CSCi 4907
Introduction to IoT and Edge Computing Applications

Prof. Kartik Bulusu, CS Dept.

Week 4 [02/16/2024]	• Guest lecture: **Anatomy of a real life IoT attack - Side Channel Attack (SCA) Analysis** by Chowdary Yanamadala, Technology Strategist ARM • Discussion on the midterm projects for updates and reporting	• dweet.io – Twitter of things • Fetching data from dweet • dweet an LED - Your first IoT program Raspberry Pi messaging with Python scripts • HW 3 assigned

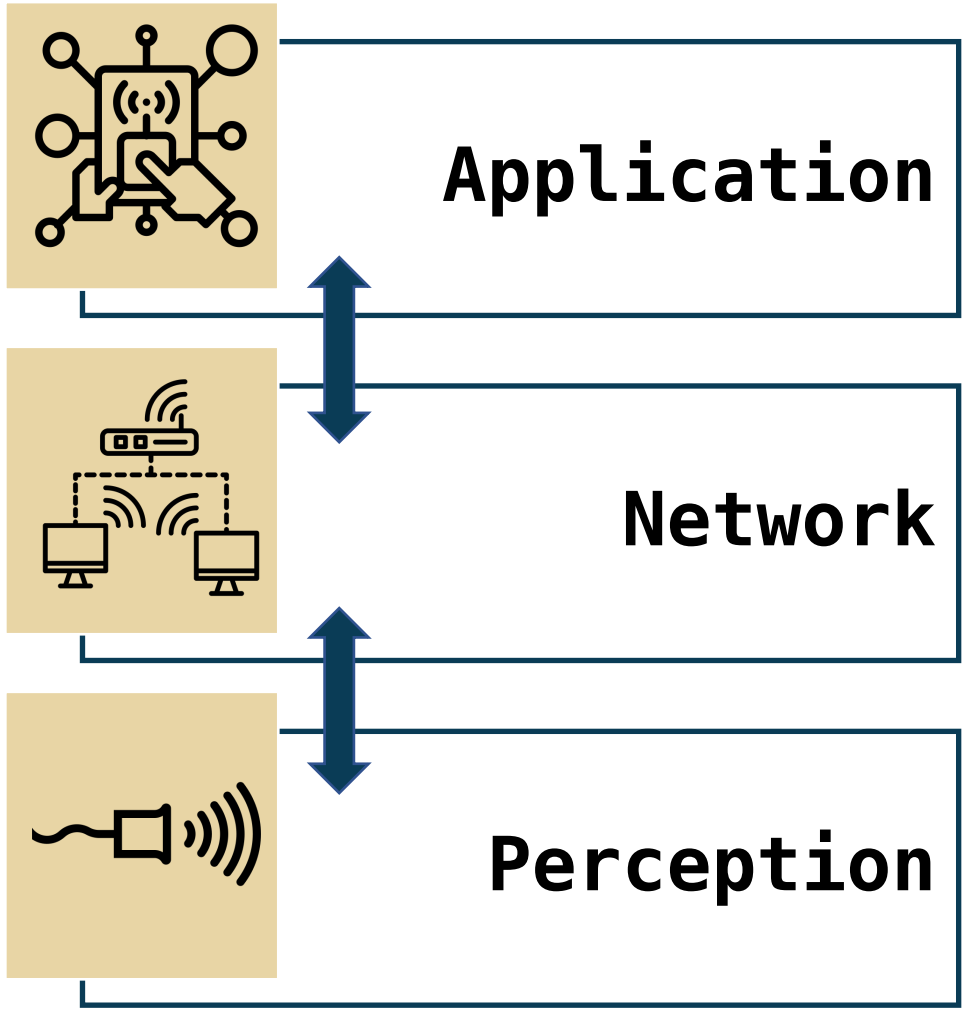
```
git clone git@github.com:gwu-csci3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

