

Networking for Connected Devices

Computer Systems Assignment 2

Student: Noel McLoughlin 20082195

Revision: 1

Table of Contents

Project Description	1
Hardware and Software Design	2
Physical Hardware and Networking	3
Software Design	4
Redundant MQTT services	5
Redundant UI Dashboards	6
Blynk Dashboards	6
Integration with WIA	8
Send data to Thingspeak from Blynk.....	9
Thanks	10

Project Description

My chosen IoT Networking project is a reproducible and redundant **Weather Station** design.

Two nodes, A & B, each have sensor boards capable of reading weather data-

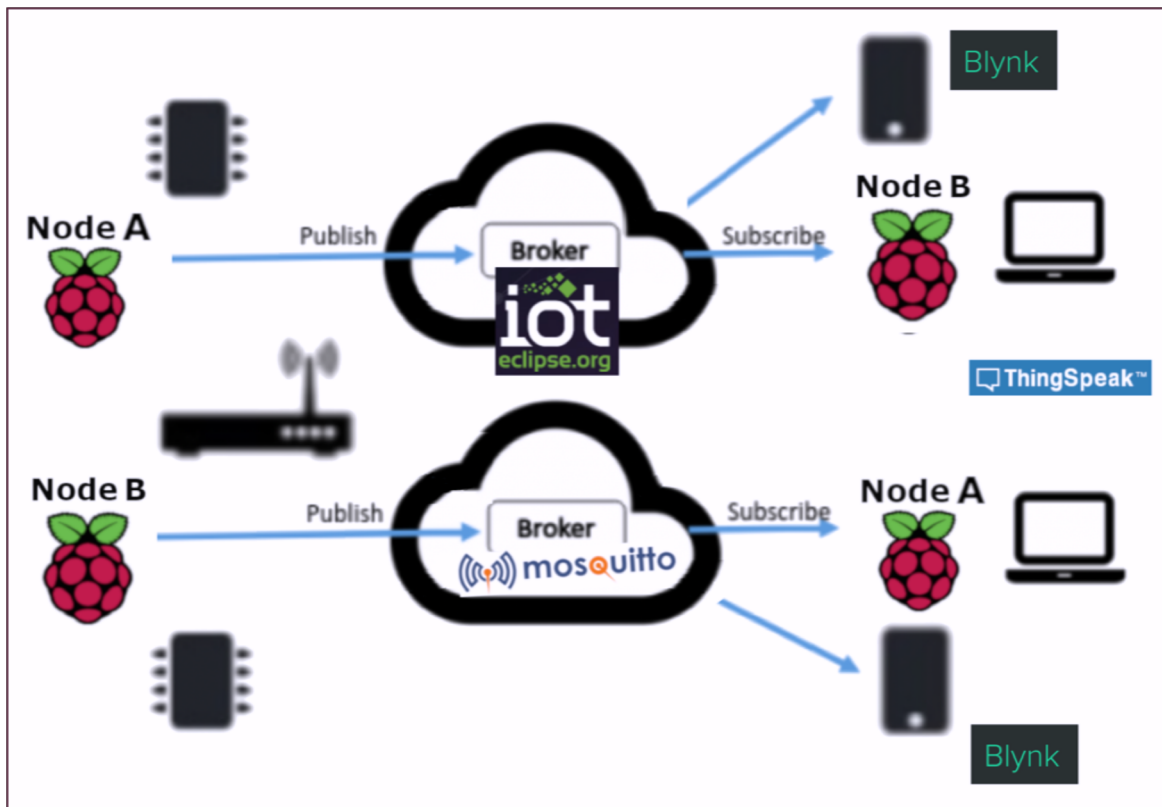
- Temperature
- Humidity
- Barometric pressure

The Internet of Things is an umbrella term to describe any environment with multiple devices (“things”) communicating over each other, and with third-party IoT cloud services. Data is collected by a two physical nodes and published as **MQTT** messages to separate MQTT brokers.

Such a redundant design requires two of ever thing where possible – two nodes, environmental boards, and MQTT brokers/publishers/subscribers.

