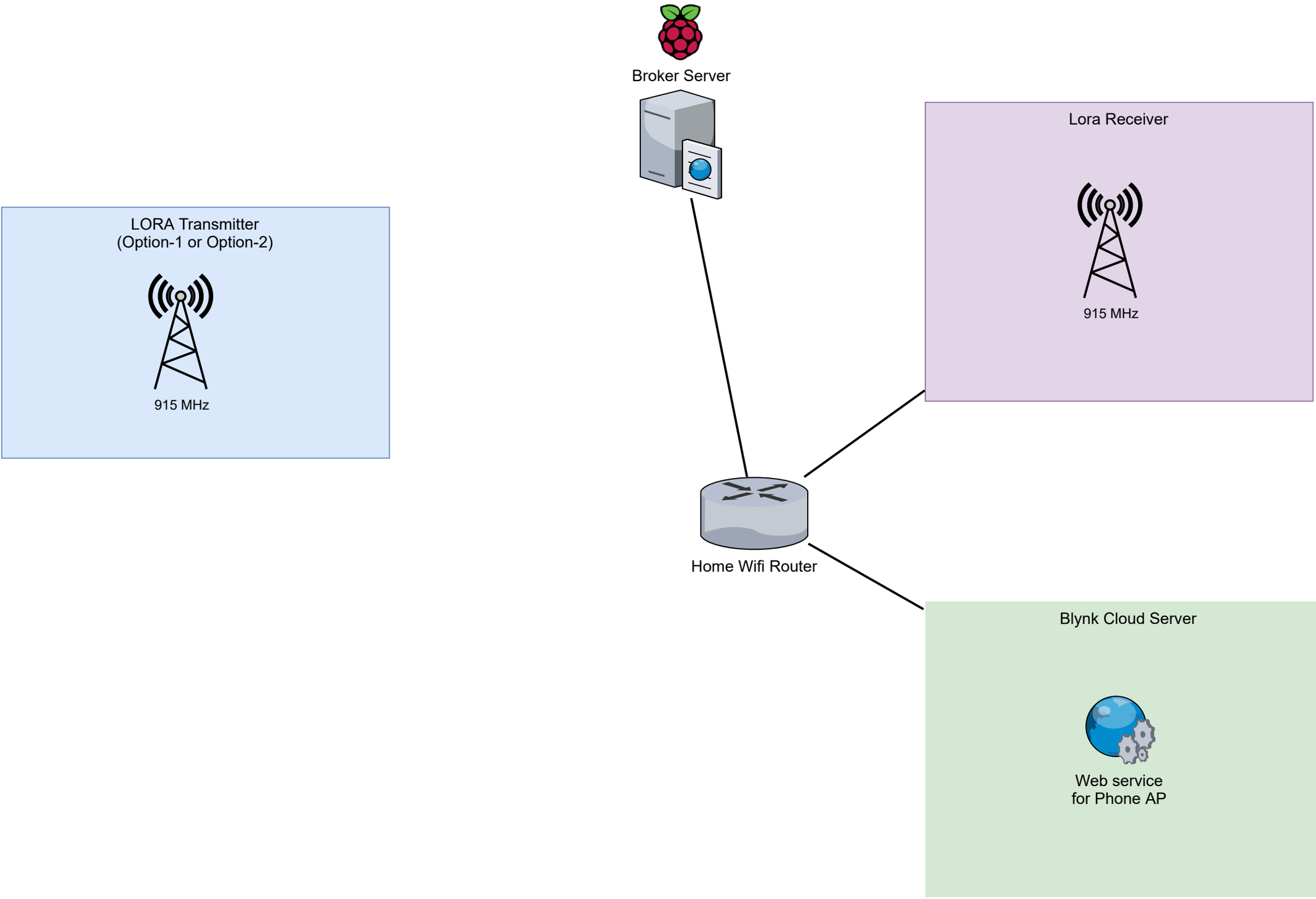
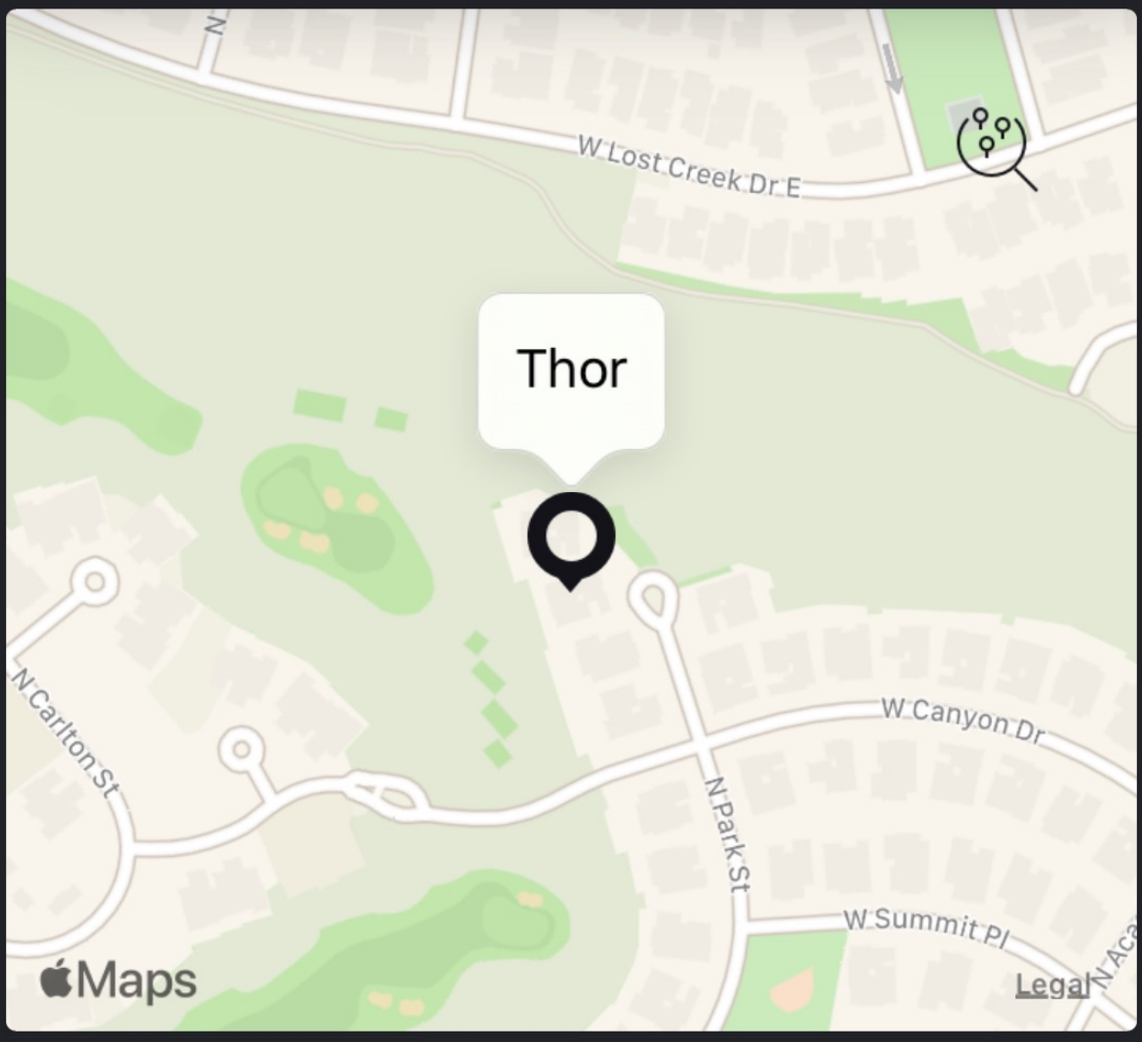
Lora GPS Tracker with MQTT and Blynk Mobile AP Integration



Thor Tracker



LATITUDE

33.487408

LONGITUDE

-112.503876

LORA READING ID:

289

BATTERY VOLTAGE

4.07

If you have any problems during the installation procedure, take a look at the ESP32 Troubleshooting Guide .

If you like the ESP32, enroll in our course: Learn ESP32 with Arduino IDE.

Prerequisites: Arduino IDE Installed

Before starting this installation procedure, make sure you have the latest version of the Arduino IDE installed in your computer. If you don't, uninstall it and install it again. Otherwise, it may not work.

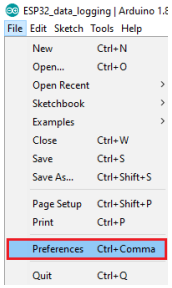
Having the latest Arduino IDE software installed from arduino.cc/en/Main/Software , continue with this tutorial.

Do you need an ESP32 board? You can buy it here .

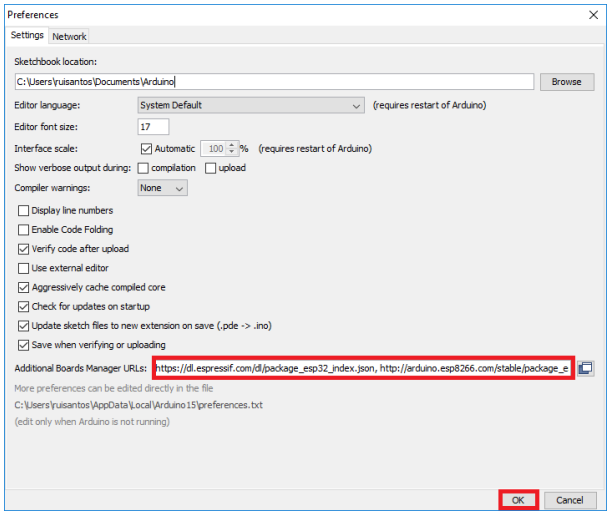
Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**



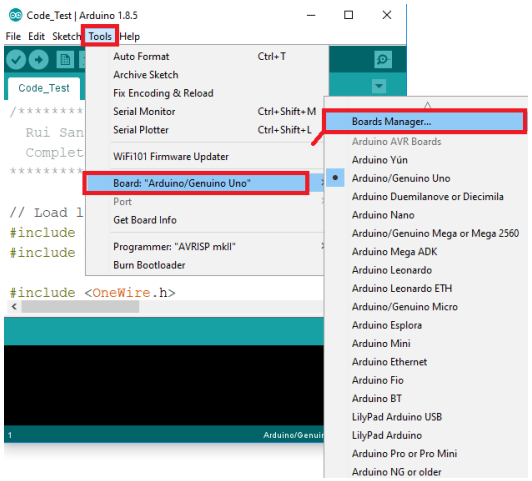
2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the "Additional Board Manager URLs" field as shown in the figure below. Then, click the "OK" button:



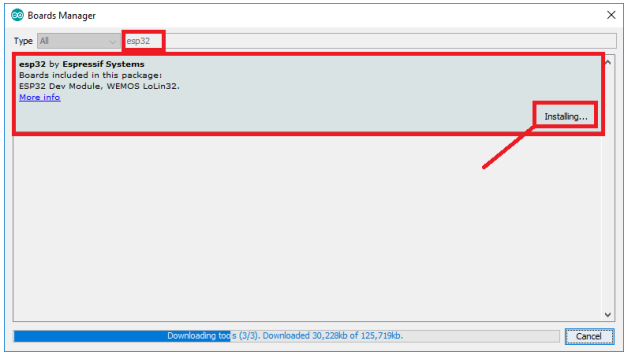
Note: if you already have the ESP8266 boards URL, you can separate the URLs with a comma as follows:

```
https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

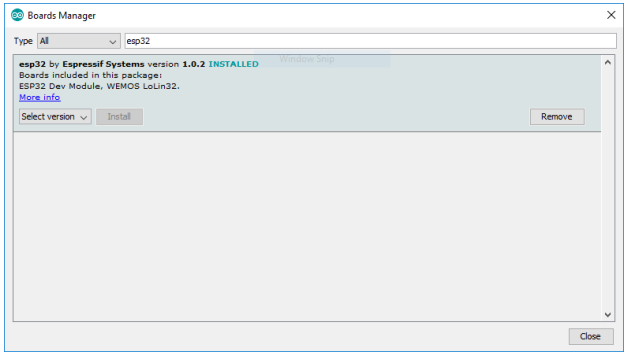
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager...**



4. Search for **ESP32** and press install button for the " **ESP32 by Espressif Systems**":



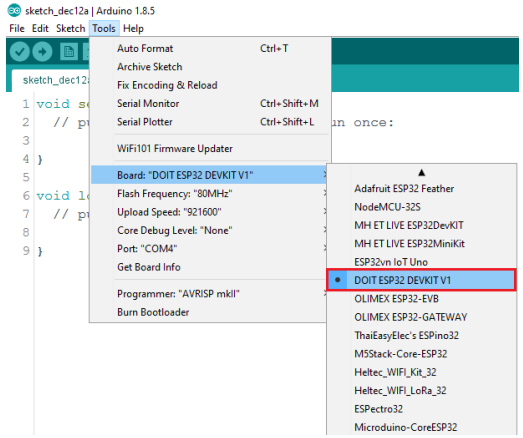
5. That's it. It should be installed after a few seconds.



Testing the Installation

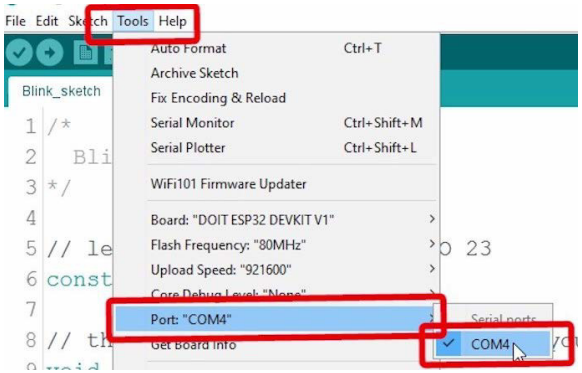
Plug the ESP32 board to your computer. With your Arduino IDE open, follow these steps:

1. Select your Board in **Tools** > **Board** menu (in my case it's the **DOIT ESP32 DEVKIT V1**)

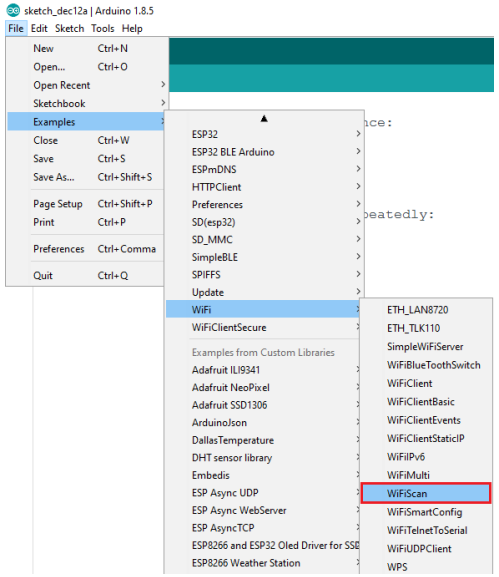


Search ...

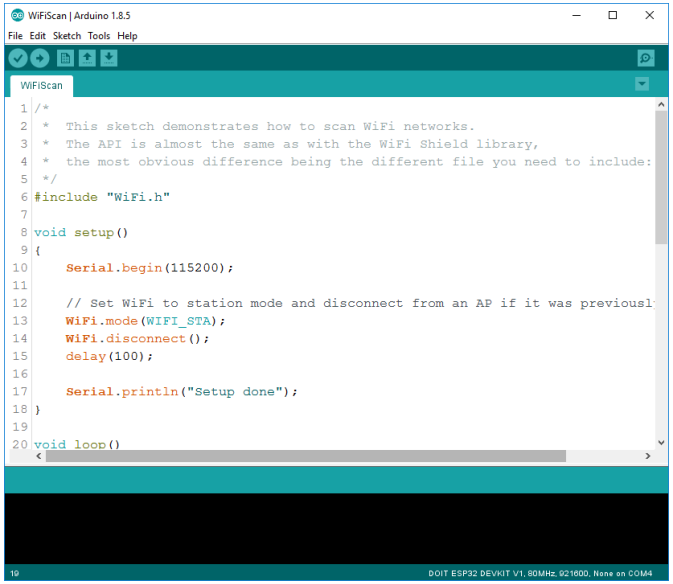
2. Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers):



3. Open the following example under **File > Examples > WiFi (ESP32) > WiFiScan**

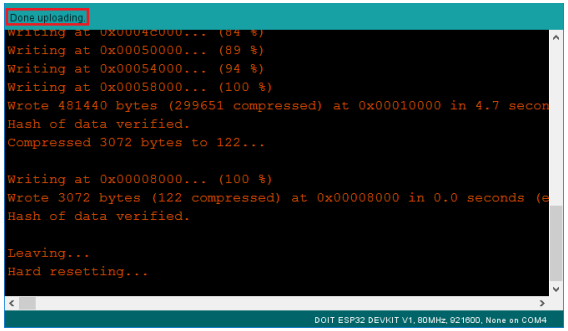


4. A new sketch opens in your Arduino IDE:



5. Press the **Upload** button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.

6. If everything went as expected, you should see a "**Done uploading.**" message.

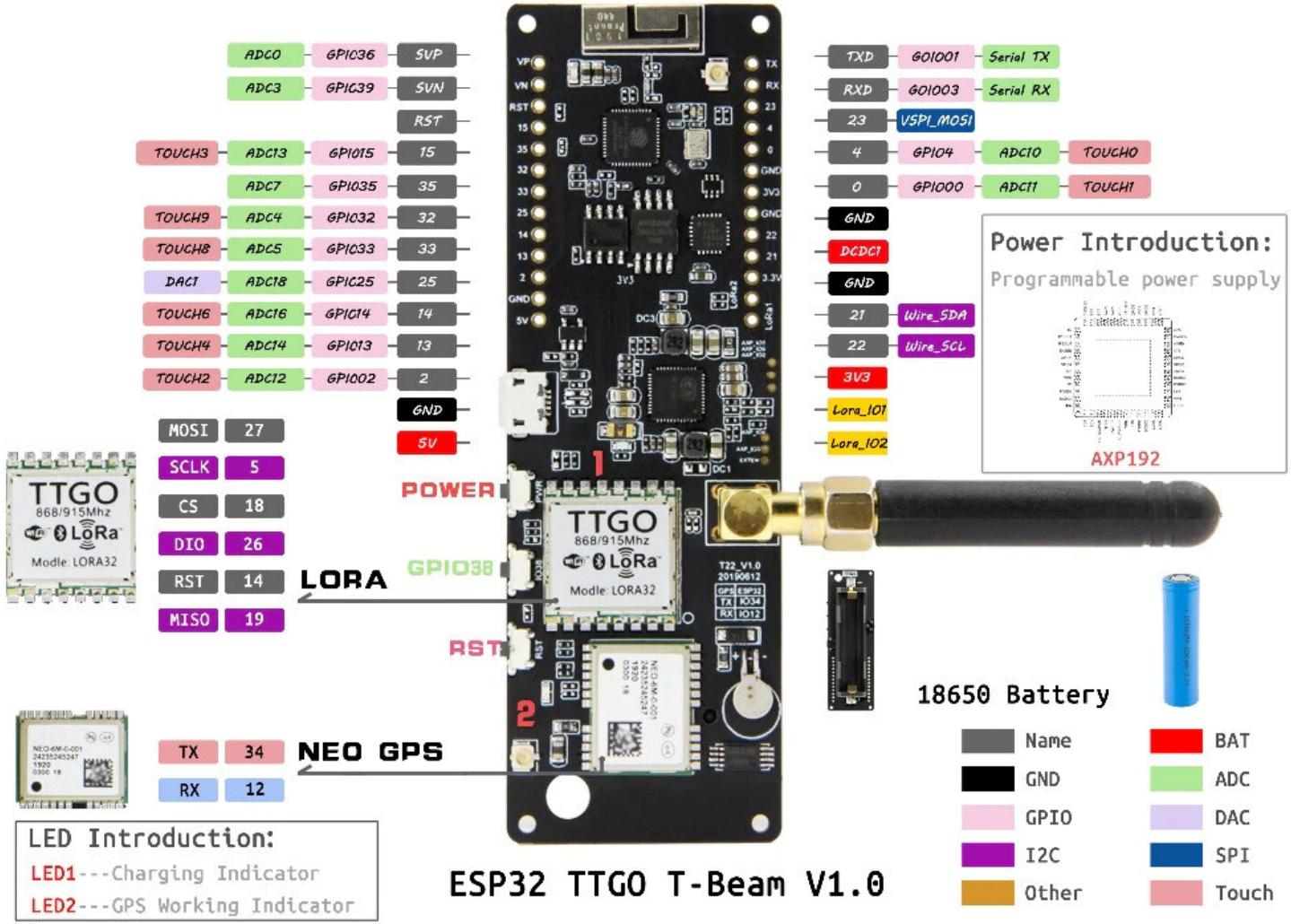


7. Open the Arduino IDE Serial Monitor at a baud rate of 115200:

8. Press the ESP32 on-board **Enable** button and you should see the networks available near your ESP32:

For this project, we'll use the following components:

Lilygo T-Beam Rev1 (This is the LORA Transmitter {option-1} to place on a dog, bike, etc.)



You will need to install the following libraries in Arduino (if not already installed):

- WiFiConnectOLED.h
- RadioLib
- Button2
- esp8266-oled-ssd1306
- TinyGPSPlus

Upload the completed sketch (See Transmitter Code Option-1) by following these instructions using the Arduino Program:

Go to **Tools** > **Port** and select the COM port the device is connected to. Then, go to **Tools** > **Board** and select the board you're using. In our case, it's the T-Beam. Now go to the top menu, and click upload.

This is a LORA GPS transmitter and receiver controlled by the center button:

The Menu choices are:

- Power Status of Module
- GPS coordinates
- LORA Transmit Function of GPS coordinates with battery condition
- LORA Receive Data Function

Note – To use this device with the Lora Receiver, set the device to transmit. The GPS coordinates will be transmitted to the Lora receiver when a satellite lock occurs (this can take up to 5 minutes) along with the battery voltage and Lora Reading.

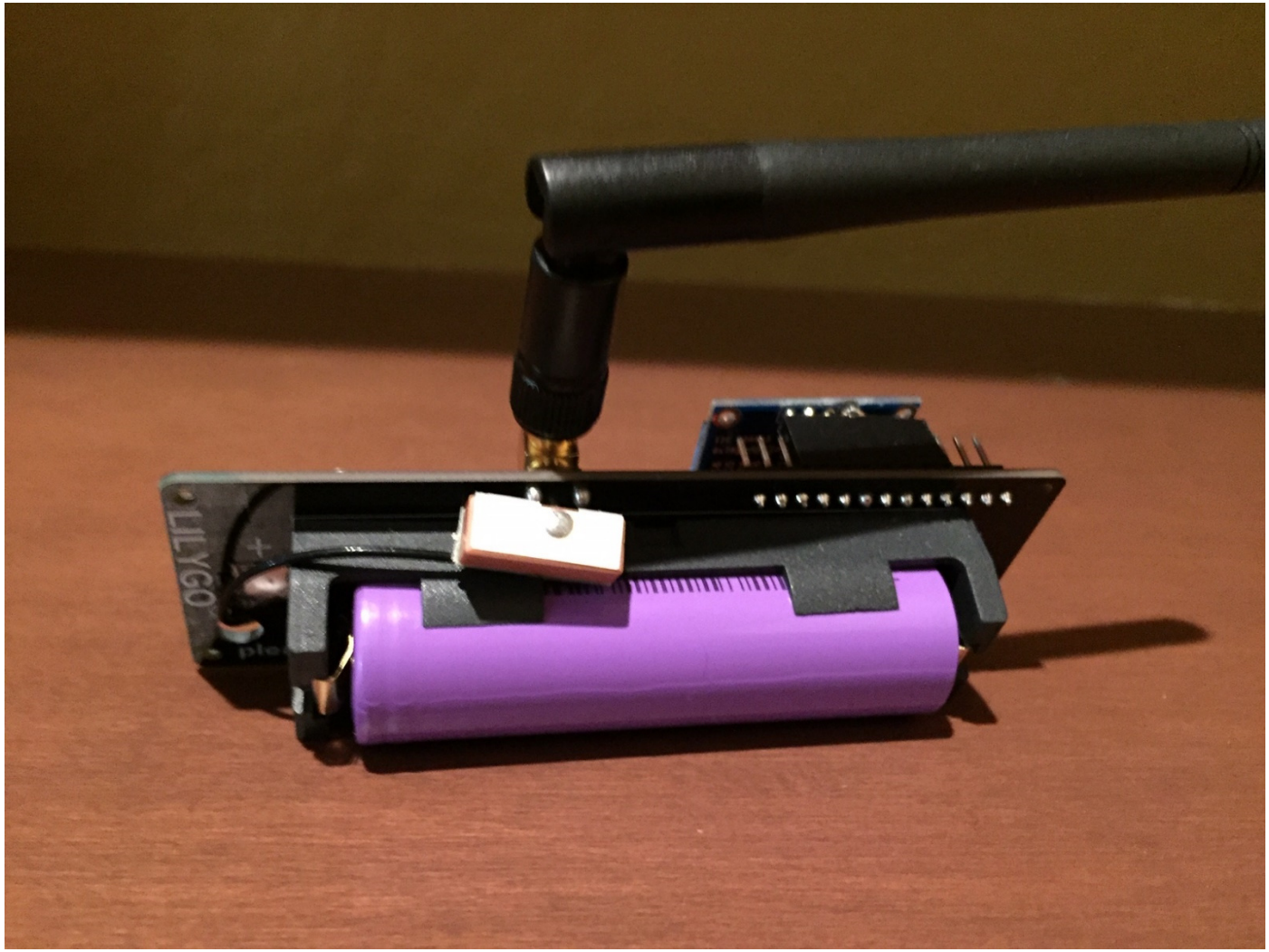
Note – If you are using the battery, Hold the middle button for three seconds and release to place device in Sleep Mode to shut off. To turn back on, and bring the device out of sleep mode, use the power button.

Note – I added an OLED display mapped to the following pins:

Lora > OLED Display

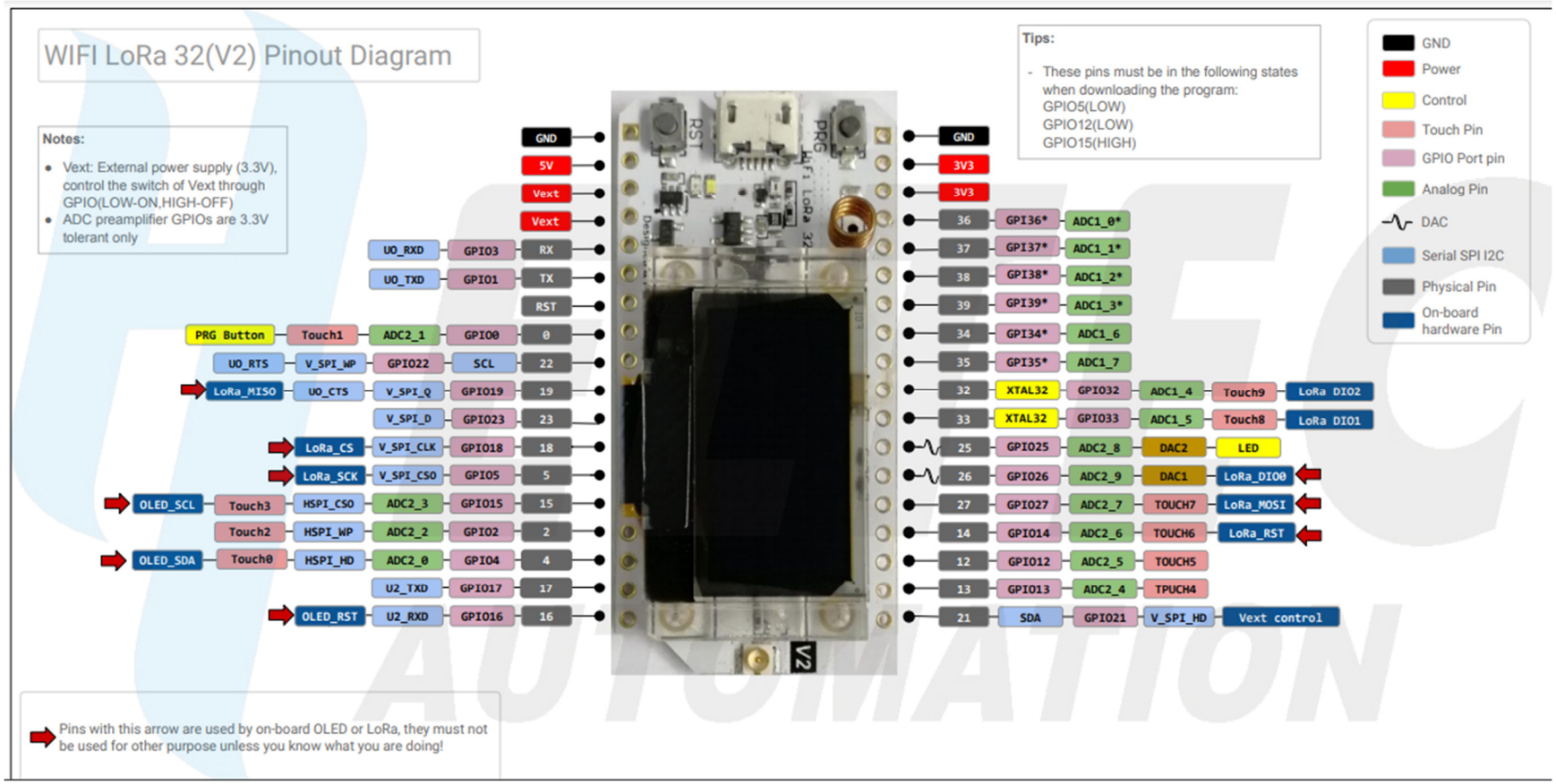
DCDC1 = 3.3v
GND = GND
21 = SDA
22 = SCL

Place device (use middle button into Lora Sender mode)


LILYGO

For this project, we'll use the following components:

- **Heltec LoRa32 SX1276 OLED board (This is the LORA Transmitter that will be placed on a dog, bike, etc.)**



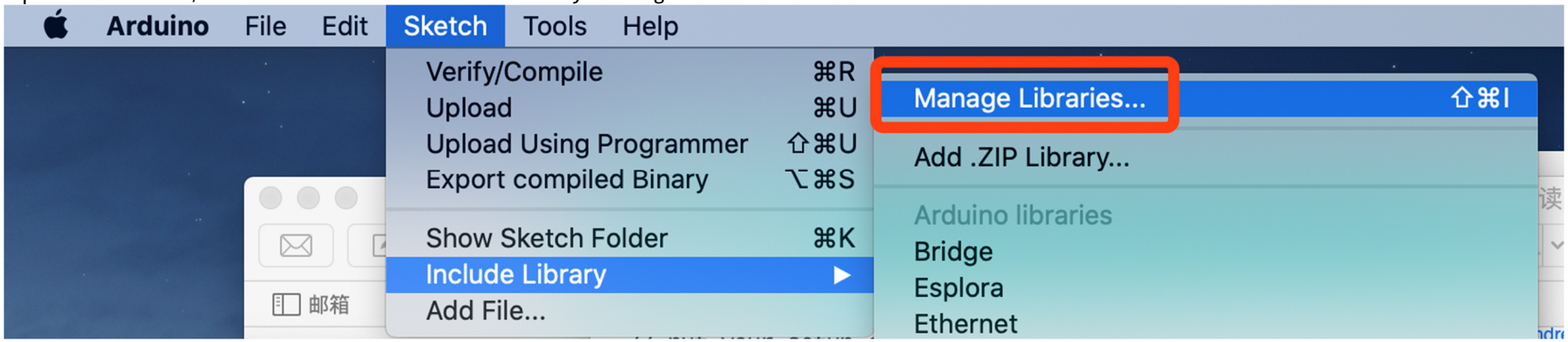
Heltec_ESP32 Library

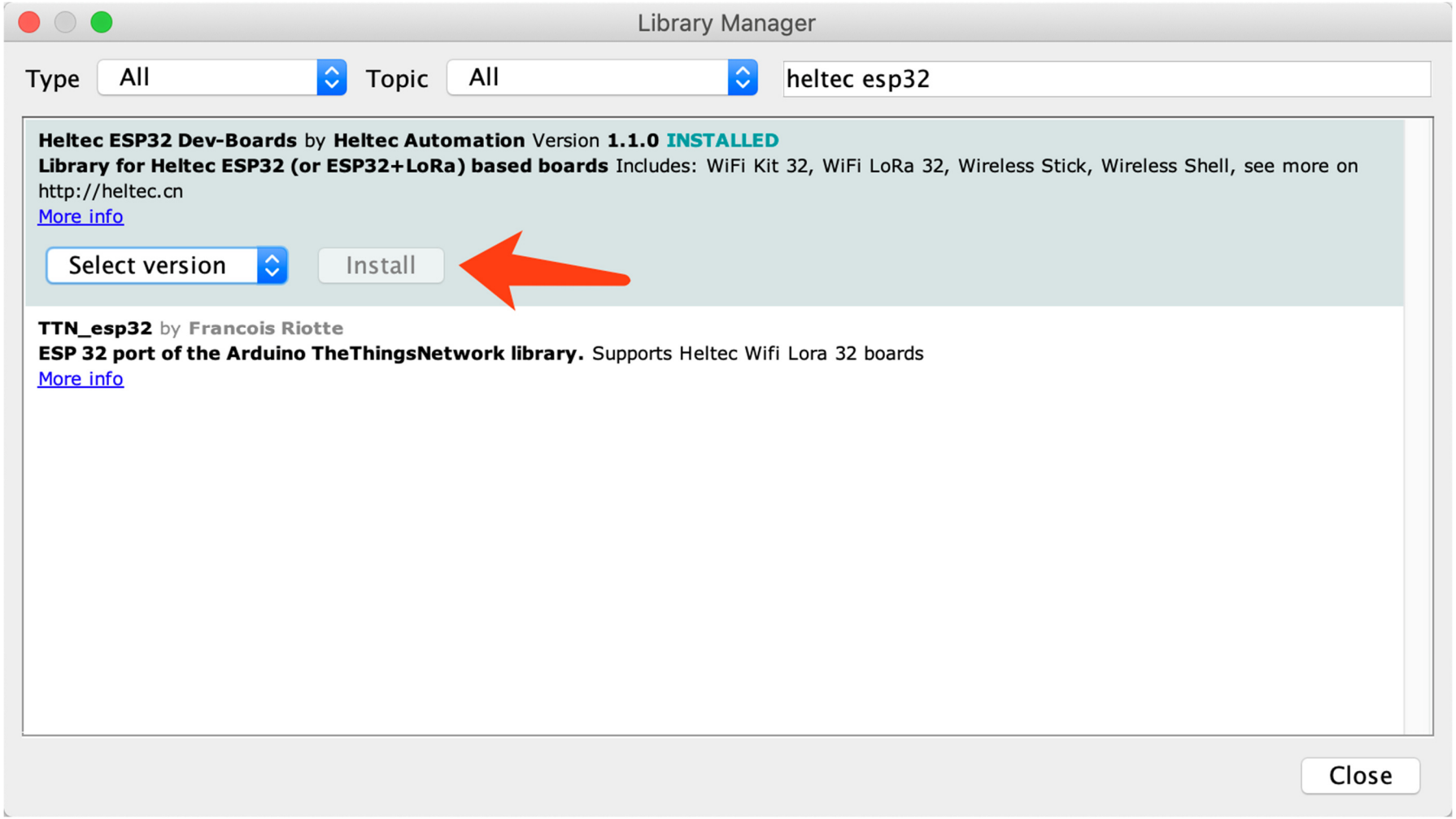
How to install this library

We recommend using the Arduino library manager, it's the simplest way

Use Arduino Library Manager

Open Arduino IDE, then Select `Sketch->Include Library->Manage Libraries...` Search `Heltec ESP32` and install it.




Library Manager
Type
All
Topic
All
heltec esp32
Heltec ESP32 Dev-Boards by Heltec Automation Version 1.1.0 INSTALLED