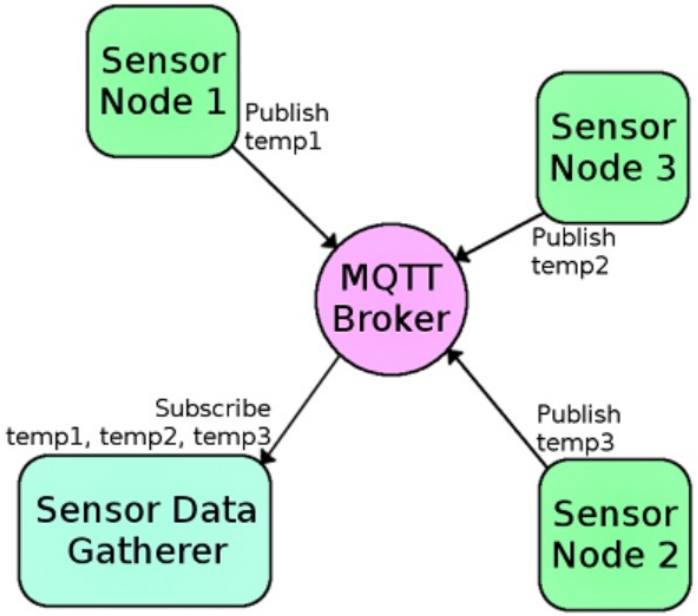
Spring 2024

Photo: Kartik Bulusu

Recap: The 3-Layer IoT Architecture



MQTT paradigm



Hardware

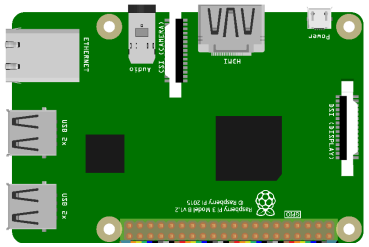
Broker

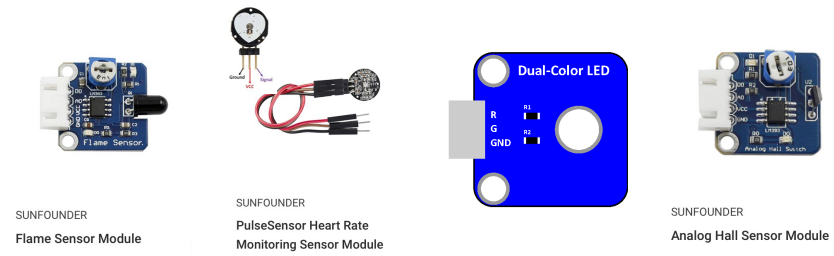
- The broker is the server
- It distributes the information to the interested devices connected to the server.

Client

- The device that connects to broker to send or receive information.

Source: https://learn.sparkfun.com/tutorials/introduction-to-mqtt#the-basics





SUNFOUNDER