Furthermore, the **Blynk** mobile application will be used to present two distinct dashboards for each Weather node.

The graphic show the project design and intention at macro level.



Hardware and Software Design

The design requires two weather station hosts to serve as compute, storage, and network hosts, and propagate power to connected devices.

- Raspberry Pi2B+ (2014)
- Raspberry Pi3B+ (2018)

Three sensor boards are connected to one or other of the Pi (only two are used).

- **Sense Hat Board** - an add-on board for Raspberry Pi, with 8x8 RGB LED matrix, five-button joystick, and environmental sensors for temperature, pressure, and humidity.
- **Sensor Node Board** – a development board. It mainly intended as a high quality environmental sensor for temperature, pressure, and humidity. Further features include TSL2591 (full/IR spectrum light sensor) and ESP-WROOM-32 (wifi and bluetooth) module.
- **BME680 Board** – an environmental sensor for humidity, barometric pressure, and VOC (volatile organic compounds) gas. It can communicate over I2C and SPI protocol.

The software required to support this project are the following-

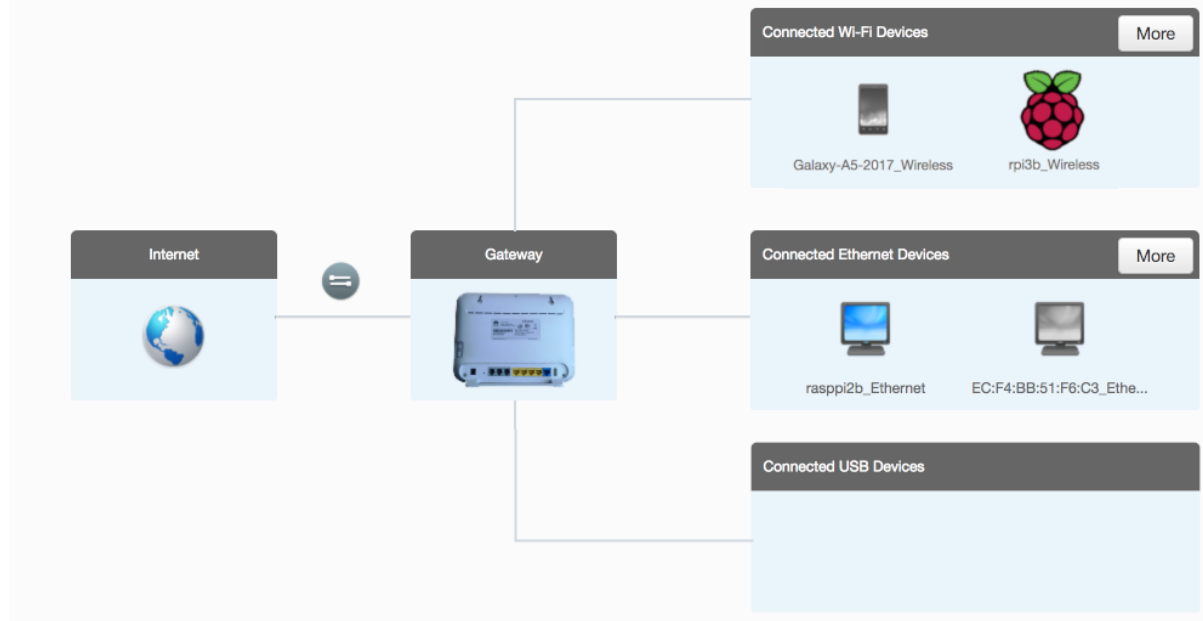
- **OS** – with Raspbian 32-bit OS.
- **Bluetooth** – A bunch of libraries (libudev, libdbus, etc) and the **bluez-5.50** package were installed to support Bluetooth and BLE. The Raspberry Pi 3B is not compatible with the latest bluez version (v5.9) – you must use the 5.50 version or the adapter is not detected.
- **Python** – Libraries required for this project are **paho-mqtt**, **sense-hat**, **bme680-python**, and **tinyDB**. Recommended libraries are- wia, gitpython, flask, and flask-cors.
- **Node** – NodeJS and npm highly recommended. There are node libraries available for some IoT platforms.
- **Code** – All code is saved on github.com. This project is designed to be reproducible.

