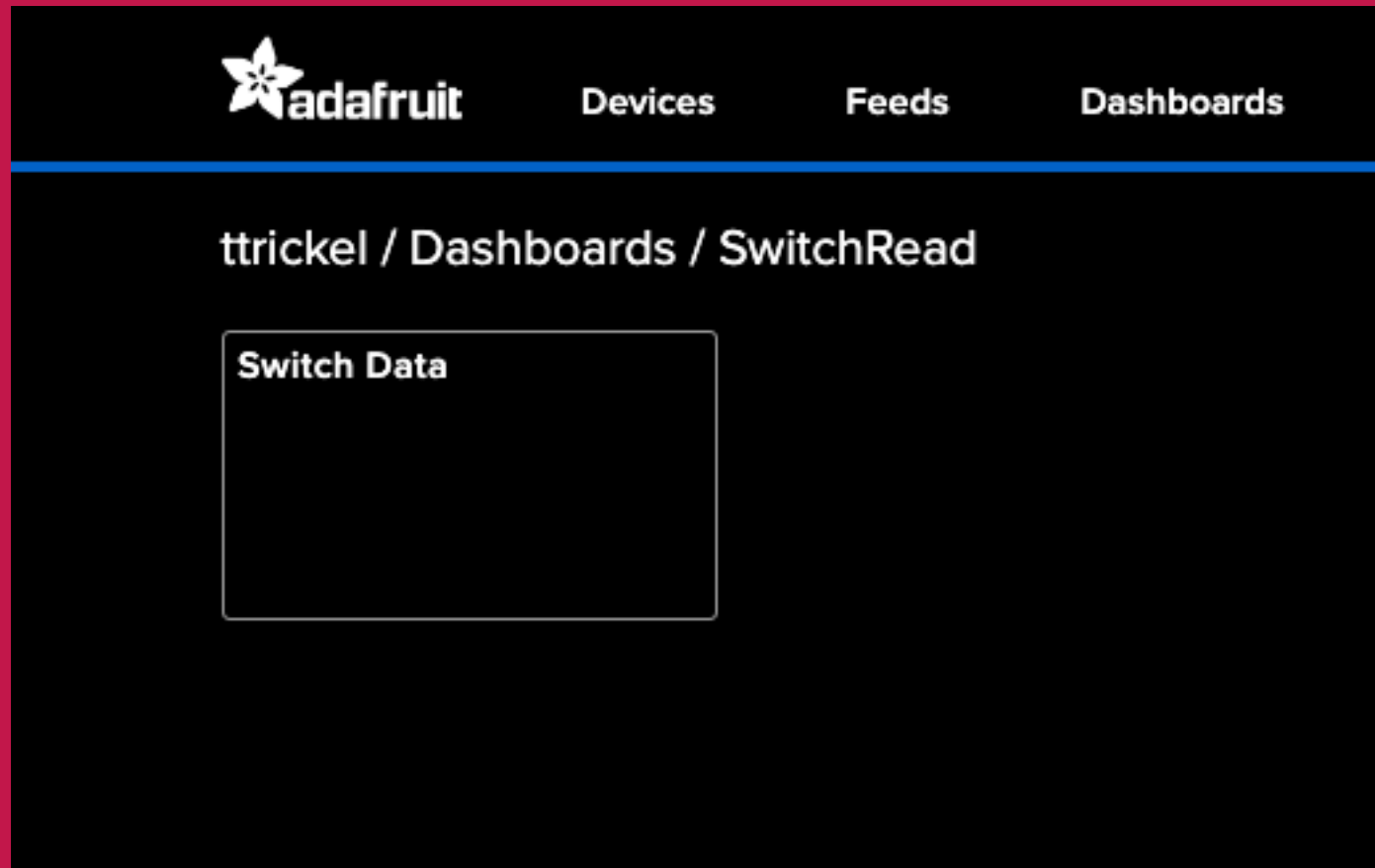
Text A text block can be used to send data as well as view data. To publish, click on the text block, enter any text, and press enter to send.

Test Value

Published Value

< Previous step Create block

# Create AIO Dashboard continued



# Send Switch Data to AIO Python Code

change

YOUR\_AIO\_KEY

YOUR\_AIO\_USERNAME



```
1  """
2  'digitalIn.py'
3  =====
4  Example of sending the status of a
5  switch to the Adafruit IO Python Client
6  """
7  # Import standard python modules
8  import time
9  import RPi.GPIO as GPIO
10
11 # import Adafruit IO REST client.
12 from Adafruit_IO import Client, Feed, RequestError
13
14 # setup GPIO
15 GPIO.setwarnings(False)
16 GPIO.setmode(GPIO.BCM)
17 GPIO.setup(25, GPIO.IN)
18
19 # Set to your Adafruit IO key.
20 # Remember, your key is a secret,
21 # so make sure not to publish it when you publish this code!
22 ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'
23
24 # Set to your Adafruit IO username.
25 # (go to https://accounts.adafruit.com to find your username)
26 ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'
27
28 # Create an instance of the REST client.
29 aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
30
31 try: # if we have a 'digital' feed
32     digital = aio.feeds('switchread-feed')
33 except RequestError: # create a digital feed
34     feed = Feed(name="switchread-feed")
35     digital = aio.create_feed(feed)
36
37 while True:
38     if GPIO.input(25):
39         aio.send(digital.key, 0)
40     else:
41         aio.send(digital.key, 1)
42
43 # timeout so we dont flood adafruit-io with requests
44 time.sleep(0.5)
45
```

# Test It

## Run Python Code



## Press Switch on Breadboard