Flame Sensor Module

SUNFOUNDER
PulseSensor Heart Rate Monitoring Sensor Module

SUNFOUNDER
Analog Hall Sensor Module

Messaging

Topic

- The name that the message is about.
- Clients publish, subscribe, or do both to a topic.

Subscribe

- Clients tell the broker which topic(s) they're interested in.

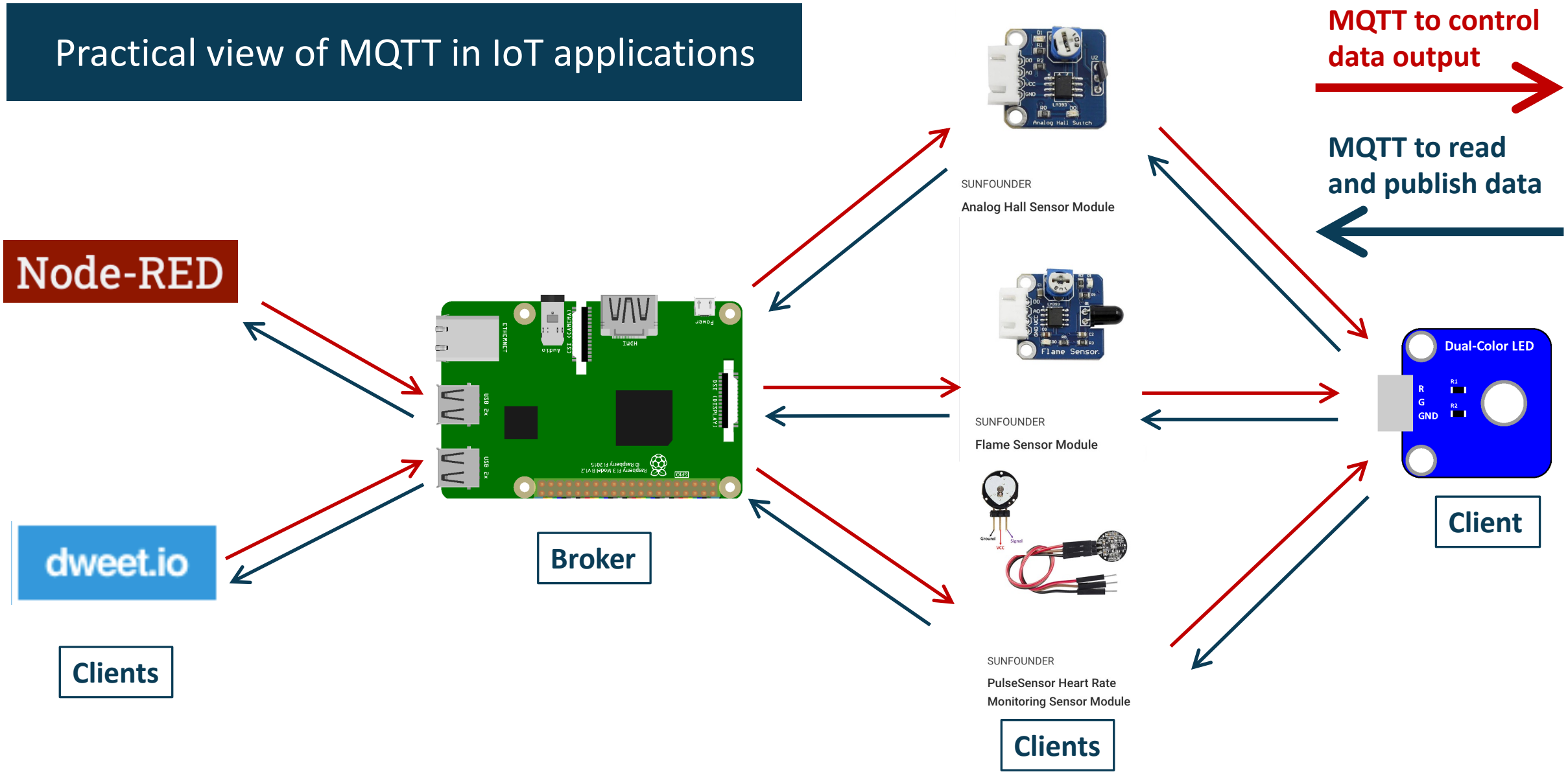
Publish

- Clients that send information to the broker to distribute to interested clients based on the topic name.

QoS

- Quality of Service to the broker
- Integer value ranging from. 0-2.

Practical view of MQTT in IoT applications