Infrastructure requirements are minimal for non-redundant implementation-

- Two micro-usb power cables for each Raspberry Pi.
- Two power supplies
- Four jumper cables for the BME680 board.
- Internet Gateway supporting Ethernet and Bluetooth (802.11).

Physical Hardware and Networking

The diagram illustrates the network design with devices communicating over TCP/IP. The Gateway device (Huawei HG659) has an embedded DHCP service allocating IPv4 addresses on the 192.168.1.0/24 subnet range – IPv6 is not enabled in this setup.

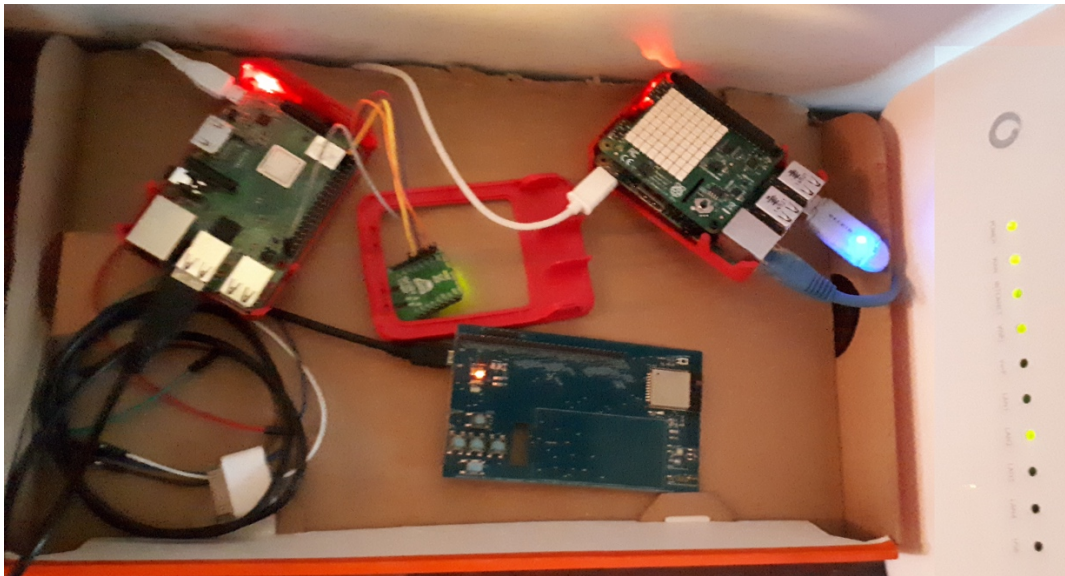


Node A

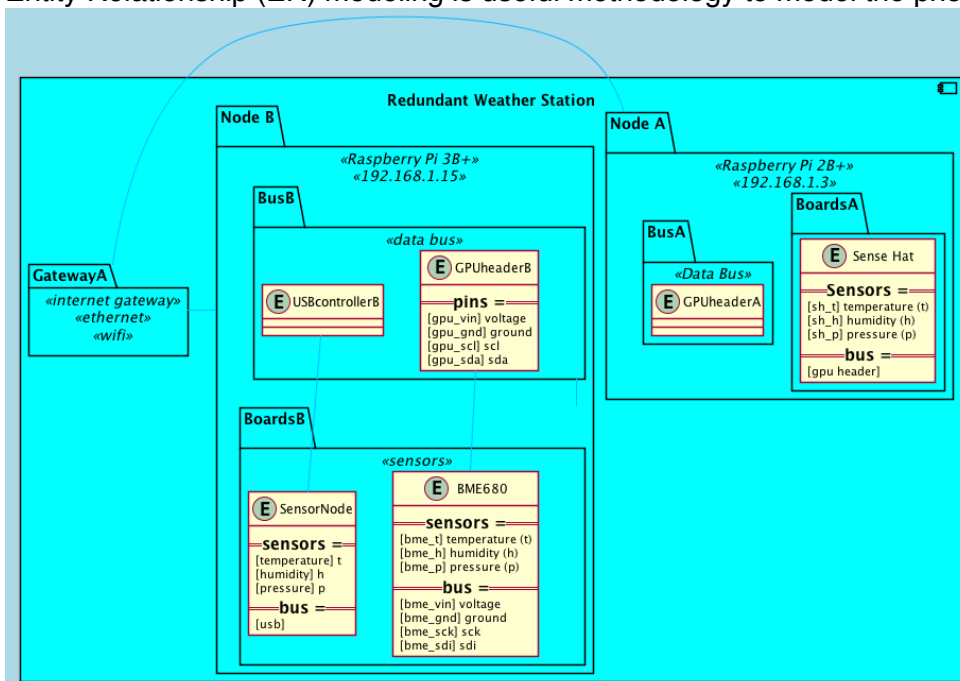
A Raspberry Pi2 connected to internet over Ethernet. Environmental data is taken by the Sense-Hat board connected to the GPU header. The “sense-hat” python library is used.