Library for Heltec ESP32 (or ESP32+LoRa) based boards Includes: WiFi Kit 32, WiFi LoRa 32, Wireless Stick, Wireless Shell, see more on http://heltec.cn
More info
Select version
Install
TTN_esp32 by Francois Riotte
ESP 32 port of the Arduino TheThingsNetwork library. Supports Heltec Wifi Lora 32 boards
More info
Close

You will need to install the following libraries in Arduino (if not already installed):

- SPI.h
- LoRa.h
- TinyGPS++.h
- Wire.h
- Adafruit_GFX.h
- Adafruit_SSD1306.h

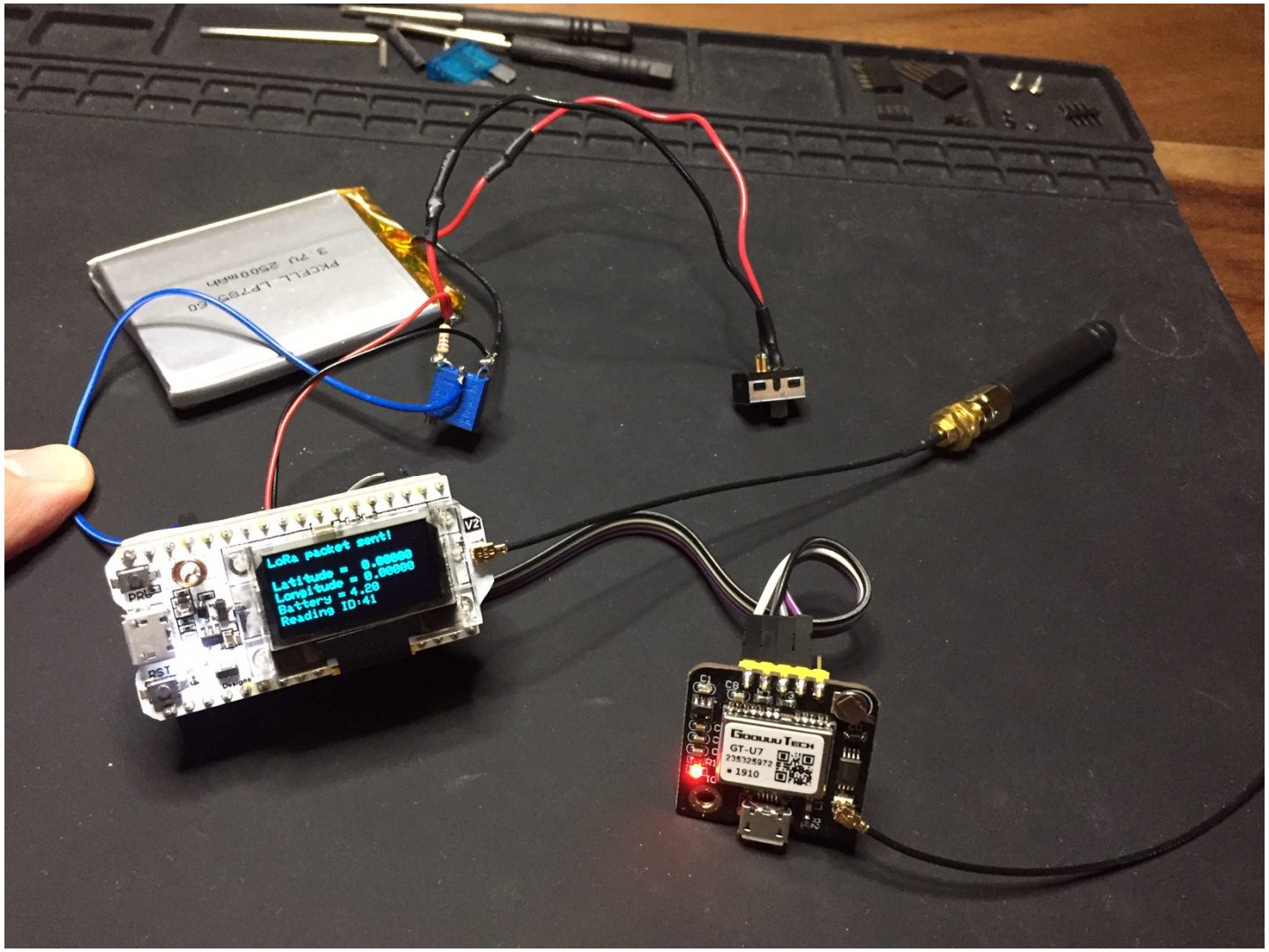
Upload the completed sketch (See Transmitter Code Option-2) by following these instructions using the Arduino Program:

Go to **Tools** > **Port** and select the COM port the device is connected to. Then, go to **Tools** > **Board** and select the board you're using. In our case, it's the Heltec WiFi Lora 32(V2). Now go to the top menu, and click upload.

Note – I added a 3.7 volt @ 1200mAH rechargeable battery and a power switch.

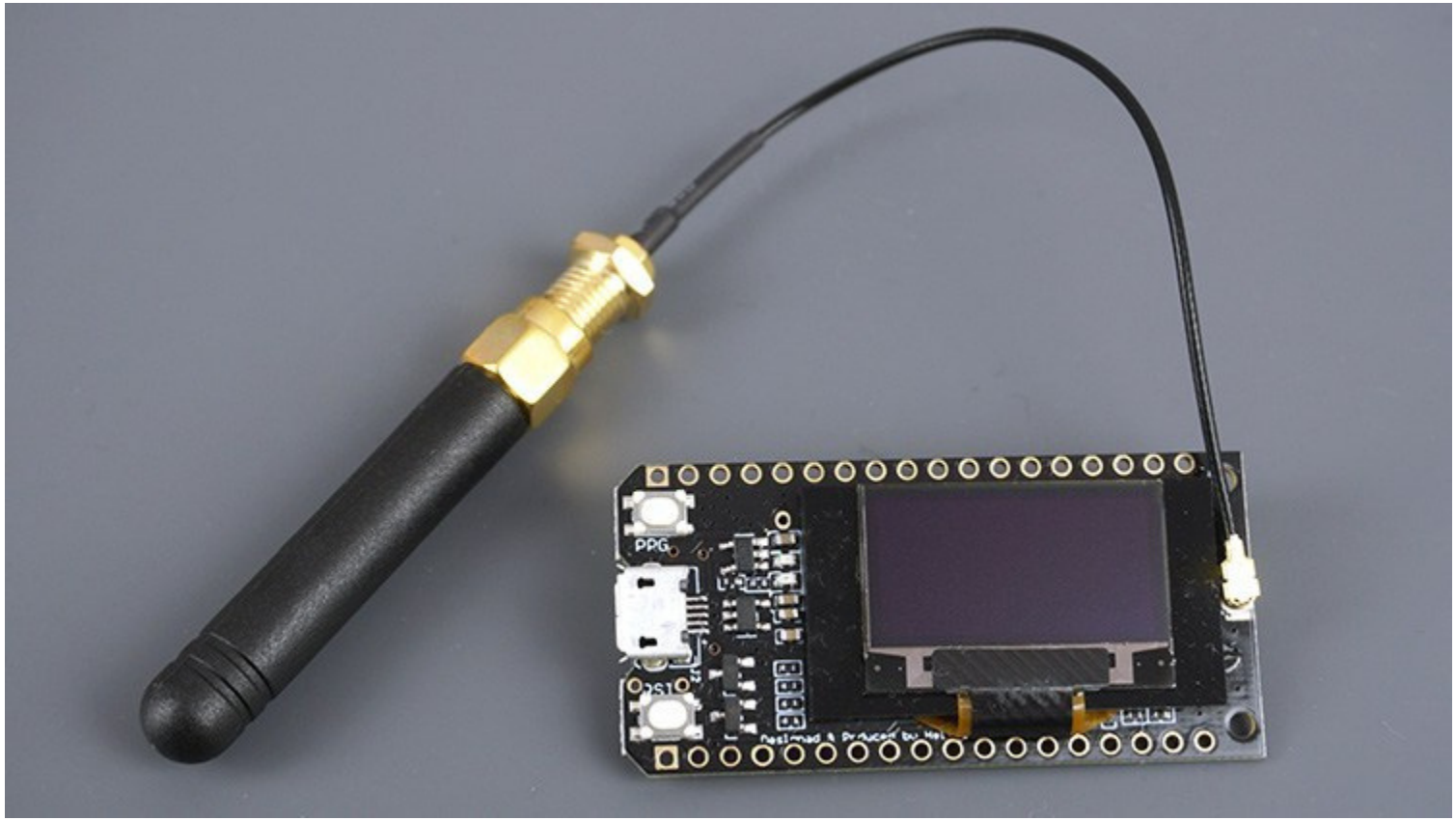
Note – When using this device with the Lora Receiver, the GPS coordinates will be transmitted to the Lora receiver when a satellite lock occurs (this can take up to 5 minutes) along with the battery voltage and Lora Reading.

Here is a picture with the battery pack, GPS module, power switch, and voltage divider network (scaled to read 5Vdc at 3.3Vdc input)



For this project, we'll use the following components:

- **TTGO LoRa32 SX1276 OLED board (This is the LORA Receiver & WiFi Module inside your house)**



You will need to install the following libraries in Arduino (if not already installed):

- WiFiConnectOLED.h
- "SSD1306.h"
- Wire

- WiFi.h
- WebServer.h
- PubSubClient.h
- LoRa.h
- SPI.h

Upload the completed sketch (See Receiver Code) by following these instructions using the Arduino Program:

Go to **Tools** > **Port** and select the COM port the device is connected to. Then, go to **Tools** > **Board** and select the board you're using. In our case, it's the TTGO LoRa32-OLED V1. Now go to the top menu, and click upload.

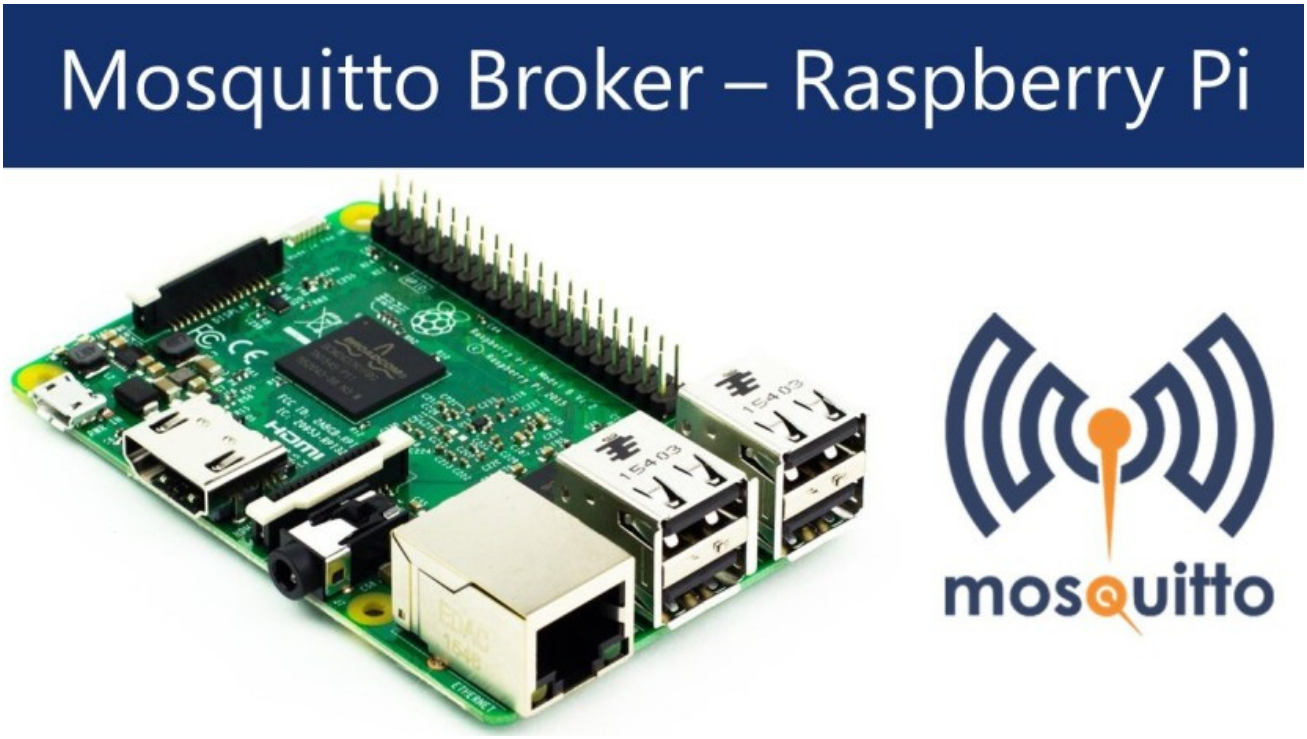
Note- Turn device on and setup you Wifi byfollowing the instructions on the "OLED Screen".

Note – This Lora receiving device works with a Raspberry Pi Model 3B+ through your house WiFi and the MQTT Protocol.. Refer to separate instructions for setting up the Raspberry Pi.

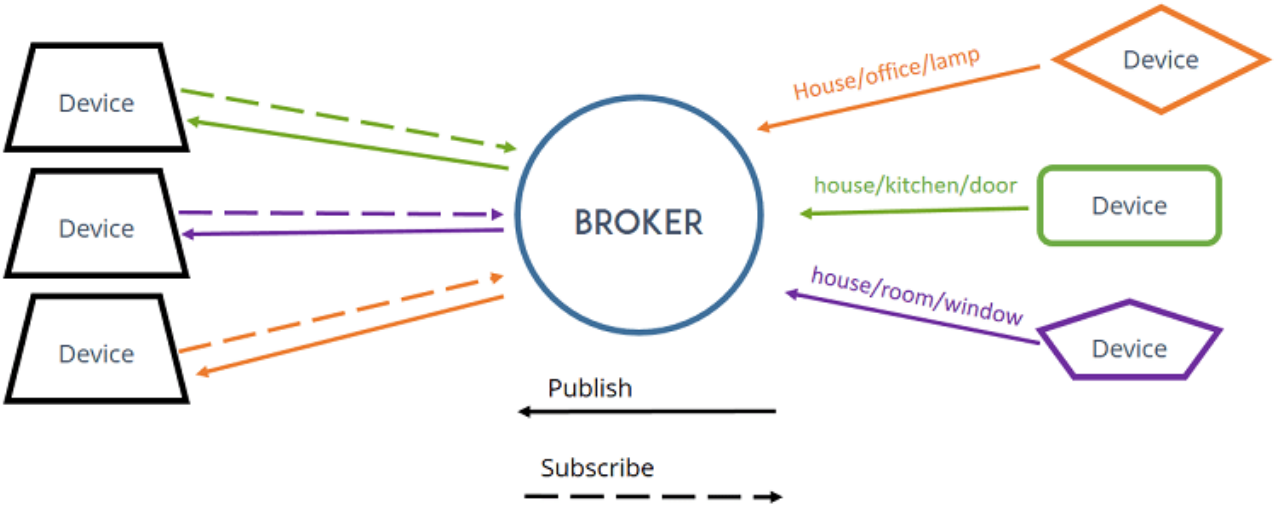
If all goes well, you should see the following screen when a massage is received and the system is running:

How to Install Mosquitto Broker on Raspberry Pi

This guide explains how to install the Mosquitto Broker for MQTT communication on a Raspberry Pi board.



The broker is primarily responsible for **receiving** all messages, **filtering** the messages, **decide** who is interested in it and then **publishing** the message to all subscribed clients.



There are several brokers you can use. In our Home Automation projects we use the **Mosquitto Broker** installed on a Raspberry Pi.

Prerequisites

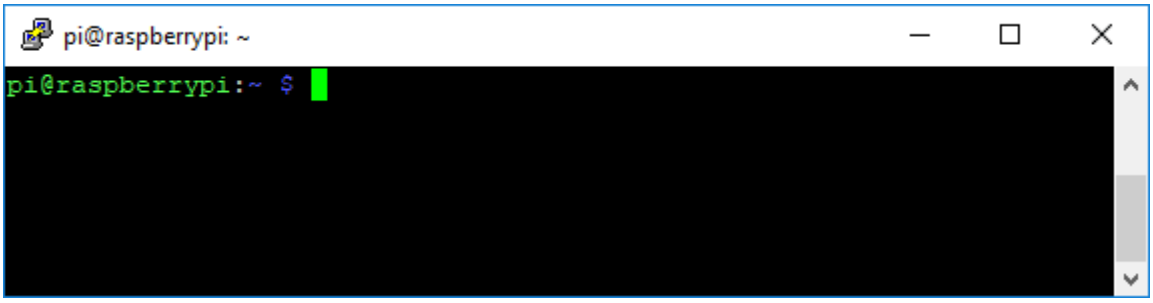
Before continuing with this tutorial

- You should be familiar with the Raspberry Pi board – read Getting Started with Raspberry Pi;
- You should have the Raspbian or Raspbian Lite operating system installed in your Raspberry Pi – read Installing Raspbian Lite, Enabling and Connecting with SSH;
- You also need the following hardware:
  - Raspberry Pi board – read Best Raspberry Pi Starter Kits
  - MicroSD Card – 16GB Class10
  - Raspberry Pi Power Supply (5V 2.5A)

After having your Raspberry Pi board prepared with Raspbian OS, you can continue with this tutorial. Let's install the **Mosquitto Broker**.

Installing Mosquitto Broker on Raspbian OS

Open a new Raspberry Pi terminal window:



To install the Mosquitto Broker enter these next commands:

```
pi@raspberry:~ $ sudo apt update
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients
```

You'll have to type **Y** and press **Enter** to confirm the installation. To make Mosquitto auto start on boot up enter:

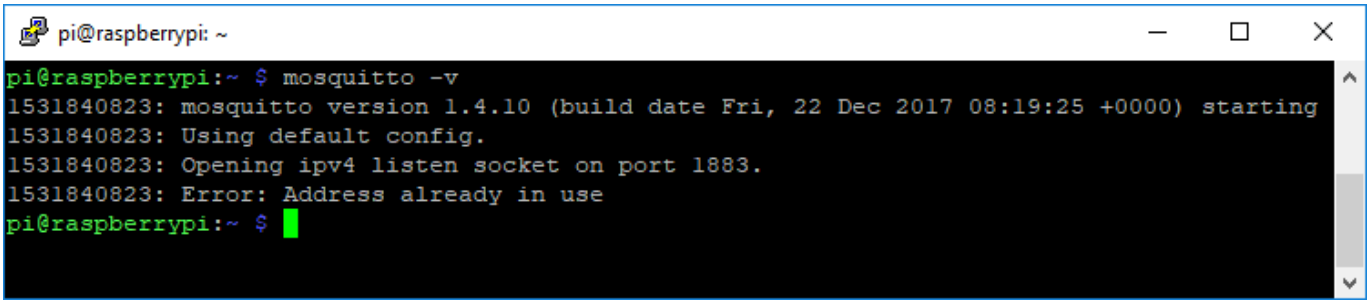
```
pi@raspberry:~ $ sudo systemctl enable mosquitto.service
```

Testing Installation

Send the command:

```
pi@raspberry:~ $ mosquitto -v
```


This returns the Mosquitto version that is currently running in your Raspberry Pi. It should be 1.4.X or above.

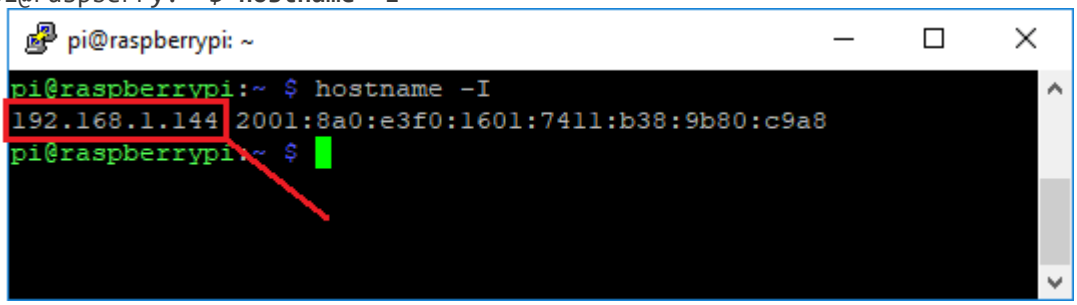


Note: sometimes the command *mosquitto -v* prompts a warning message saying "*Error: Address already in use*". That warning message means that your Mosquitto Broker is already running, so don't worry about that.

Raspberry Pi IP Address

To use Mosquitto broker later on your projects, you'll need your Raspberry Pi IP address. To retrieve your Raspberry Pi IP address, type the next command in your Terminal window:

```
pi@raspberry:~ $ hostname -I
```



In our case, the Raspberry Pi IP address is **192.168.1.144**. Save your Raspberry Pi IP address because you'll need it in future projects.

Note – Make this IP address "Static". Details are not shown here, can be Googled.

Testing MQTT Broker Installation

After installing MQTT Broker, I recommend installing an MQTT CIlient to test the Broker installation and publish sample messages.
The next command shows how to install MQTT Mosquitto Client:

```
pi@raspberry:~ $ sudo apt-get install mosquitto-clients
```

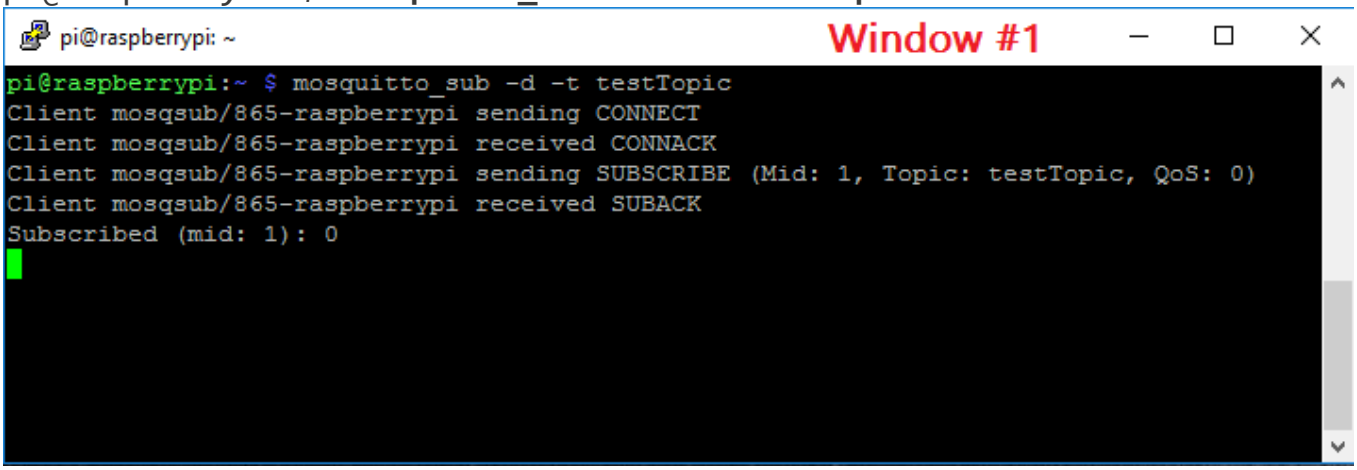
You'll have to type **Y** and press **Enter** to confirm the installation.
Run Mosquitto on background as a daemon:

```
pi@raspberry:~ $ mosquitto -d
```

Subscribing to testTopic Topic

To subscribe to an MQTT topic with Mosquitto Client open a terminal Window #1 and enter the command:

```
pi@raspberry:~ $ mosquitto_sub -d -t testTopic
```

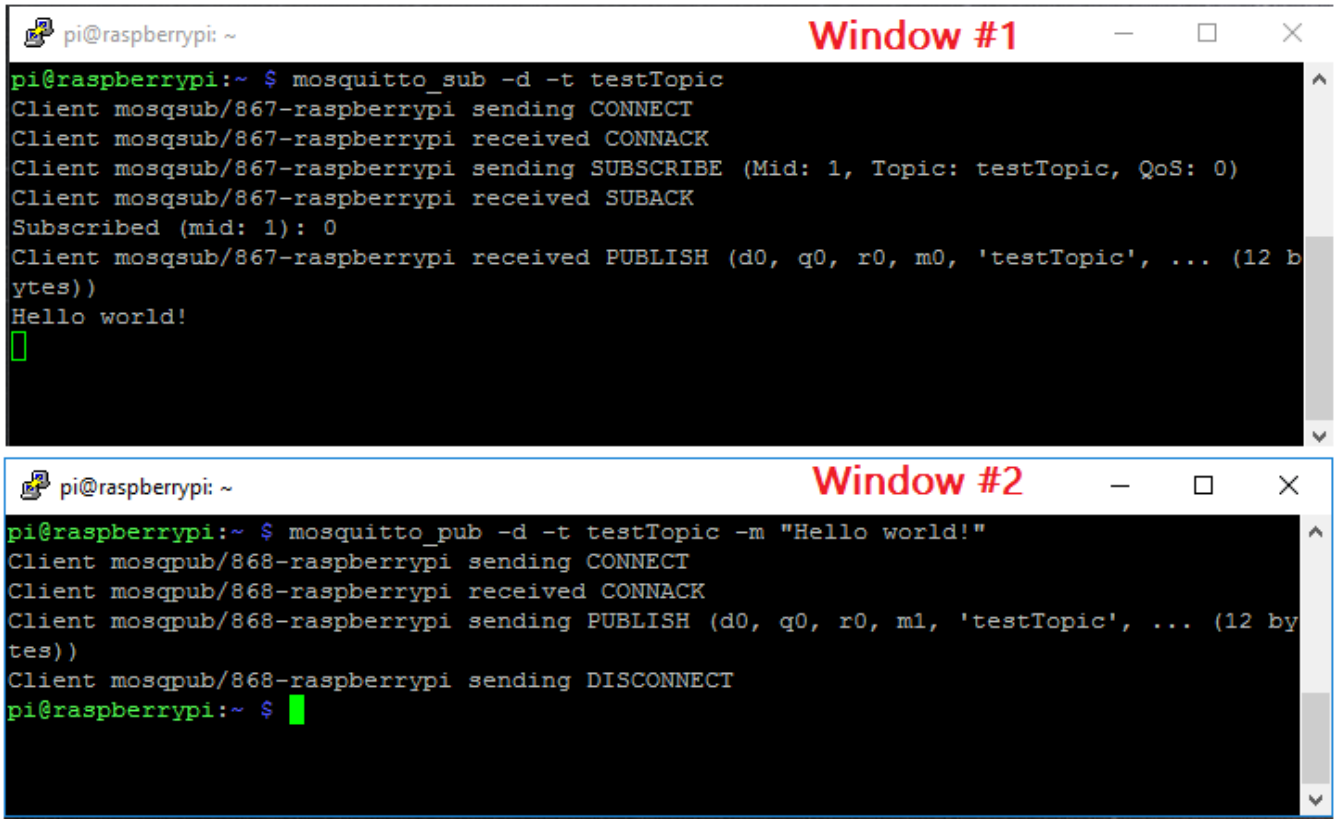


You're now subscribed to a topic called **testTopic.**

Publishing "Hello World!" Message to testTopic Topic

To publish a sample message to **testTopic**, open a terminal Window #2 and run this command:

```
pi@raspberry:~ $ mosquitto_pub -d -t testTopic -m "Hello world!"
```



The message "**Hello World!**" is received in Window #1 as illustrated in the figure above.

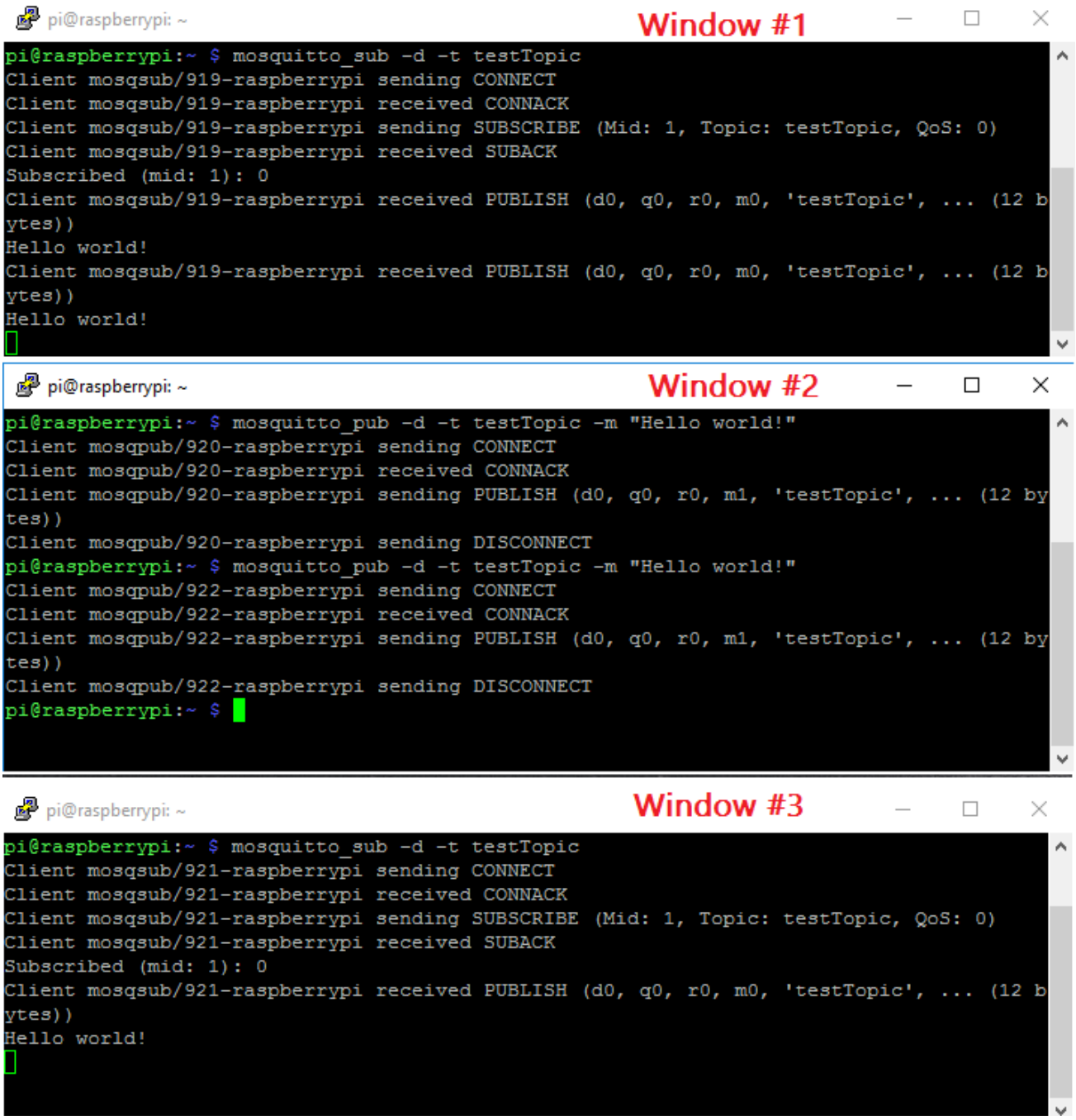
Publishing a Message to Multiple Clients

Having Window #1 still subscribed to topic testTopic, open a new terminal Window #3 and run this command to subscribe to **testTopic** topic:


```
pi@raspberry:~ $ mosquitto_sub -d -t testTopic
```

On Window #2 publish the "**Hello World!**" message:

```
pi@raspberry:~ $ mosquitto_pub -d -t testTopic -m "Hello world!"
```



Since two clients are subscribed to **testTopic** topic, they will both receive **“Hello world!”** message.

Next, if Node Red is not already installed on your raspberry pi, follow these instructions. If it was installed go to the “auto” start instructions and follow the rest of the instructions from there.

Installing Node-RED

Getting Node-RED installed in your Raspberry Pi is quick and easy. It just takes a few commands.

Having an SSH connection established with your Raspberry Pi, enter the following commands to install Node-RED:

```
pi@raspberry:~ $ bash <(curl -sL
https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

Autostart Node-RED on boot

To automatically run Node-RED when the Pi boots up, you need to enter the following command:

```
pi@raspberry:~ $ sudo systemctl enable nodered.service
```

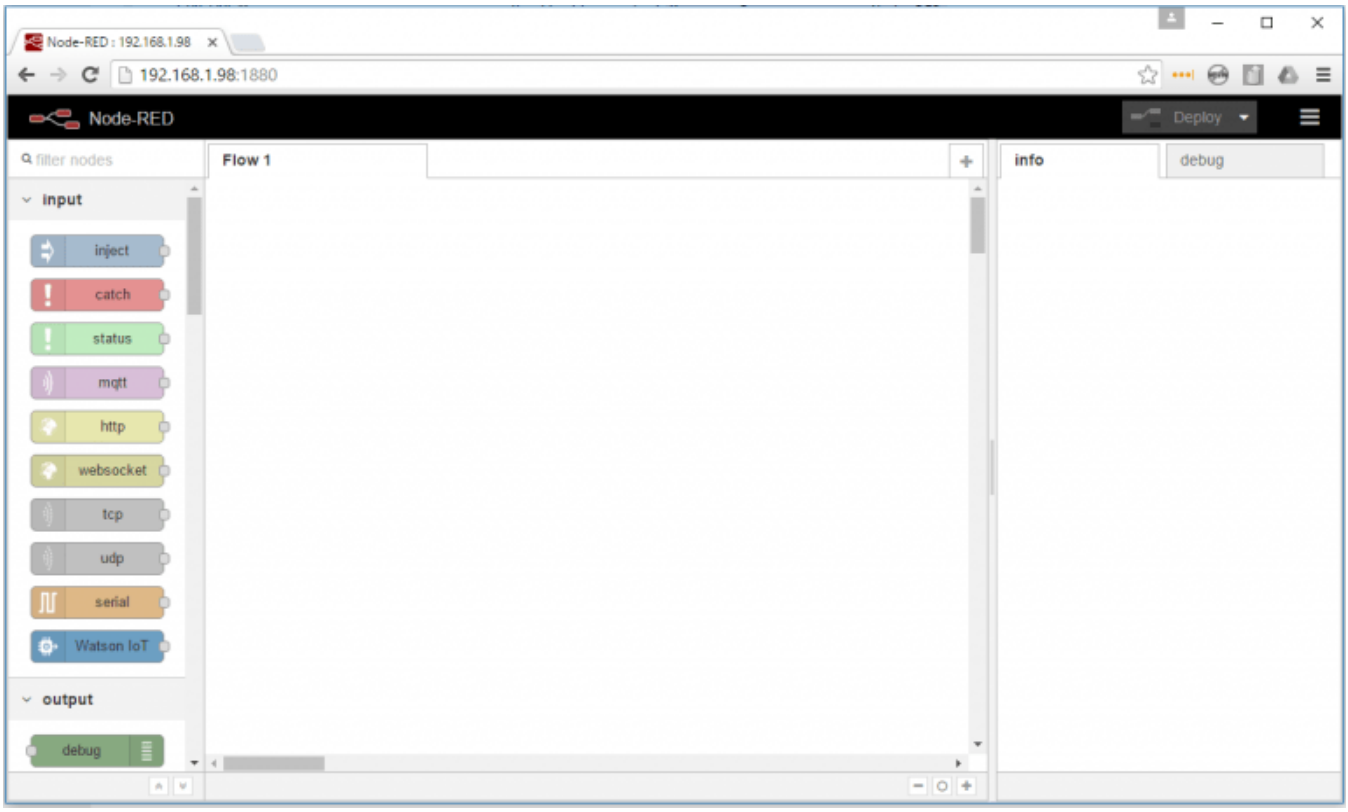
Now, restart your Pi so the autostart takes effect:

```
pi@raspberry:~ $ sudo reboot
```

Testing the Installation

When your Pi is back on, you can test the installation by entering the IP address of your Pi in a web browser followed by the **1880** port number:
http://YOUR_RPi_IP_ADDRESS:1880
In my case is:

http://192.168.1.98:1880
A page like this loads:



Now to load the Nodes you can go to **Menu** > **Import** and copy the following json file to your **Clipboard** to create your Node-RED flow.