dweet.io

A simple publishing and subscribing tool for IoT devices ..

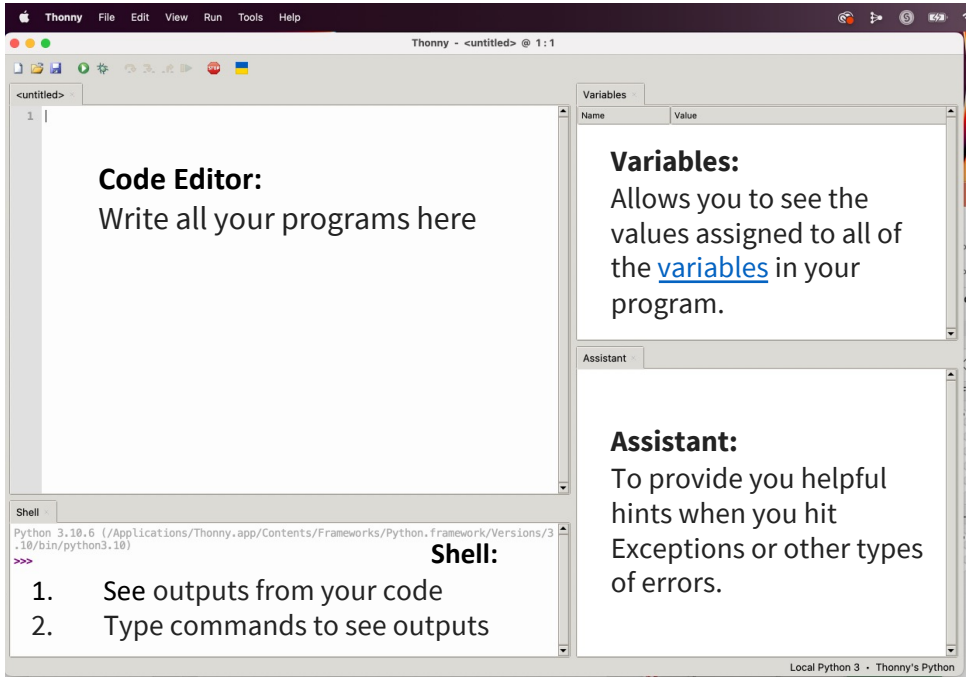
or anybody's thing

It's like Twitter of things!

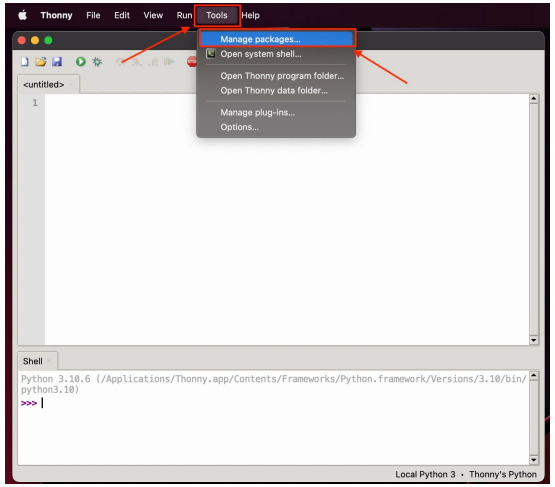
Goal-1:
Fetch dweet-data using Python

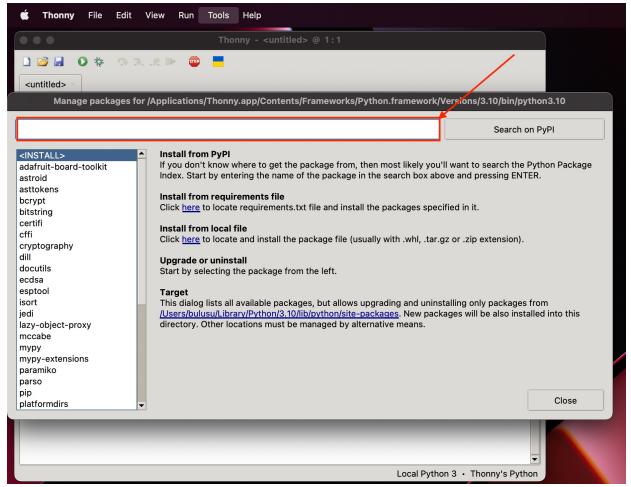
Step-1:
Install Thonny IDE

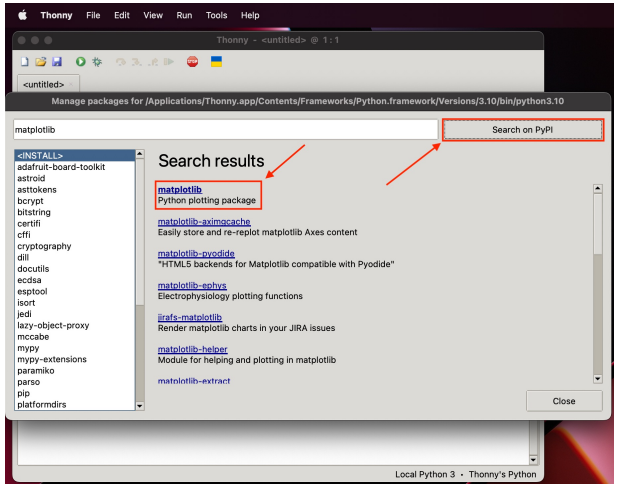
https://thonny.org



Step-2
Install the following libraries:
1. matplotlib
2. pandas
3. numpy
4. dweepy





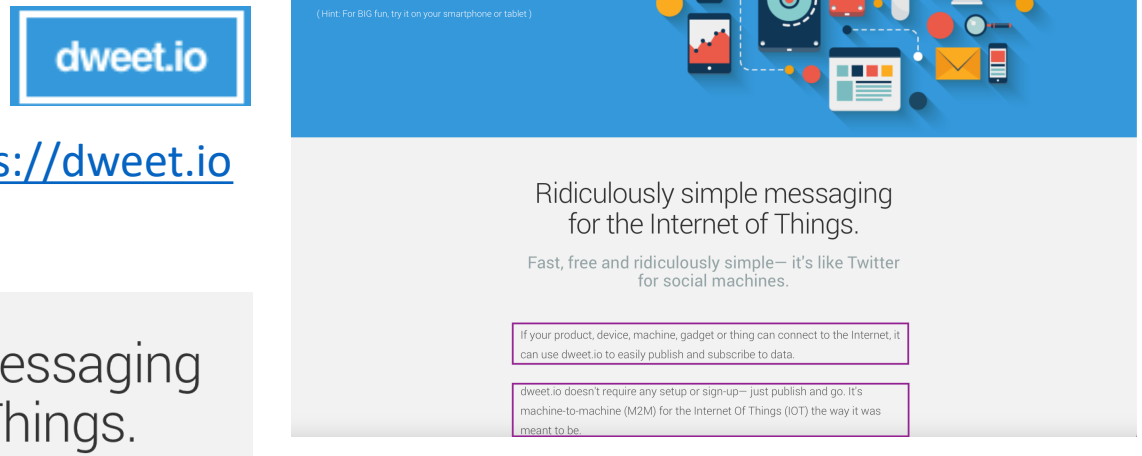


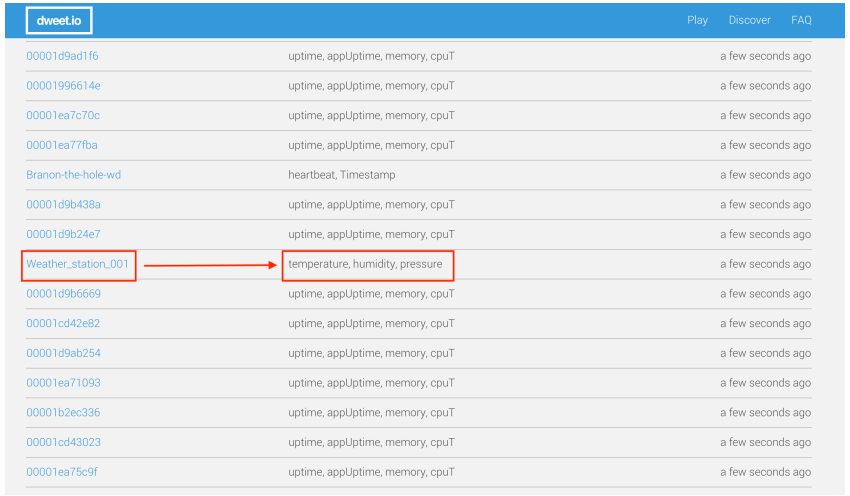