Node B

A Raspberry Pi3, using WiFi (802.11) for internet access. Two boards are wired - the SensorNode over USB (not used), and BME680 via GPU header (used). BME680 data is accessible using the “bme680-python” python library, looping over the “sensor.get_sensor_data()” library function.



Entity Relationship (ER) modeling is useful methodology to model the photo.



The Internet Gateway is a Home Gateway router supplied by Vodafone.

Software Design

Two important mechanisms for system intercommunication are HTTP (rfc2616) and MQTT (standard v3.1.1). protocols. These technologies are generic, stateless and well simple.

HTTP is the ubiquitous technology underpinning the Internet, with http messaging involving of requests from client to server and responses from server to client. MQTT is a Client Server publish/subscribe messaging transport protocol suitable for environments with constrained compute and network characteristics. Two MQTT brokers were used-

- `mqtt://iot.eclipse.org:1883`

- `mqtt://test.mosquitto.org:1883`

All code was written in Python. The following sections describe the software implementation.

Redundant MQTT services

My solution is published online (<https://github.com/noelmcloughlin/iot-edge-stepping-stones/tree/master/mqtt>) but reproduced here. All code is my own.

- Execute command to setup required software

```
./mqtt/mqtt.py -a install
```

The SenseHAT has temperature, pressure, and humidity sensors useful as Weather station. Let's start publishing data to a Cloud MQTT broker. The publisher defaults to 'sense_hat' board but 'bme680' is supported too.

- On hostA (default 'sense_hat' board), publish mqtt weather messages-

```
sudo ./mqtt.py -a publish -u mqtt://iot.eclipse.org:1883/NOELWEATHER_A
Connected to iot.eclipse.org:1883/NOELWEATHER_A Result: 0
{'h': 49.61815643310547, 't': 29.33, 'p': 1043.434326171875}
{'h': 49.139583587646484, 't': 29.33, 'p': 1043.46826171875}
```

- On hostB, subscribe to same mqtt weather messages-

```
./mqtt.py -a subscribe -u mqtt://iot.eclipse.org:1883/NOELWEATHER_A
Connected to iot.eclipse.org:1883/NOELWEATHER_A Result: 0
topic:NOELWEATHER_A/h, val:48.9634399414
topic:NOELWEATHER_A/t, val:29.42
topic:NOELWEATHER_A/p, val:1043.48852539
```

- On hostB, persist the same mqtt weather messages to TinyDB by setting flag-

```
sudo ./mqtt.py -a subscribe -u mqtt://iot.eclipse.org:1883/NOELWEATHER_A --persist True
Connected to iot.eclipse.org:1883/NOELWEATHER_A Result: 0
Insert DB: NOELWEATHER_A/h, val:48.7972717285
Insert DB: NOELWEATHER_A/t, val:29.31
Insert DB: NOELWEATHER_A/p, val:1043.44677734
```

- Let's push the solution harder by using second ('bme680') board and MQTT broker...
- Open New Terminal on hostB and publish to/from different broker/board-

```
./mqtt.py -a publish -u mqtt://test.mosquitto.org:1883/NOELWEATHER_B --board bme680
Connected to test.mosquitto.org:1883/NOELWEATHER_B Result: 0
topic:NOELWEATHER_B/h, val:48.9634399414
topic:NOELWEATHER_B/t, val:29.42
topic:NOELWEATHER_B/p, val:1043.48852539
```