Load the following node libraries by going to the Main Menu and then **Manage Palette** > **Install Tab** and load the following additional nodes:

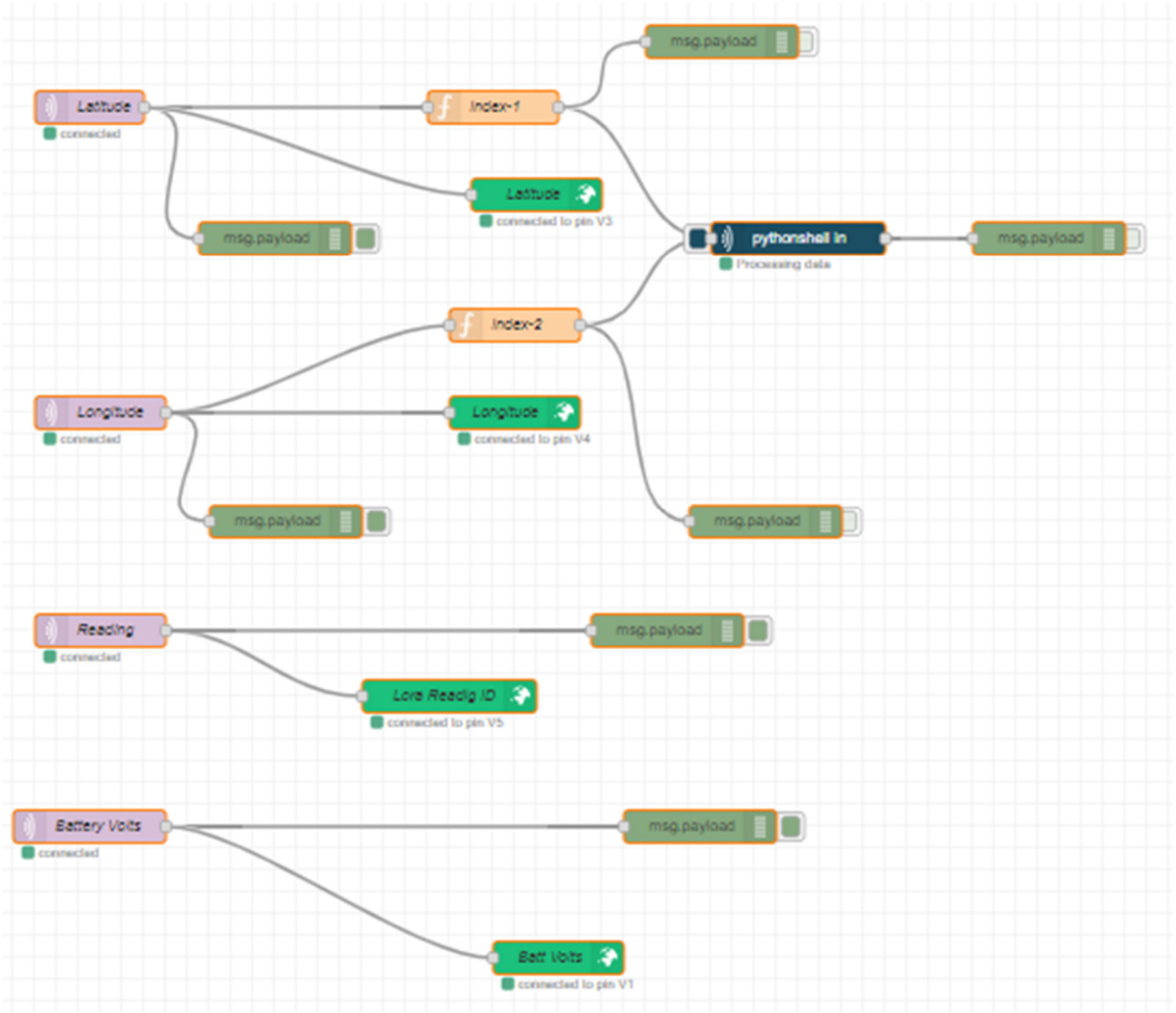
- node-red-contrib-blynk-ws
- node-red-contrib-pythonshell

Create the Lora Receiver MQTT Broker Server by importing or pasting in the follwing “json” code by going to the main menu and then selecting **Import** to a new flow.

```
[{"id":"4b106197.f097f","type":"tab","label":"MQTT
Mapping","disabled":false,"info":""},{"id":"87136311.25a3f","type":"mqtt
in","z":"4b106197.f097f","name":"Reading","topic":"esp32/reading","qos":"0
","datatype":"utf8","broker":"44555e9d.933f3","x":100,"y":580,"wires":[["2
c63d0b4.b1dd9","d4eea22f.7d0da"]]},{"id":"d4eea22f.7d0da","type":"debug","
z":"4b106197.f097f","name":"","active":true,"tosidebar":true,"console":fal
se,"tostatus":false,"complete":"payload","targetType":"msg","x":620,"y":58
0,"wires":[]},{"id":"49c66435.2d531c","type":"mqtt
in","z":"4b106197.f097f","name":"Latitude","topic":"esp32/latitude","qos":
"0","datatype":"utf8","broker":"44555e9d.933f3","x":90,"y":100,"wires":[["
41392935.3badc8","7a473223.d82f5c","71f3959b.3e41ec"]]},{"id":"4242d1bf.92
6f7","type":"mqtt
in","z":"4b106197.f097f","name":"Longitude","topic":"esp32/longitude","qos
":"0","datatype":"utf8","broker":"44555e9d.933f3","x":100,"y":380,"wires":
[["f48507ae.bc5538","242e993d.8ca0f6","fa075c0e.08ed5"]]},{"id":"c0fb2997.
3e3868","type":"mqtt in","z":"4b106197.f097f","name":"Battery
Volts","topic":"esp32/batteryvolts","qos":"0","datatype":"utf8","broker":"
44555e9d.933f3","x":90,"y":760,"wires":[["8702e46b.4b5168","214a0508.85442
a"]]},{"id":"8702e46b.4b5168","type":"debug","z":"4b106197.f097f","name":"
","active":true,"tosidebar":true,"console":false,"tostatus":false,"complet
e":"false","x":650,"y":760,"wires":[]},{"id":"214a0508.85442a","type":"bly
nk-ws-out-write","z":"4b106197.f097f","name":"Batt
Volts","pin":"1","pinmode":0,"client":"2a8a9326.5e1ffc","x":520,"y":880,"w
ires":[]},{"id":"2c63d0b4.b1dd9","type":"blynk-ws-out-
write","z":"4b106197.f097f","name":"Lora Readig
ID","pin":"5","pinmode":0,"client":"2a8a9326.5e1ffc","x":420,"y":640,"wire
s":[]},{"id":"71f3959b.3e41ec","type":"blynk-ws-out-
write","z":"4b106197.f097f","name":"Latitude","pin":"3","pinmode":0,"clien
t":"2a8a9326.5e1ffc","x":500,"y":180,"wires":[]},{"id":"fa075c0e.08ed5","t
ype":"blynk-ws-out-
write","z":"4b106197.f097f","name":"Longitude","pin":"4","pinmode":0,"clie
nt":"2a8a9326.5e1ffc","x":480,"y":380,"wires":[]},{"id":"abdfad5b.9c4a6","
type":"debug","z":"4b106197.f097f","name":"","active":false,"tosidebar":tr
ue,"console":false,"tostatus":false,"complete":"false","x":970,"y":220,"wi
res":[]},{"id":"c03191f1.e2dc7","type":"pythonshell
in","z":"4b106197.f097f","name":"","pyfile":"/home/pi/Blynk
Mapper.py","virtualenv":"","continuous":true,"stdInData":true,"x":740,"y":
220,"wires":[["abdfad5b.9c4a6"]]},{"id":"7a473223.d82f5c","type":"function
","z":"4b106197.f097f","name":"Index-1","func":"msg.payload = \"1:\" +
msg.payload;\nreturn
msg;","outputs":1,"noerr":0,"x":460,"y":100,"wires":[["c03191f1.e2dc7","db
59830a.84b4d"]]},{"id":"242e993d.8ca0f6","type":"function","z":"4b106197.f
097f","name":"Index-2","func":"msg.payload = \"2:\" + msg.payload;\nreturn
msg;","outputs":1,"noerr":0,"x":480,"y":300,"wires":[["c03191f1.e2dc7","15
76dbd4.ee05f4"]]},{"id":"db59830a.84b4d","type":"debug","z":"4b106197.f097
f","name":"","active":false,"tosidebar":true,"console":false,"tostatus":fa
lse,"complete":"false","x":670,"y":40,"wires":[]},{"id":"1576dbd4.ee05f4",
"type":"debug","z":"4b106197.f097f","name":"","active":false,"tosidebar":t
rue,"console":false,"tostatus":false,"complete":"false","x":710,"y":480,"w
ires":[]},{"id":"41392935.3badc8","type":"debug","z":"4b106197.f097f","nam
e":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"com
plete":"false","x":260,"y":220,"wires":[]},{"id":"f48507ae.bc5538","type":
"debug","z":"4b106197.f097f","name":"","active":true,"tosidebar":true,"con
sole":false,"tostatus":false,"complete":"false","x":270,"y":480,"wires":[]
```

```
},{"id":"44555e9d.933f3","type":"mqtt-
broker","z":"","name":"Pi_Server","broker":"localhost","port":"1883","clie
ntid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":
true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","clo
seQos":"0","closePayload":"","willTopic":"","willQos":"0","willPayload":""
},{"id":"2a8a9326.5e1ffc","type":"blynk-ws-
client","z":"","name":"Raspberry Pi MQTT Server","path":"ws://blynk-
cloud.com/websockets","key":"30hl3GOMmrR99j58hw8lcJz4tcMwMI6s","dbg_all":f
alse,"dbg_read":false,"dbg_write":false,"dbg_notify":false,"dbg_mail":fals
e,"dbg_prop":false,"dbg_sync":false,"dbg_bridge":false,"dbg_low":false,"db
g_pins":"","multi_cmd":false,"proxy_type":"no","proxy_url":"","enabled":tr
ue}]
```


After import is complete, you should see the following nodes appear:



Next create the following Python Script named “Blynk Mapper” and insert your Blynk Account API key into the “red line” and save it as a python file in your pi home directory:

```
""" Python Script to Run Blynk Mapping Function """

# Import Modules
# --------------
import requests
import sys
from time import sleep

# Global Variables
# ----------------
v="1"
lat= "43.30"
lon= "5.449"
msg = "Thor"

# Main Program:
# -------------
while (True):
    sleep(1)
    line = sys.stdin.readline().rstrip('\n').split(":")
    if line [0] == '1':
        lat = line [1]
        print("Latitude = " + lat)
    if line [0] == '2':
        lon = line [1]
        print("Longitude = " + lon)
    pay = {'value': [v , lat, lon, msg]}
    r=requests.get('http://blynk-cloud.com/your Blynk Account API
Key/update/V0',params=pay)
```

I added a MHS-3.5” TFT Screen and Project Case.

To load the driver for the display, copy the following information, and enter it into a command terminal:

```
sudo rm -rf LCD-show

git clone https://github.com/goodtft/LCD-show.git

chmod -R 755 LCD-show

cd LCD-show/

sudo ./MHS35-show
```

Here is what my Raspberry Pi Project looks like:

Note – you will also have to disable screen blanking and play with the default screen size to get all the icons on the screen. This is located under raspberry pi “appearance settings”.

Also, if you plan to use the touch screen, I would load Matchbox keyboard by typing the following from a terminal command line:

Start off by making sure you Raspberry Pi is up-to-date

sudo apt-get update
sudo apt-get upgrade

Now simply install the matchbox-keyboard package

sudo apt-get install matchbox-keyboard

Blynk Phone AP Instructions:

1) Sign up and create a free account with “Blynk” (Visit Website for instructions) and create a mobile app that looks like the picture enclosed. Special settings within the App are as follows:
    - Choose Raspberry Pi B+ as the device
    - Map Widget = V0
    - Latitude Gauge = V3
    - Longitude Gauge = V4
    - Lora Reading Gauge = V5
    - Battery Voltage Gauge = V1