School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

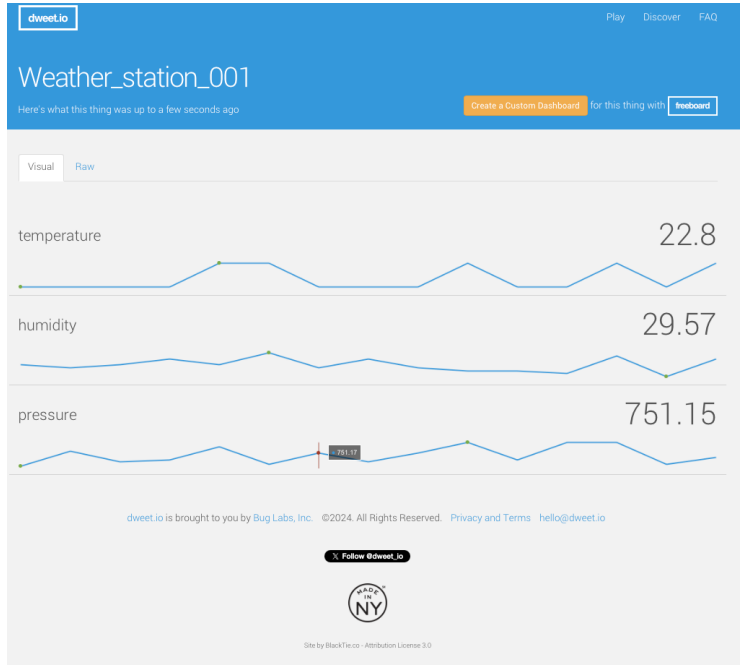
GW

Step-3:
Go to dweet.io

https://dweet.io







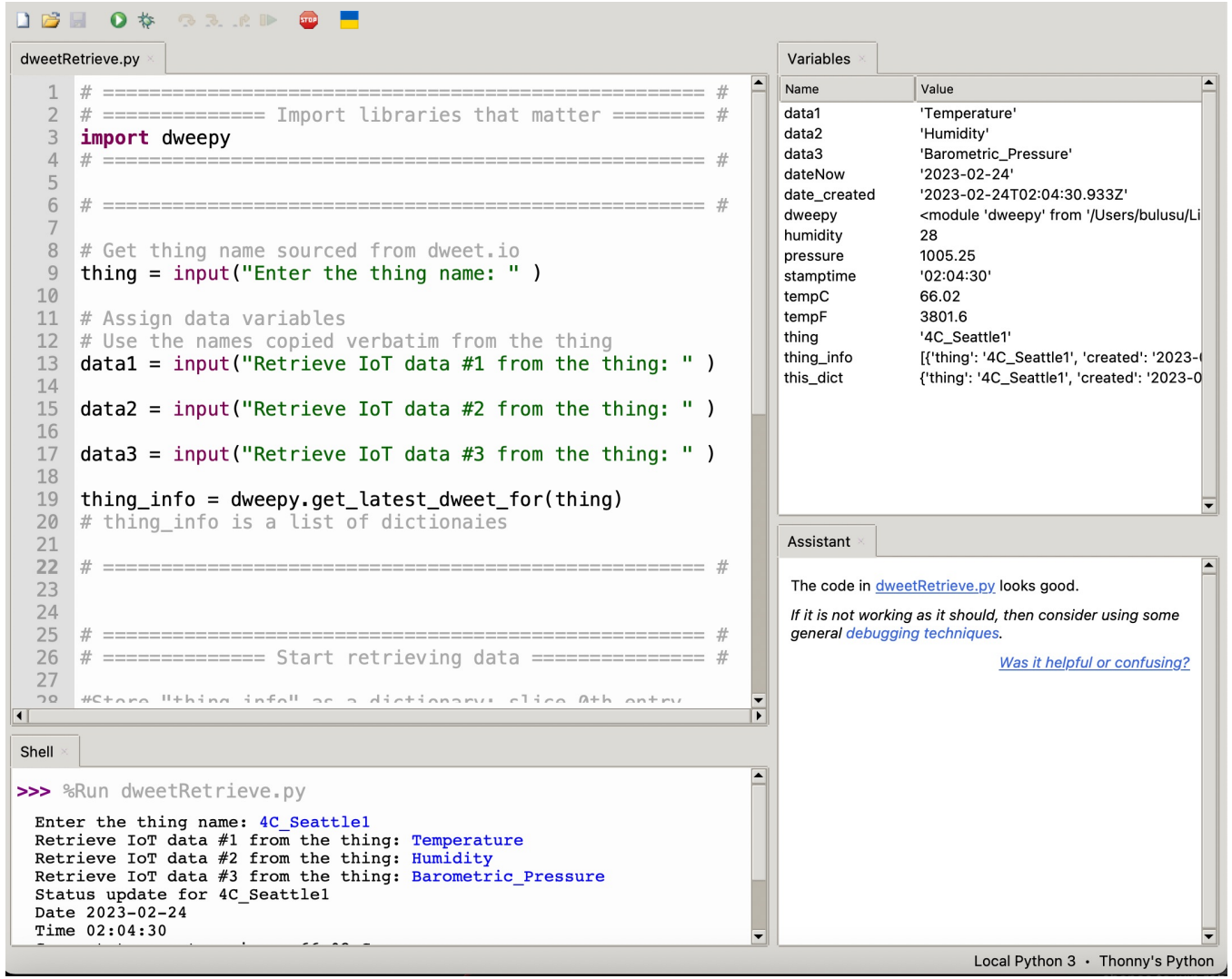
Step-4:

Let's write a Python program to retrieve dweet-data from a thing

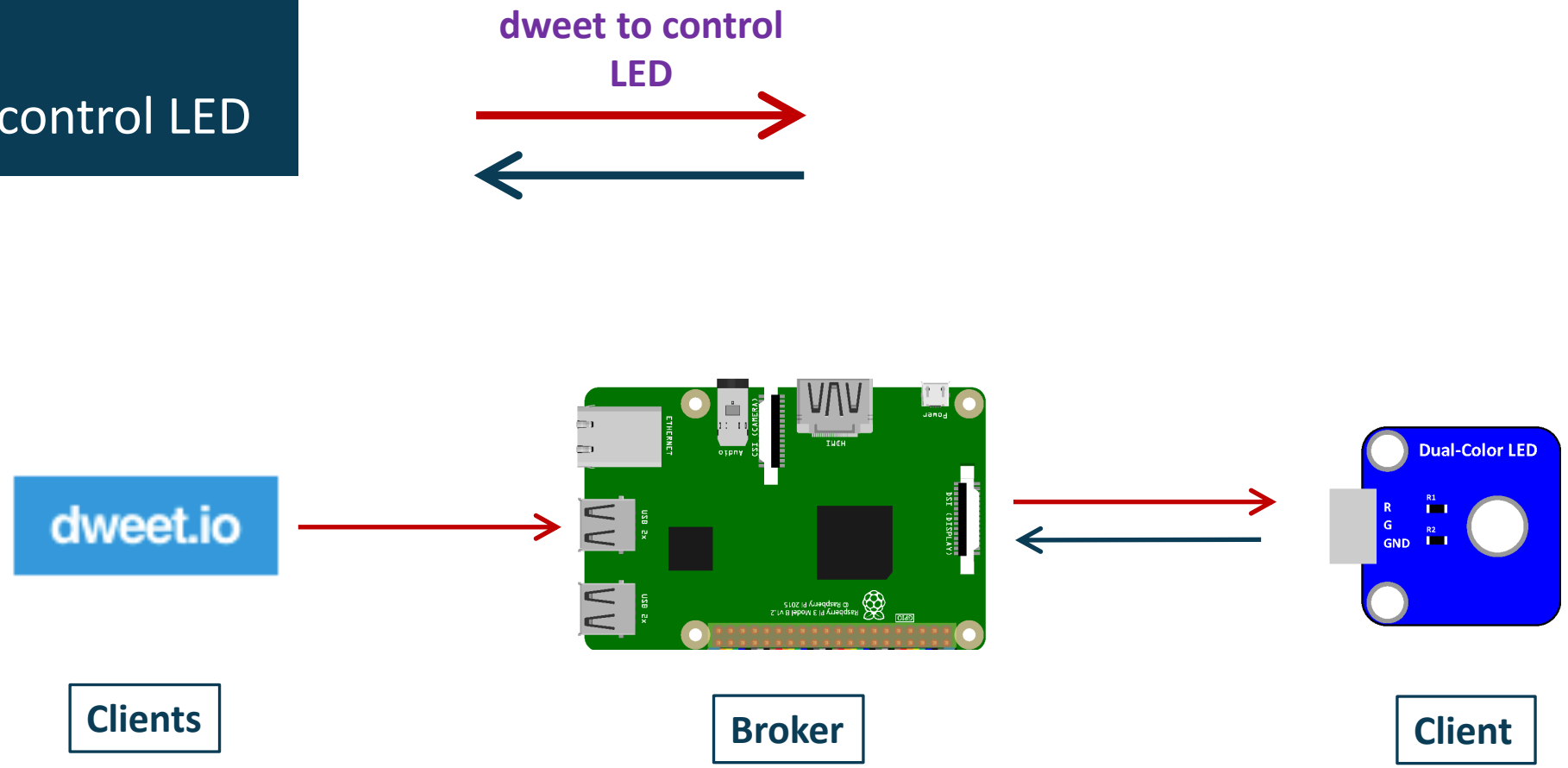
Test drive:

http://www.dweet.io/follow/Weather_station_001

dweet-thing: Weather_station_001
data1: temperature
data2: humidity
data3: pressure



Goal-2:
dweet to control LED



Let's get started

Preliminaries:

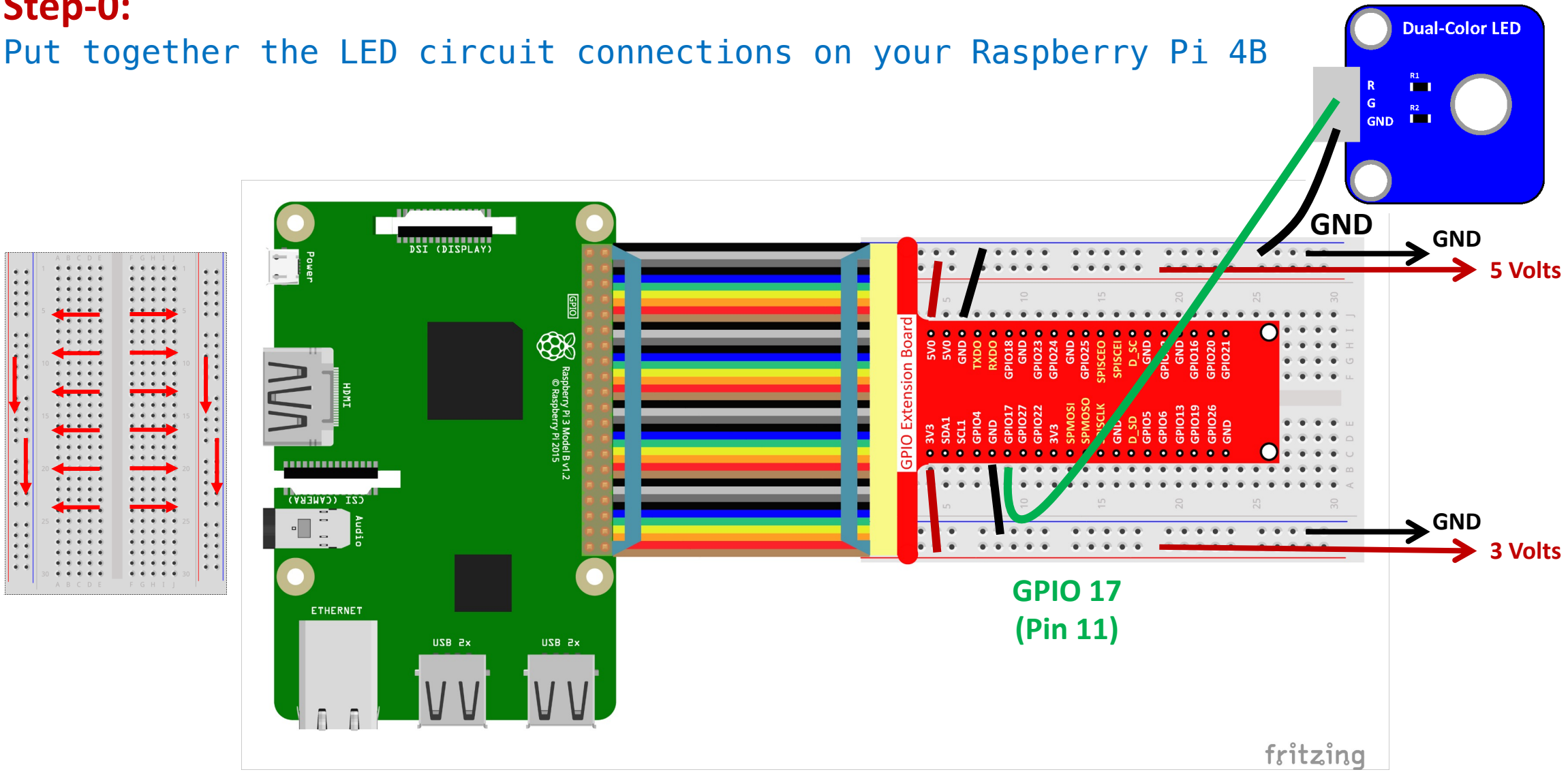
1. git-clone