- Back on HostA, subscribe to the new channel and persist data too-

* On hostB, persist the same mqtt weather messages to TinyDB by setting flag-

```
sudo ./mqtt.py -a subscribe -u mqtt://test.mosquitto.org:1883/NOELWEATHER_B --persist True
```

```
Connected to test.mosquitto.org:1883/NOELWEATHER_B Result: 0
Insert DB: NOELWEATHER_B/h, val:48.7972717285
Insert DB: NOELWEATHER_B/t, val:29.31
Insert DB: NOELWEATHER_B/p, val:1043.44677734
```

This shows a working MQTT publisher/subscriber redundant weather station solution.

My MQTT python solution is very extensible. For example-

- Use systemd to manage services (see example for ble in same repo).
- Add more boards (see mqtt/lib/myboard/myboard.py class library).
- Add /etc/solution.conf file – we would probably need this when (A) using systemd to manager services and/or (B) by extending the code to read the configuration file and launch threads for each publisher/subscriber service.
- Use MongoDB as backend database.
- Build dashboards to read the persisted weather data.

Redundant UI Dashboards

The missing ingredient is a front end. I am unsure how to integrate the 'db.json' database with Cloud IoT Platforms and service providers but Blynk illustrates the requirement instead.

Blynk Dashboards

My solution is published but reproduced here. All code is my own.

- Clone the repo.

```
git clone https://github.com/noelmcloughlin/iot-edge-stepping-stones.git
cd iot-edge-stepping-stones
```

- Setup iot platform access tokens how you like. I use ~/.bash_profile to manage as environment variables. Remember sudo -E for scripts needing elevated privledges.

```
vi ~/.bash_profile
### My IoT Device Inventory
export BLYNK_TOKEN_SENSEHAT="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
export BLYNK_TOKEN_RPI3BPLUS=""

### environment variables needed by iot-pi-stepping-stones ###
export MY_BLYNK_TOKEN="${BLYNK_TOKEN_SENSEHAT}"
```

- Optionally reinstall node.js:

```
./blynk/configure_blynk.py -n reinstall
```

- Use Blynk to control your device using a Virtual Pin.
- Ensure nodeJs is installed:

```
./blynk/configure_blynk.py -n install
```

- Ensure node libraries are encapsulated with device NodeJS code:

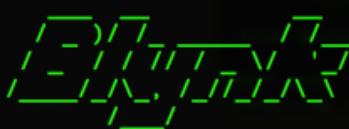
```
./blynk/configure_blynk.py -d pi/2b
```

- Start nodejs service on-device:

```
sudo -E node ./pi/2b/index.js
```

- You should see output like the following:

```
pi@rasppi2b:~/iot-edge-stepping-stones/blynk$ vi sensehat/4vpin/index.js
pi@rasppi2b:~/iot-edge-stepping-stones/blynk$ sudo -E node sensehat/4vpin/index.js
Settings file RTIMULib.ini loaded
Using fusion algorithm RTQF
min/max compass calibration not in use
Ellipsoid compass calibration not in use
Accel calibration not in use
LSM9DS1 init complete
```