```
git clone git@github.com:gwu-i3907/Spring2024.git

git clone https://github.com/gwu-csci3907/Spring2024.git
```

2. I have the files on a USB in case there is a slowdown!

Step-0:

Put together the LED circuit connections on your Raspberry Pi 4B



Run
`dweet_LED_Example`

Your first IoT program

Graded Lab Activity (10 points)

Step-1:

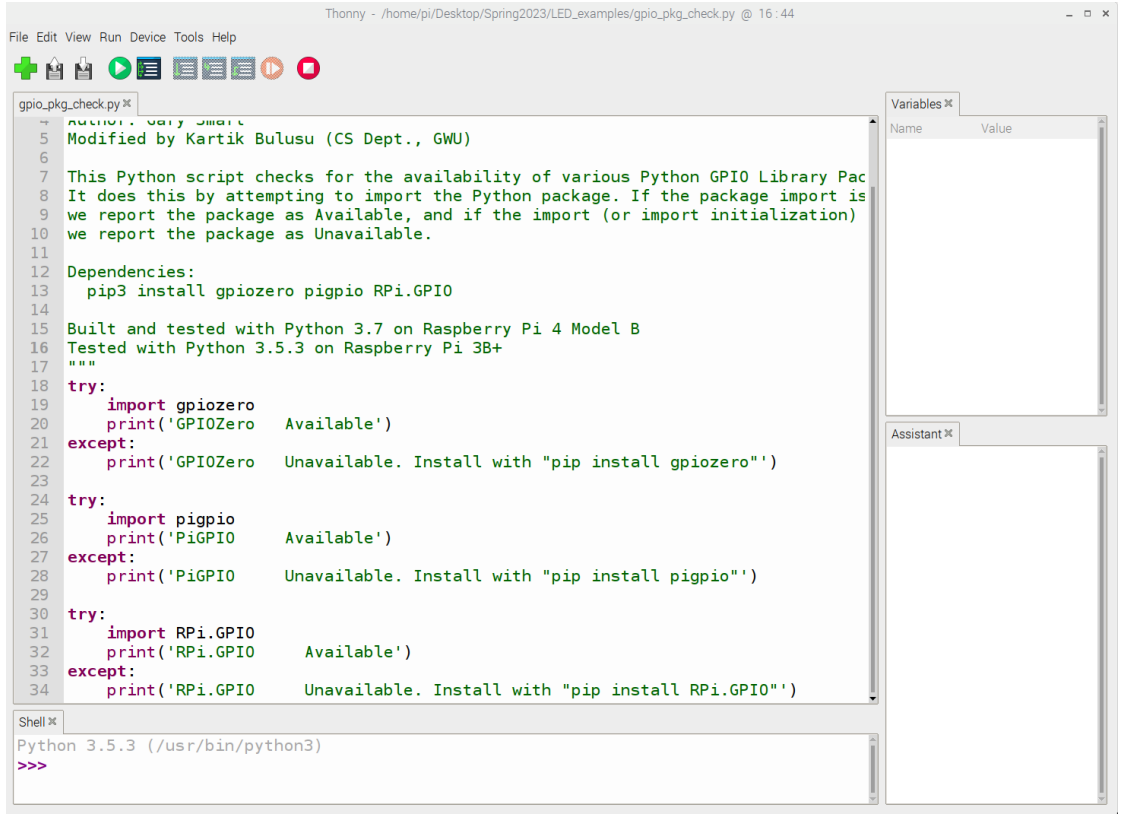
- Create `python` virtual environment
  - Activate `python-venv`
  - Install `pip`

```
pi@raspberrypi:~$ cd Desktop/Spring2024/dweet_LED_Example/

pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $

pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ python -m venv venv

pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ source venv/bin/activate

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip install --upgrade pip

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip list
```

Step-2:

- Review of the codes provided to you
- Execute `gpio_pkg_check.py` from the terminal
- See example screenshot for Thonny IDE

Note: You are still in `python-venv`



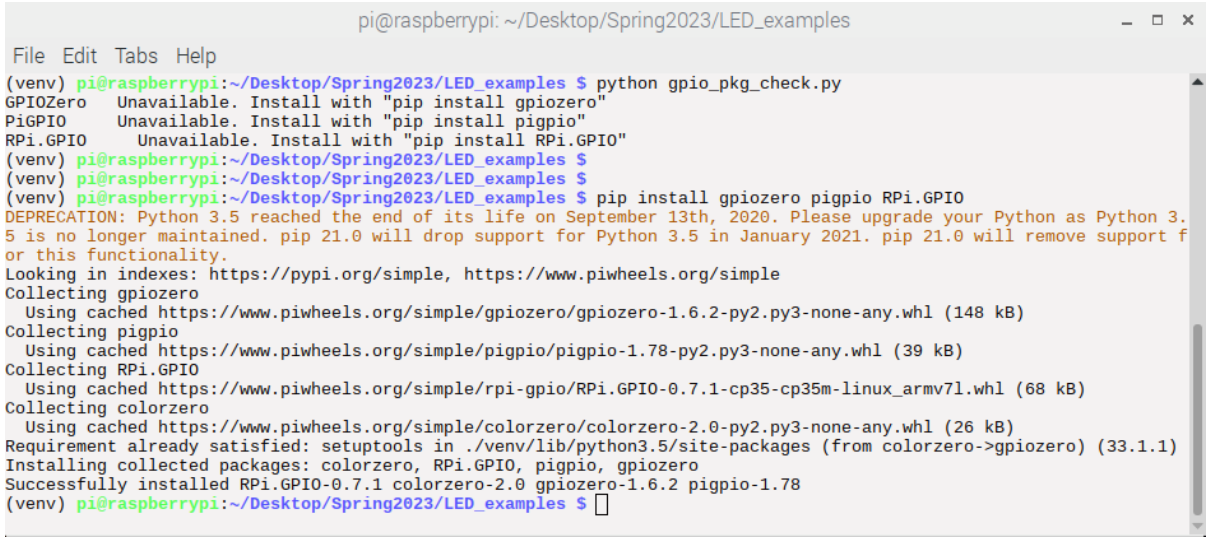

```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ python gpio_pkg_check.py
```

Step-3:

Install `GPIO & requests` packages

Note:

- You are still in `python-venv`
- Installation of all the packages may take a while
- You may need to repeat this step again.
  - Follow the subsequent steps to complete the installations




```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip install gpiozero pigpio
RPi.GPIO numpy matplotlib requests yagmail
```

Step-4:

Install system level dependencies needed to work with the following packages:

- `matplotlib`
- `numpy`
- `yagmail`

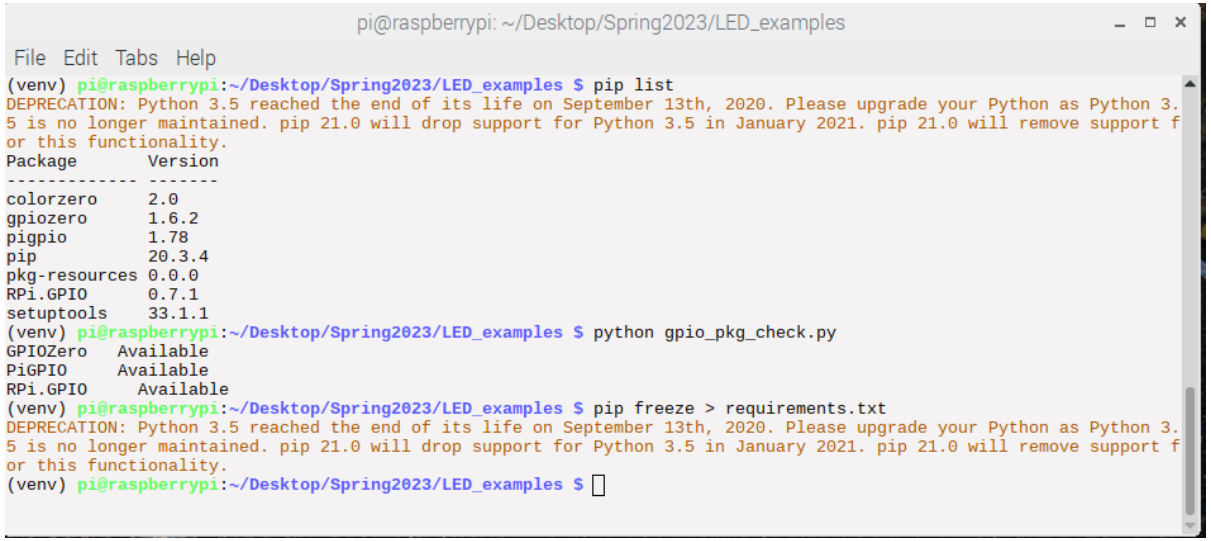
Note:

- These updates may take a while to complete
- Repeat installation `GPIO & requests` packages as suggested in Step-2 to complete all installations

```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo apt-get install libopenblas-dev

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo apt-get install libxml2-dev libxslt-dev

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip install gpiozero pigpio RPi.GPIO numpy matplotlib requests yagmail
```

Step-5:

- Execute `gpio_pkg_check.py` from the terminal
- Perform some checks and balances
  - `pip list`
  - `pip freeze`
  - See example screenshot for what to expect




```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ python gpio_pkg_check.py

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip list

(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ pip freeze > requirements.txt
```

Step-6:

Initialize `pigpio` daemon

```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo pigpiod
```

Alternatively:

Initialize `pigpio` daemon using the following terminal commands

```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo systemctl enable pigpiod
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo systemctl start pigpiod
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo systemctl stop pigpiod
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ sudo systemctl start pigpiod
```

Step-7:

Run the `python` codes related to `dweet` following the next set of steps

Breakdown of your first IoT program

Skeleton of the updated Python program written for Raspberry Pi

```
import library1 as name1
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(12, GPIO.OUT)

def function_name(arguments):
        statement
        statement
        . . .
        return value

if __name__ == "__main__":
        try:
                function_name1()
        except KeyboardInterrupt:
                function_name2()
```

Import libraries that are relevant for interaction with the Raspberry Pi hardware such as GPIO pins, camera ports etc.

Set up GPIO pins as data outputs or inputs for sensors and actuators

Create user-defined functions to modularize your code and make it easy to work.

Create user-defined functions to release the GPIO pins

Create entry point into the program and pull all functions in

Create a keyboard-interrupt exception clause

`signal module,` facilitates means through which a program can receive information (events from the Operating System and pass them to programs.

- For example, signal `SIGINT` is generated when we press the keystrokes Ctrl + C on our keyboard

```
import signal
import json
import os
import sys
import logging
from gpiozero import Device, LED
from gpiozero.pins.pigpio import PiGPIOFactory
from time import sleep
from uuid import uuid1
import requests
import RPi.GPIO as GPIO
```

`logging module`, tracks (disruptive) events that occur in a computer system, such as problems, errors or just information on current operations

`json` (JavaScript Object Notation) `module` used for storing and transporting data from a server to a web page.

`os module` in Python provides functions for interacting with the operating system such as

- Handling and creating directories
- Listing out Files and Directories
- Deleting Directory or Files using Python

`requests module` is used making HTTP requests to a specified URL by sending and receiving data from websites by providing a uniform interface for both `get` and `post` methods.

For example,

- `get method` is used to retrieve information from the given server using a given URL
- `post method` sends the encoded user information appended to the page request.
- The page and the encoded information are separated by the '?' character.

`uuid` (Universally Unique Identifier) `module` in Python, is a 128-character string of alphanumeric variable type, that uniquely identifies an object, entity, or resource in both space and time of a table.

- For example, `uuid.uuid1()` creates a UUID by utilizing the computer's MAC address and the current time in accordance with the RFC 4122 definition.

Sources:
https://docs.python.org/3/library/logging.html
https://docs.python.org/3/library/uuid.html
https://favtutor.com/blogs/uuid-python
https://www.geeksforgeeks.org/python-requests-tutorial/

Review your LED connection to make sure you are connected to GPIO17. Got back to Step-0 to confirm.