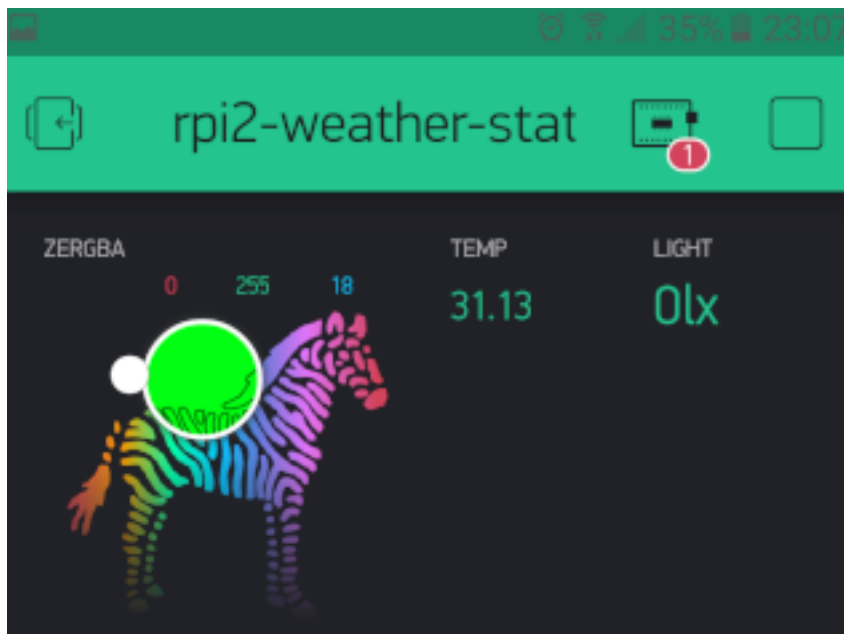
```
Give Blynk a Github star! => https://github.com/vshymanskyi/blynk-library-js
```

```
OnOff mode
Connecting to TCP: blynk-cloud.com 80
Connected
```

- Go to BlynkApp and Interact with your device using this virtual pin.
- Tip: remember to run `./configure_blynk.py -d <devsubdir>` to avoid this error:

```
$ sudo -E node sensehat/3vpin/index.js
module.js:549
  throw err;
  ^
```

The required **Dashboard** can be extended from widgets in the Blynk Mobile App-



Integration with WIA

- Clone this repo.

```
git clone https://github.com/noelmccloughlin/iot-edge-stepping-stones.git
cd iot-edge-stepping-stones
```

- Setup iot platform access tokens how you like. I use ~/.bash_profile to manage as environment variables. Remember sudo -E for scripts needing elevated priviledges.

```
vi ~/.bash_profile
### My IoT Device Inventory
export WIA_TOKEN_SENSEHAT="d_sk_XXXXXXXXXXXXXXXXXXXXXpi"
export WIA_TOKEN_RPI3BPLUS="d_sk_YYYYYYYYYYYYYYYYYYYYYYY"
export MY_WIA_TOKEN="${WIA_TOKEN_SENSEHAT}"

### environment variables needed ##
export MY_WIA_TOKEN="${WIA_TOKEN_SENSEHAT}"
```

- Execute command to setup the software

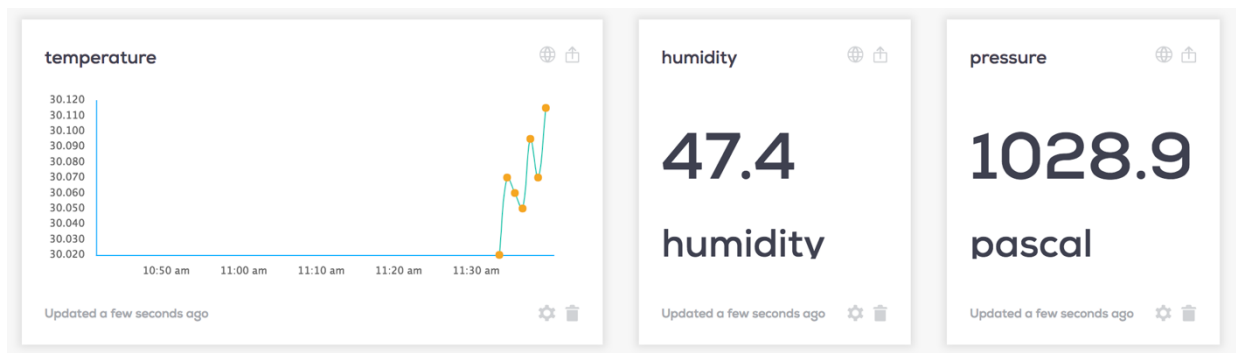
```
./wia/configure_wia.py -a install
```

The SenseHAT has temperature, pressure, and humidity sensors useful as Weather station.

- Open Terminal #1 and run script to publish weather events to WIA-

```
sudo -E ./wia/configure_wia.py -s weather
```

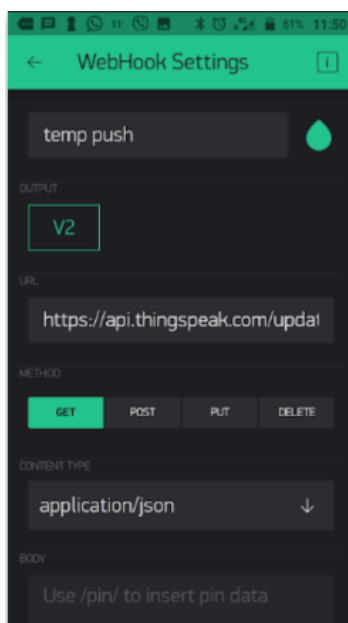
- WIA consumes your Pi SenseHat temp/humidity/pressure data.



Send data to Thingspeak from Blynk

BlynkApp Webhook widget can communicate with 3rd party services like ThingSpeak.

- Go to ThingSpeak.com and create new temperature channel.
- Go to BlynkApp and add "Webhook" from the widget box.
- Monitor V2 and send data to Thingspeak (see: <https://docs.blynk.cc/#widgets-other-webhook>):



My solution is published but reproduced here. All code is my own.

- Clone the repo.

```
git clone https://github.com/noelmcloughlin/iot-edge-stepping-stones.git
cd iot-edge-stepping-stones
```

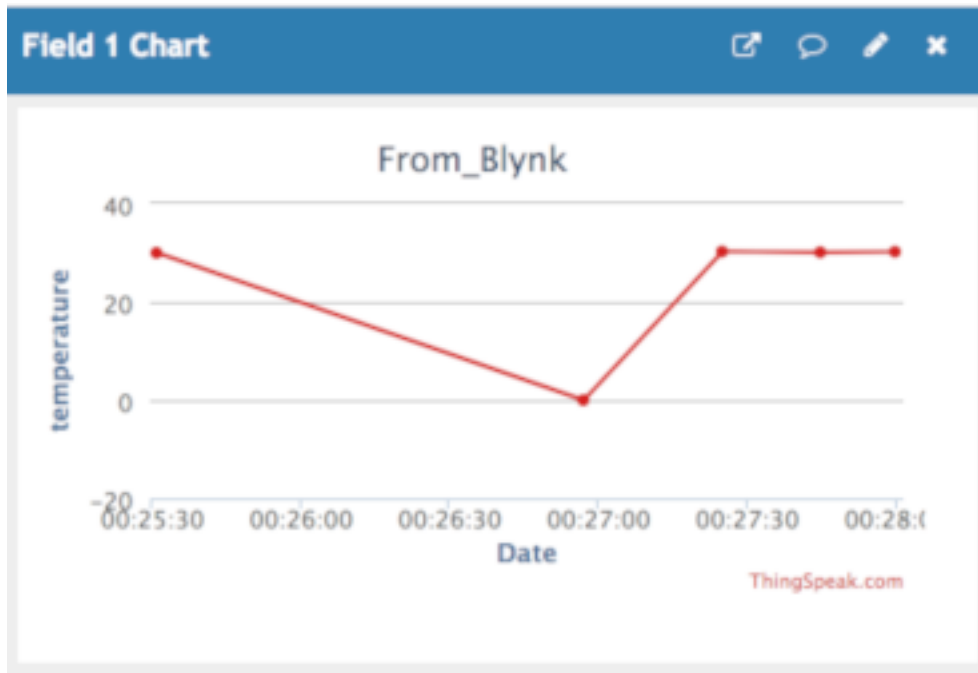
- Setup nodejs dependencies:

```
./thingspeak/configure_blynk.py -d sensehat/3vpin
```

- Run node app again:

```
sudo -E node ./sensehat/3vpin/index.js
```

- Go to your thingspeak.com channel to observe temperature data from blynk:



Thanks

Thanks to Frank Walsh and Caroline Cahill at Waterford Institute of Technology for educating me on the fundamental building blocks for this project.