```
# Global Variables
LED_GPIO_PIN = 17                 # GPIO Pin that LED is connected to
THING_NAME_FILE = 'thing_name.txt'  # The name of our "thing" is persisted into this file
URL = 'https://dweet.io'          # Dweet.io service API
last_led_state = None             # Current state of LED ("on", "off", "blinking")
thing_name = None                 # Thing name (as persisted in THING_NAME_FILE)
led = None                        # GPIOZero LED instance


# Initialize Logging
logging.basicConfig(level=logging.WARNING)  # Global logging configuration
logger = logging.getLogger('main')  # Logger for this module
logger.setLevel(logging.INFO)  # Debugging for this file.                    # (2)


# Initialize GPIO
Device.pin_factory = PiGPIOFactory()
```

`logging module`, tracks (disruptive) events that occur in a computer system, such as problems, errors or just information on current operations

`gpiozero module` is a simple interface to GPIO devices with Raspberry Pi, developed and maintained by Ben Nuttalland Dave Jones.

- It is a simpler alternative to `RPi.GPIO module`
- `Documentation:` https://gpiozero.readthedocs.io/en/latest/

```
# Function Definitions
def init_led():
    """Create and initialise an LED Object"""
    global led
    led = LED(LED_GPIO_PIN)
    led.off()


def resolve_thing_name(thing_file):
    """Get existing, or create a new thing name"""
    if os.path.exists(thing_file):
        with open(thing_file, 'r') as file_handle:
            name = file_handle.read()
            logger.info('Thing name ' + name + ' loaded from ' + thing_file)
            return name.strip()
    else:
        name = str(uuid1())[:8]  # UUID object to string.
        logger.info('Created new thing name ' + name)

        with open(thing_file, 'w') as f:
            f.write(name)

    return name
```

`gpiozero module`, is used to initialize the GPIO pin where the LED is connected.

(3)

`uuid` (Universally Unique Identifier) `module` in Python creates your unique thing-name:

- `uuid.uuid1()` creates a UUID by utilizing the computer's MAC address and the current time in accordance with the RFC 4122 definition.

(4)

(5)

`logging module`, tracks (disruptive) events that occur in a computer system:

- `logging.info()` records information on a piece of code that works as intended.

```python
def get_latest_dweet():
    """Get the last dweet made by our thing."""
    resource = URL + '/get/latest/dweet/for/' + thing_name
    logger.debug('Getting last dweet from url %s', resource)

    r = requests.get(resource)

    if r.status_code == 200:
        dweet = r.json()  # return a Python dict.
        logger.debug('Last dweet for thing was %s', dweet)

        dweet_content = None

        if dweet['this'] == 'succeeded':
            # We're just interested in the dweet content property.
            dweet_content = dweet['with'][0]['content']

        return dweet_content

    else:
        logger.error('Getting last dweet failed with http status %s', r.status_code)
        return {}
```

`requests module` is utilized here to retrieve the URL

- `get method` is used to retrieve information from the given server using a given URL

(6)

`logging module`, tracks (disruptive) events that occur in a computer system:

- `logging.debug()`, provides a flexible framework for emitting log messages from Python programs

(7)

(8)

`json` (JavaScript Object Notation) `module` used for converting data from a server to json data structure that is analogous to a dictionary in Python.

(9)

`'this':value` is one of key-value pairs in the `dweet json object`

(10)

`logging module`, tracks (disruptive) events that occur in a computer system:

- `logging.error()`, provides a will give you that nicely formatted ERROR message in addition to the Traceback and Stack.

```
def process_dweet(dweet):
    """Inspect the dweet and set LED state accordingly"""
    global last_led_state

    if not 'state' in dweet:
        return

    led_state = dweet['state']

    if led_state == last_led_state:
        return  # LED is already in requested state.

    if led_state == 'on':
        led.on()
    elif led_state == 'blink':
        led.blink()
    else:  # Off, including any unhandled state.
        led_state = 'off'
        led.off()

    if led_state != last_led_state:
        last_led_state = led_state
        logger.info('LED ' + led_state)
```

`json` (JavaScript Object Notation) `module` used for converting data from a server to json data structure that is analogous to a dictionary in Python.

- `'state':value` is one of key-value pairs in the `dweet json object`

`# (14)`

`# (15)`

`# (16)`

`dweet['state']` can be initialized to `'on'`, `'off'` or `'blink'`

- `'state':value` is one of key-value pairs in the `dweet json object`
- `gpiozero module` is used to initialize the state-value.

`logging module`, tracks (disruptive) events that occur in a computer system:

- `logging.info()` records if the state of the LED works as intended.

```
def poll_dweets_forever(delay_secs=2):
    """Poll dweet.io for dweets about our thing."""
    while True:
        dweet = get_latest_dweet()
        if dweet is not None:
            process_dweet(dweet)

        sleep(delay_secs)
```

When a `dweet` is available, from the `get_latest_dweet()` it is handled by `process_dweet()`.

We `sleep` for a default of 2 seconds before continuing with the while-loop

```
def print_instructions():
    """Print instructions to terminal."""
    print("LED Control URLs - Try them in your web browser:")
    print("  On    : " + URL + "/dweet/for/" + thing_name + "?state=on")
    print("  Off   : " + URL + "/dweet/for/" + thing_name + "?state=off")
    print("  Blink : " + URL + "/dweet/for/" + thing_name + "?state=blink\n")
```

`print_instructions()` is responsible for printing the `dweet` URLs to the terminal.

```
def signal_handler(sig, frame):
    """Release resources and clean up as needed."""
    print('You pressed Control+C')
    led.off()
    sys.exit(0)
```

`signal_handler()`, is responsible for turning off the LED.
`sys.exit(0)`, with argument-0 raises an exception i.e., "successful termination".

`resolve_thing_name()` is responsible for creating a unique dweet thing-name.

`init_led()` initializes the GPIO pin for the LED using `gpiozero module`

`signal module`, facilitates means through which a program can receive information (events from the Operating System and pass them to programs.

- For example, signal `SIGINT` is generated when we press the keystrokes Ctrl + C on our keyboard

```
# Initialise Module
thing_name = resolve_thing_name(THING_NAME_FILE)
init_led()


# Main entry point
if __name__ == '__main__':
    signal.signal(signal.SIGINT, signal_handler)  # Capture CTRL + C
    print_instructions()

    # Initialise LED from last dweet.
    last_dweet = get_latest_dweet()
    if (last_dweet):
        process_dweet(last_dweet)

    print('Waiting for dweets. Press Control+C to exit.')
    # Only use one of the following. See notes later in Chapter.
    # stream_dweets_forever() # Stream dweets real-time.
    poll_dweets_forever()  # Get dweets by polling a URL on a schedule.
```

(17)

(18) `print_instructions()` is responsible for printing the `dweet` URLs to the terminal.

(19) We start polling the dweet.io service to access the latest dweets using `poll_dweets_forever()`

Step-8:

- You should have completed your test runs
- You are still in `python-venv`
- `deactivate` your venv
- See example screenshot


```
(venv) pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $ deactivate

pi@raspberrypi:~$/Desktop/Spring2024/dweet_LED_Example $
```

Congratulations!
You just completed running you first IoT program!

Midterm projects
- open discussions on how to proceed