Note - Choose any color you want for each widget

```
/*
*   This is a LORA GPS transmitter and receiver controlled by a button:
*     Menu choices are:
*     - Power Status of Module
*     - GPS coordinates
*     - LORA Transmit Function of GPS coordinates with battery condition
*     - LORA Receive Data Function
*
*    Note - Hold button for three seconds and release to place device in Sleep Mode
*           Press the power button to start up again.
*
*   Created by Roy Guerra
* */

// Libraries:
// ----------
#include <WiFi.h>
#include <Wire.h>
#include "axp20x.h"
#include <Button2.h>
#include "SSD1306.h"
#include <RadioLib.h>
#include <SPI.h>
#include <TinyGPS++.h>

// Program Definitions:
// --------------------
#ifndef AXP192_SLAVE_ADDRESS
#define AXP192_SLAVE_ADDRESS    0x34  // T-Beam V1 Onboard Power Management Hex Address
#endif
#define SSD1306_ADDRESS         0x3C  // OLED Display Hex Address
#define RADIO_TYPE      SX1262  // Choices are SX1262, SX1276, SX1278, etc.
#define LORA_SCK        5   // LORA Radio Interfaces
#define LORA_MISO       19
#define LORA_MOSI       27
#define LORA_SS         18
#define LORA_DI0        26
#define LORA_RST        23
#define LORA_DIO1       33
#define LORA_BUSY       32
#define GPS_BAUD_RATE   9600  // GPS Interfaces
#define GPS_RX_PIN  34
#define GPS_TX_PIN  12
#define BUTTON_PIN  38  // T-Beam Selection Button
#define BUTTON_PIN_MASK  GPIO_SEL_38  //0x4000000000   // Mask = (2^GPIO# +
2^GPIO#.....for each button, then convert to hex)
#define BAND  915.0 // LORA Radio Frequency
#define SF    7  // LORA Spreading Factor
#define BW    125.0  // LORA Bandwidth
#define TX_POWER    22 // LORA Transmit Power
#define SYNCH   0X12 // LORA Sync Word
#define SYMBOLS   8 // LORA Preamable Length
#define CODING_RATE   5 // LORA Coding Rate
#define I2C_SDA             21 // T-Beam I2C Communication Pins
#define I2C_SCL             22
#define PMU_IRQ             35
#define BOARD_LED    4

// Global Variables
// ----------------
uint8_t program = 0;
bool ssd1306_found = false;
bool axp192_found = false;
bool loraBeginOK = false;
uint64_t gpsSec = 0;
bool pmu_irq = false;
Button2 buttonA = Button2 (BUTTON_PIN);
String baChStatus = "Not charging";
String recv = "";
```

```
bool receivedFlag = false;
bool enableInterrupt = true;
int readingID = 0;
int count = 0;
String LoRaMessage = "";
float latitude = 0.000000;
float longitude = 0.000000;
int hours = 0;
int minutes = 0;
int seconds = 0;
int Num_Sats = 0;
float Batt_V = 0.0;
float Batt_I = 0.0;
float Batt_Disch = 0.0;
const unsigned long eventInterval_1 = 1500;  // 1.5 seconds
unsigned long previousTime_1 = 0;
const unsigned long eventInterval_2 = 3000;  // 3 seconds
unsigned long previousTime_2 = 0;

// Define Class Objects:
// ---------------------
AXP20X_Class axp;
TinyGPSPlus gps;
SSD1306  display(SSD1306_ADDRESS, I2C_SDA, I2C_SCL);
RADIO_TYPE radio = new Module(LORA_SS, LORA_DIO1, LORA_RST, LORA_BUSY);

// Function to initialize Button & Handlers:
// -----------------------------------------
void button_init(){
    buttonA.setChangedHandler(changed);
    buttonA.setTapHandler(tap);
    buttonA.setLongClickHandler(longpress);
}

// Button Handler Functions:
//--------------------------
void pressed(Button2& btn) {
    Serial.println("Button Pressed");
}
void released(Button2& btn) {
    Serial.print("Button Released: ");
    Serial.println(btn.wasPressedFor());
}
void longpress(Button2& btn) {
    unsigned int time = btn.wasPressedFor();
    if (time > 3000) {
        Serial.println("Button Long Handler Initiated, Going to Sleep");
        display.displayOn();
        display.clear();
        displayscreen(0, 0, 16, 0, "Going To Sleep"); // Write third line of Text
        through Function
        display.display();  // Write to Display Buffer
        delay(3000);  // 3 second delay
        display.displayOff();
        Serial.println("Go to Sleep");
        disablePeripherals(); // Goto Function
    }
}
void changed(Button2& btn) {
    Serial.println("Button Changed");
}
void tap(Button2& btn) {
    Serial.println("Button Tap");
    Serial.println("Count = " + String(count));
    count++;
    if (count == 3){
      Serial.print(F("[RADIO] Starting to listen ... "));
      int state = radio.startReceive();
        if (state == ERR_NONE) {
```

```
137               Serial.println(F("Radio Receive Success!"));
138               display.clear();  // Clear Display
139               displayscreen(0, 0, 16, 0, " Lora Receiver");
140               displayscreen(0, 50, 10, 0, "Radio Receive Initialize");
141               display.display();  // Write to Display Buffer
142           } else {
143               Serial.print(F("Radio Receive Failed, code "));
144               Serial.println(state);
145               display.clear();  // Clear Display
146               displayscreen(0, 0, 16, 0, " Lora Receiver");
147               displayscreen(0, 50, 10, 0, "Radio Receive Failure");
148               display.display();  // Write to Display Buffer
149               while (true);
150           }
151       }
152       if (count >= 4){
153         count = 0;
154       }
155   }
156
157   // Function to initialize OLED:
158   // ----------------------------
159   void startOLED(){
160     display.init(); // Initialize Display
161     display.displayOn();  // Turn Display On
162     display.clear();  // Clear Display
163     display.setContrast(255); // Set Display to full Brightness
164     display.flipScreenVertically();  // Set Display Orientation
165     display.clear();  // Clear Display
166     displayscreen(0, 0, 16, 0, "  T-Beam LORA"); // Write first line of Text through
      Function
167     displayscreen(0, 20, 10, 0, " GPS with Lora-Transmit"); // Write second line of Text
      through Function
168     displayscreen(0, 30, 10, 0, "and Lora-Receive Function"); // Write third line of Text
      through Function
169     displayscreen(0, 45, 10, 0, "  By Roy H Guerra Jr"); // Write third line of Text
      through Function
170     display.display();  // Write to Display Buffer
171     delay(3000);  // 3 second delay
172   }
173
174   // Function to Write Text on OLED Screen:
175   // --------------------------------------
176   void displayscreen(int x, int y, int font, int align, String(text)){
177     switch (align) {  // 0 = left, 1 = middle, 2 = right
178       case 0:
179         display.setTextAlignment(TEXT_ALIGN_LEFT);
180         break;
181       case 1:
182         display.setTextAlignment(TEXT_ALIGN_CENTER);
183         break;
184       case 2:
185         display.setTextAlignment(TEXT_ALIGN_RIGHT);
186         break;
187       default:
188         display.setTextAlignment(TEXT_ALIGN_LEFT);
189       break;
190     }
191     switch (font) {  // Choices are 10, 16, 24; (ArielMT_Font Size)
192       case 10:
193         display.setFont(ArialMT_Plain_10); // Size 10 font
194         break;
195       case 16:
196         display.setFont(ArialMT_Plain_16); // Size 16 font
197         break;
198       case 24:
199         display.setFont(ArialMT_Plain_24); // Size 24 font
200         break;
201       default:
```

```
      display.setFont(ArialMT_Plain_16); // Size 16 font
    break;
  }
  display.drawString(x, y, text); // Write text at the x-y cursor position
}

// Function to get Power Status:
// -----------------------------
void Bat_Stat(){
    if (axp.isBatteryConnect()) {
        Batt_V = axp.getBattVoltage() / 1000.0;
        Batt_I = axp.getBattDischargeCurrent();
     }
        if (axp.isChargeing()){
          Batt_Disch = axp.getBattChargeCurrent();
        }
        else {
          Batt_Disch = 0.00;
        }
}

// Function is called when a complete packet is received by the module:
// ---------------------------------------------------------------------
void setFlag(void)
{
    // check if the interrupt is enabled
    if (!enableInterrupt) {
        return;
    }
    // we got a packet, set the flag
    receivedFlag = true;
}

// Function to initialize LORA Radio:
// ----------------------------------
void lora_init()
{
    SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI, LORA_SS);
    Serial.print(F("[Radio] Initializing ... "));
    int state = radio.begin(BAND, BW, SF, CODING_RATE, SYNCH, TX_POWER, SYMBOLS);
    if (state == ERR_NONE) {
        loraBeginOK = true;
        Serial.println(F("success!"));
    } else {
        Serial.print(F("failed, code "));
        Serial.println(state);
        while (true);
    }

    // set the function that will be called
    // when new packet is received
    radio.setDio1Action(setFlag);
}

// Function to Read GPS:
// ---------------------
void GPS_Read() {
  while (Serial1.available() > 0) {
      gps.encode(Serial1.read());
  }
       hours = gps.time.hour();
       minutes = gps.time.minute();
       seconds = gps.time.second();
       Num_Sats = gps.satellites.value();
       latitude = gps.location.lat();
       longitude = gps.location.lng();
       Serial.print("Latitude= ");
       Serial.println(latitude);
       Serial.print("Longitude= ");
```

```
        Serial.println(longitude);
        Serial.print("UTC Time= ");
        Serial.print(hours);
        Serial.print(minutes);
        Serial.println(seconds);
        Serial.print("Number of Satellites = ");
        Serial.println(Num_Sats);
}

// Function to put ESP32 in Sleep Mode:
// ------------------------------------
void disablePeripherals() {
  int ret;
  do {
        // In order to ensure that it is set correctly,
        // the loop waits for it to return the correct return value
        Serial.println("Set AXP192 in sleep mode");
        ret = axp.setSleep();
        delay(500);
    } while (ret != AXP_PASS) ;
    // Turn off all power channels, only use PEK or AXP GPIO to wake up
    // After setting AXP202/AXP192 to sleep,
    // it will start to record the status of the power channel that was turned off after setting,
    // it will restore the previously set state after PEK button or GPIO wake up
    // Turn off all AXP192 power channels
    ret = axp.setPowerOutPut(AXP192_LDO2, AXP202_OFF);
    Serial.printf("Set Power AXP192_LDO2:%s\n", ret == AXP_PASS ? "OK" : "FAIL");
    ret = axp.setPowerOutPut(AXP192_LDO3, AXP202_OFF);
    Serial.printf("Set Power AXP192_LDO3:%s\n", ret == AXP_PASS ? "OK" : "FAIL");
    ret = axp.setPowerOutPut(AXP192_DCDC1, AXP202_OFF);
    Serial.printf("Set Power AXP192_DCDC1:%s\n", ret == AXP_PASS ? "OK" : "FAIL");
    ret = axp.setPowerOutPut(AXP192_DCDC2, AXP202_OFF);
    Serial.printf("Set Power AXP192_DCDC2:%s\n", ret == AXP_PASS ? "OK" : "FAIL");
    ret = axp.setPowerOutPut(AXP192_EXTEN, AXP202_OFF);
    Serial.printf("Set Power AXP192_EXTEN:%s\n", ret == AXP_PASS ? "OK" : "FAIL");
    Serial.flush();
    // Uncomment the following delay line below to view status live time.
    //delay(1000);
    // Tbeam v1.0 uses DC3 as the MCU power channel, turning it off as the last
    ret = axp.setPowerOutPut(AXP192_DCDC3, AXP202_OFF);
    Serial.printf("Set Power AXP192_DCDC3:%s\n", ret == AXP_PASS ? "OK" : "FAIL");

    // Turn off all AXP202 power channels
    // axp.setPowerOutPut(AXP202_LDO2, AXP202_OFF);
    // axp.setPowerOutPut(AXP202_LDO3, AXP202_OFF);
    // axp.setPowerOutPut(AXP202_LDO4, AXP202_OFF);
    // axp.setPowerOutPut(AXP202_DCDC2, AXP202_OFF);
    // axp.setPowerOutPut(AXP202_DCDC3, AXP202_OFF);
    // axp.setPowerOutPut(AXP202_EXTEN, AXP202_OFF);

    // If you set the power supply to sleep mode and you turn off the power supply of the MCU,
    // you will not be able to use the wake-up mode provided by the MCU.
    // If you do not turn off the power of the MCU, you can continue to use it
    delay(20);
    esp_sleep_enable_ext1_wakeup(BUTTON_PIN_MASK, ESP_EXT1_WAKEUP_ALL_LOW);
    esp_deep_sleep_start();
}

// Main Program:
// =============
void setup(){
    Serial.begin(115200);
    delay(1000);
    Wire.begin(I2C_SDA, I2C_SCL);
       if (axp.begin(Wire, AXP192_SLAVE_ADDRESS) != AXP_FAIL){
                axp192_found = true;
                Serial.println("AXP192 Begin PASS");
```

```
338         }
339         else {
340             Serial.println("AXP192 Begin FAIL");
341         }
342         // axp.setChgLEDMode(LED_BLINK_4HZ);
343 
344         axp.setDCDC1Voltage(3300);  // VDD 3v3 for OLED
345         axp.setLDO2Voltage(3300);   //LORA VDD set 3v3
346         axp.setLDO3Voltage(3300);   //GPS VDD      3v3
347         axp.setPowerOutPut(AXP192_LDO2, AXP202_ON);
348         axp.setPowerOutPut(AXP192_LDO3, AXP202_ON);
349         axp.setPowerOutPut(AXP192_DCDC2, AXP202_ON);
350         axp.setPowerOutPut(AXP192_EXTEN, AXP202_ON);
351         axp.setPowerOutPut(AXP192_DCDC1, AXP202_ON);
352         Serial.printf("DCDC1: %s\n", axp.isDCDC1Enable() ? "ENABLE" : "DISABLE");
353         Serial.printf("DCDC2: %s\n", axp.isDCDC2Enable() ? "ENABLE" : "DISABLE");
354         Serial.printf("LDO2: %s\n", axp.isLDO2Enable() ? "ENABLE" : "DISABLE");
355         Serial.printf("LDO3: %s\n", axp.isLDO3Enable() ? "ENABLE" : "DISABLE");
356         Serial.printf("DCDC3: %s\n", axp.isDCDC3Enable() ? "ENABLE" : "DISABLE");
357         Serial.printf("Exten: %s\n", axp.isExtenEnable() ? "ENABLE" : "DISABLE");
358         Serial.println("----------------------------------------");
359         pinMode(PMU_IRQ, INPUT_PULLUP);
360         attachInterrupt(PMU_IRQ, [] {
361           pmu_irq = true;
362         }, FALLING);
363         axp.adc1Enable(AXP202_VBUS_VOL_ADC1 |
364                        AXP202_VBUS_CUR_ADC1 |
365                        AXP202_BATT_CUR_ADC1 |
366                        AXP202_BATT_VOL_ADC1,
367                        AXP202_ON);
368 
369         axp.enableIRQ(AXP202_VBUS_REMOVED_IRQ |
370                       AXP202_VBUS_CONNECT_IRQ |
371                       AXP202_BATT_REMOVED_IRQ |
372                       AXP202_BATT_CONNECT_IRQ,
373                       AXP202_ON);
374 
375         //axp.adc1Enable(AXP202_BATT_CUR_ADC1, 1);
376         //axp.enableIRQ(AXP202_VBUS_REMOVED_IRQ | AXP202_VBUS_CONNECT_IRQ |
            AXP202_BATT_REMOVED_IRQ | AXP202_BATT_CONNECT_IRQ, 1);
377         axp.clearIRQ();
378         if (axp.isChargeing()) {
379             baChStatus = "Charging";
380         }
381     button_init();
382     Serial1.begin(GPS_BAUD_RATE, SERIAL_8N1, GPS_RX_PIN, GPS_TX_PIN);
383     startOLED();
384     lora_init();
385 }
386 
387 void loop(){
388     display.displayOn();
389     display.clear();  // Clear Display
390     unsigned long currentTime_1 = millis();  // Get a timestamp
391     unsigned long currentTime_2 = millis();  // Get a timestamp
392     buttonA.loop();
393     if (axp192_found && pmu_irq) {
394         pmu_irq = false;
395         axp.readIRQ();
396         if (axp.isChargingIRQ()) {
397             baChStatus = "Charging";
398         } else {
399             baChStatus = "Not Charging";
400         }
401         if (axp.isVbusRemoveIRQ()) {
402             baChStatus = "Not Charging";
403         }
404         //digitalWrite(2, !digitalRead(2));
405         axp.clearIRQ();
```

```
406       }
407       switch (count) {
408         case 0:
409             Bat_Stat(); // Goto Function
410             displayscreen(0, 0, 16, 0, " Power Status:");
411             displayscreen(0, 20, 10, 0, "Batt Volts = " + String(Batt_V) + " Vdc" );
412             displayscreen(0, 30, 10, 0, "Batt Disch. = " + String(Batt_I) + " mA");
413             displayscreen(0, 40, 10, 0, "Batt Charge = " + String(Batt_Disch) + "
                mA");
414             displayscreen(0, 50, 10, 0, "Batt Mode = " + baChStatus);
415             display.display();  // Write to Display Buffer
416             Serial.println("Batt Volts = " + String(Batt_V) + " Vdc");
417             Serial.println("Batt Discharge = " + String(Batt_I) + " mA");
418             Serial.println("Batt Charge = " + String(Batt_Disch) + " mA");
419             Serial.println("Batt Mode = " + baChStatus);
420             delay(10); // 10mS delay
421           break;
422         case 1:
423             displayscreen(0, 0, 16, 0, "T-Beam GPS");  // First line in Menu
424             display.display();  // Write to Display Buffer
425             GPS_Read(); // Goto Function
426             if (Num_Sats < 2) {
427                 displayscreen(0, 0, 16, 0, "T-Beam GPS");
428                 displayscreen(0, 35, 16, 0, "No GPS Detected");  // First line in Menu
429                 display.display();  // Write to Display Buffer
430              } else {
431                  displayscreen(0, 0, 16, 0, "T-Beam GPS");
432                  displayscreen(0, 20, 10, 0, "Latitude = " + String(latitude,5)); // 5
                     decimal places
433                  displayscreen(0, 30, 10, 0, "Longitude = " + String(longitude,5)); // 5
                     decimal places
434                  displayscreen(0, 40, 10, 0, "UTC Time = " + String(hours) + ":" +
                     String(minutes) + ":" + String(seconds));
435                  displayscreen(0, 50, 10, 0, "Number of Satellites = " +
                     String(Num_Sats));
436                  display.display();  // Write to Display Buffer
437              }
438           break;
439         case 2:
440              displayscreen(0, 0, 16, 0, "  Lora Sender");  // First line in Menu
441              display.display();  // Write to Display Buffer
442              if (currentTime_2 - previousTime_2 >= eventInterval_2) {
443                GPS_Read(); // Goto Function
444                Bat_Stat(); // Goto Function
445                LoRaMessage = String(readingID) + "/" + String(latitude,6) + "&" +
                   String(longitude,6) + "#" + String(Batt_V); // LoRa packet to receiver
446                int transmissionState = ERR_NONE;
447                transmissionState = radio.startTransmit(LoRaMessage);
448                // check if the previous transmission finished
449               if (receivedFlag) {  // disable the interrupt service routine while
                  processing the data
450                   enableInterrupt = false;
451                   receivedFlag = false; // reset flag
452                   if (transmissionState == ERR_NONE) { // packet was successfully sent
453                       Serial.println(F("transmission finished!"));
454                       displayscreen(0, 0, 16, 0, "  Lora Sender");
455                       displayscreen(0, 20, 10, 0, "Packet Number = " +
                          String(readingID));
456                       displayscreen(0, 30, 10, 0, "UTC Time = " + String(hours) + ":" +
                          String(minutes) + ":" + String(seconds));
457                       displayscreen(0, 40, 10, 0, "Status = Complete");
458                       display.display();  // Write to Display Buffer
459                       readingID++; // Increment Reading
460                       delay(2000);
461                   } else {
462                       Serial.print(F("failed, code "));
463                       Serial.println(transmissionState);
464                       display.clear();  // Clear Display
465                       displayscreen(0, 0, 16, 0, "  Lora Sender");
```

```
                displayscreen(0, 40, 10, 0, "Status = Failed");
                display.display();  // Write to Display Buffer
              }

              /* you can transmit C-string or Arduino string up to 256 characters long

                you can also transmit byte array up to 256 bytes long
                                byte byteArr[] = {0x01, 0x23, 0x45, 0x67,
                                  0x89, 0xAB, 0xCD, 0xEF};
                int state = radio.startTransmit(byteArr, 8);
              */

              enableInterrupt = true; // enable interrupt service routine
            }
            previousTime_2 = currentTime_2;  // Store the
            timestamp
        }
        break;
    case 3:
        displayscreen(0, 0, 16, 0, " Lora Receiver");  // First line in Menu
        display.display();  // Write to Display Buffer
        if (receivedFlag) { // check if the flag is set.  Disable the interrupt service
        routine while processing the data
            enableInterrupt = false;
            receivedFlag = false; // reset flag
            String str;
            int state1 = radio.readData(str); // you can read received data as an
            Arduino String
            /*
             * // you can also read received data as byte array
              byte byteArr[8];
              int state = radio.readData(byteArr, 8);
            */
            if (state1 == ERR_NONE) {
                Serial.println(F("[RADIO] Received packet!")); // packet was
                successfully received print data of the packet
                Serial.print(F("[RADIO] Data:\t\t"));
                Serial.println(str);
                Serial.print(F("[RADIO] RSSI:\t\t"));
                Serial.print(radio.getRSSI());
                Serial.println(F(" dBm"));
                Serial.print(F("[RADIO] SNR:\t\t"));
                Serial.print(radio.getSNR());
                Serial.println(F(" dB"));
                display.clear();  // Clear Display
                displayscreen(0, 0, 16, 0, " Lora Receiver");
                displayscreen(0, 20, 10, 0, "Data: " + str);
                displayscreen(0, 30, 10, 0, "RSSI = " + String(radio.getRSSI()) + "
                dBm");
                displayscreen(0, 40, 10, 0, "SNR =" + String(radio.getSNR()) + " dB");
                displayscreen(0, 50, 10, 0, "Status = No Errors");
                display.display();  // Write to Display Buffer
            } else if (state1 == ERR_CRC_MISMATCH) {
                // packet was received, but is malformed
                Serial.println(F("CRC error!"));
                display.clear();  // Clear Display
                displayscreen(0, 0, 16, 0, " Lora Receiver");
                displayscreen(0, 50, 10, 0, "Status = CRC Error");
                display.display();  // Write to Display Buffer
            } else {
                // some other error occurred
                Serial.print(F("failed, code "));
                Serial.println(state1);
                display.clear();  // Clear Display
                displayscreen(0, 0, 16, 0, " Lora Receiver");
                displayscreen(0, 50, 10, 0, "Status = Failed");
                display.display();  // Write to Display Buffer
            }
            delay(1000);
```

```
530             radio.startReceive();  // put module back to listen mode we're ready to
                receive more packets
531             enableInterrupt = true; // enable interrupt service routine
532         }
533         break;
534     }
535 }
```

```
/*********************************************************************************
 LORA 915 MHz Sender (Heltec Wiffi-32 LORA Board V2)
 - Uses SX1276 chip based on ESP32 WIFI with OLED
 - Burn with Heltec WiFi Lora 32(V2)
 - Must load boards library and support libraries first!!!!!

 GPS Pin Mapping:
 ----------------
  The Neo7 module GND pin is connected to ESP32 GND pin
  The Neo7 module RX pin is connected to ESP32 pin 12
  The Neo7 module TX pin is connected to ESP32 pin 13
  The Neo7 module VCC pin is connected to ESP32 3.3V pin

 Batt Volts:
 -----------
  The Battery is tied to a high impedance voltage divider (3.3v FS)
 and connected to ESP32 GND pin #38

*********************************************************************************/

// Libraries for LoRa
// ------------------
#include <SPI.h>
#include <LoRa.h>

// Libraries for GPS
// -----------------
#include <TinyGPS++.h>

// Libraries for OLED Display
// --------------------------
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>


// Define the pins used by the LoRa transceiver module
// ---------------------------------------------------
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26
#define DIO1 35
#define DIO2 34

// Define GPS Pins:
// ----------------
#define GPS_RX_PIN   13
#define GPS_TX_PIN   12
#define SerialGPS    Serial1

// Choose frequency (uncomment):
// -----------------------------
//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 915E6

// Display OLED Pins & Screen:
// ---------------------------
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Global Variables:
```

```
// -----------------
int readingID = 0;
int counter = 0;
String LoRaMessage = "";
float latitude = 0.000000;
float longitude = 0.000000;
float batt = 0.00;
const int Analog_channel_pin= 36;
int ADC_VALUE = 0;
const float V_Scale = 5.0;  // Full Scale Scaling Factor

// Create Class Objects:
// ---------------------
TinyGPSPlus gps;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);

// Initialize OLED Display
// -----------------------
void startOLED(){
  //reset OLED display via software
  pinMode(25, OUTPUT); // on board LED
  digitalWrite(25, LOW);  // LED Off
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  //initialize OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Address 0x3C for
  128x32
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("LORA SENDER");
  display.setCursor(0,20);
  display.print("With GPS");
  display.setCursor(0,30);
  display.print("By Roy H Guerra Jr");
  display.display();
  delay(2000);
}

// Initialize LoRa Module:
// -----------------------
void startLoRA(){
  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);

  while (!LoRa.begin(BAND) && counter < 10) {
    Serial.print(".");
    counter++;
    delay(500);
  }
  if (counter == 10) {
    // Increment readingID on every new reading
    readingID++;
    Serial.println("Starting LoRa failed!");
  }
  Serial.println("LoRa Initialization OK!");
  display.setCursor(0,10);
  display.clearDisplay();
  display.print("LoRa Initializing OK!");
```

```
  display.display();
  delay(2000);
}

// Function to Read GPS:
// ---------------------
void readGPS(){
  while (SerialGPS.available() > 0) {
     gps.encode(SerialGPS.read());
     }
      latitude = gps.location.lat();
      longitude = gps.location.lng();
      Serial.print("Latitude= ");
      Serial.println(latitude);
      Serial.print("Longitude= ");
      Serial.println(longitude);
}

// Function to read Battery Voltage:
// ---------------------------------
void batt_volts(){
  ADC_VALUE = analogRead(Analog_channel_pin);
  Serial.print("ADC VALUE = ");
  Serial.println(ADC_VALUE);
  delay(100);
  batt = (ADC_VALUE * V_Scale) / 4095.0;  // Maximum input is 3.3vots, scale to 5 volts
  Serial.print("Voltage = ");
  Serial.print(batt);
  Serial.println(" volts");
}

// Function to Send LORA Readings:
// -------------------------------
void sendReadings() {
  LoRaMessage = String(readingID) + "/" + String(latitude,6) + "&" +
  String(longitude,6) + "#" + String(batt);
  //Send LoRa packet to receiver
  LoRa.beginPacket();
  /*
   * LoRa.setTxPower(txPower,RFOUT_pin);
   * txPower -- 0 ~ 20
   * RFOUT_pin could be RF_PACONFIG_PASELECT_PABOOST or RF_PACONFIG_PASELECT_RFO
   *   - RF_PACONFIG_PASELECT_PABOOST -- LoRa single output via PABOOST, maximum output
   20dBm
   *   - RF_PACONFIG_PASELECT_RFO     -- LoRa single output via RFO_HF / RFO_LF,
   maximum output 14dBm
  */
  //LoRa.setTxPower(20,RF_PACONFIG_PASELECT_PABOOST); // Boost output power
  LoRa.setTxPower(20);  // Boost output power
  LoRa.print(LoRaMessage);
  LoRa.endPacket();
  display.clearDisplay();
  display.setCursor(0,0);
  display.setTextSize(1);
  display.print("LoRa packet sent!");
  display.setCursor(0,20);
  display.print("Latitude = ");
  display.setCursor(72,20);
  display.print(String(latitude,5));  // Keep String data to 5 decimal places.
  display.setCursor(0,30);
  display.print("Longitude = ");
  display.setCursor(72,30);
  display.print(String(longitude,5)); // Keep String data to 5 decimal places.
  display.setCursor(0,40);
  display.print("Battery = ");
  display.setCursor(58,40);
  display.print(batt);
  display.setCursor(0,50);
  display.print("Reading ID:");
```

```
  display.setCursor(66,50);
  display.print(readingID);
  display.display();
  Serial.print("Sending packet: ");
  Serial.println(readingID);
  digitalWrite(25, HIGH);  // LED
  delay(1000);
  digitalWrite(25, LOW);  // LED
  readingID++;
  display.clearDisplay();
  display.setTextColor(BLACK);
  display.display();
  display.setTextColor(WHITE);
}

// Main Program:
// =============
void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);
  SerialGPS.begin(9600, SERIAL_8N1, GPS_RX_PIN, GPS_TX_PIN);
  startOLED();
  startLoRA();
}

void loop() {
  batt_volts();
  readGPS();
  sendReadings();
  delay(5000);
}
```

```
/*********************************************************************************
 LORA 915 MHz Sender (TTGO Wiffi-32 LORA Board V2)
 - Uses SX1276 chip based on ESP32 WIFI with OLED
 - Burn with TTGO Lora32 OLED V1
 - Must load boards library and support libraries first!!!!!

 Created by Roy H Guerra Jr
*********************************************************************************/
// Wi-Fi & MQTT libraries:
// -----------------------
#include "WiFiConnectOLED.h" //include before SSD1306.h if using custom fonts
#include <Wire.h>
#include "SSD1306.h"
#include <WiFi.h>
#include <WebServer.h>
#include <PubSubClient.h>

#ifdef ESP32
#define OLED_RESET 16
#else
#define OLED_RESET 10
#endif

#ifdef ESP32
SSD1306  display(0x3c, 4, 15);
#define oledPWR 16 //Pin to supply power to OLED
#else
SSD1306  display(0x3c, 4, 5);
#define oledPWR 10 //Pin to supply power to OLED
#endif

WiFiConnectOLED wc(&display, oledPWR); //Initialise our connector with an OLED display

// Libraries for LoRa
// ------------------
#include <SPI.h>
#include <LoRa.h>

// Define the pins used by the LoRa transceiver module
// ---------------------------------------------------
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

// Choose frequency (uncomment):
// -----------------------------
//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 915E6

// WIFI LED Pin
// ------------
#define WLED 2

// Initialize Global variables and save Data:
// ------------------------------------------
unsigned long previousMillis = 0;   // Stores last time data was published
const long interval = 5000;        // Interval at which to publish sensor readings
int rssi;
String loRaMessage;
String latitude;
String longitude;
String batt;
String readingID;
long lastReconnectAttempt = 0;
```

```
char msg_1[50];
char msg_2[50];
char msg_3[50];
char msg_4[50];

// Add your MQTT Broker IP address, example:
//const char* mqtt_server = "192.168.1.144";
const char* mqtt_server = "192.168.50.68";

// Initializes the espClient & MQTT:
// ---------------------------------
WiFiClient espClient;
PubSubClient client(espClient);

// Access Point Callback Function:
// -------------------------------
void configModeCallback(WiFiConnect *mWiFiConnect) {
  Serial.println("Entering Access Point");
}

// WiFi Connection Function;
// -------------------------
void startWiFi(boolean showParams = false) {
  wc.begin(true);
  wc.setDebug(true);
  /* Set our callbacks */
  wc.setAPCallback(configModeCallback);
  //wc.screenTest(); //test screen by cycling through the presete screens
  //wc.resetSettings(); //helper to remove the stored wifi connection, comment out after first upload and re upload
    /*
       AP_NONE = Continue executing code
       AP_LOOP = Trap in a continuous loop
       AP_RESET = Restart the chip
    */
    if (!wc.autoConnect()) { // try to connect to wifi
      /* We could also use button etc. to trigger the portal on demand within main loop */
      wc.startConfigurationPortal(AP_LOOP);//if not connected show the configuration portal
    }
  // when displayLoop is called from main loop, will turn of display after time period
  wc.displayTurnOFF((60 * 1000 * 10));  // 10 minutes
}

// Initialize OLED display:
// ------------------------
void startOLED(){
  display.displayOn();  // Turn Display On
  display.setContrast(255); // Set Display to full Brightness
  display.flipScreenVertically();  // Set Display Orientation
  display.clear();  // Clear Display
  displayscreen(0, 0, 16, 0, "Lora TTGO Receiv"); // Write first line of Text through Function
  displayscreen(0, 20, 16, 0, "Inc MQTT Publish"); // Write second line of Text through Function
  displayscreen(0, 40, 16, 0, "By Roy Guerra Jr"); // Write third line of Text through Function
  display.display();  // Write to Display Buffer
  delay(3000);  // 3 second delay
  blankscreen();  // Goto Screen Blanking Function
}

// Function to Turn Off OLED Screen:
// ---------------------------------
void blankscreen(){
  display.clear();  // Clear Display
  display.display();  // Write to Display Buffer
  display.displayOff(); // Switch display off
```

```
}

// Function to Write Text on OLED Screen:
// --------------------------------------
void displayscreen(int x, int y, int font, int align, String(text)){
  switch (align) {  // 0 = left, 1 = middle, 2 = right
    case 0:
      display.setTextAlignment(TEXT_ALIGN_LEFT);
      break;
    case 1:
      display.setTextAlignment(TEXT_ALIGN_CENTER);
      break;
    case 2:
      display.setTextAlignment(TEXT_ALIGN_RIGHT);
      break;
    default:
      display.setTextAlignment(TEXT_ALIGN_LEFT);
    break;
  }
  switch (font) {  // Choices are 10, 12, 16, 24; (Roboto_Font Size)
    case 10:
      display.setFont(Roboto_10); // Size 10 font
      break;
    case 12:
      display.setFont(Roboto_12); // Size 12 font
      break;
    case 16:
      display.setFont(Roboto_16); // Size 16 font
      break;
    case 24:
      display.setFont(Roboto_24); // Size 24 font
      break;
    default:
      display.setFont(Roboto_16); // Size 16 font
    break;
  }
  display.drawString(x, y, text); // Write text at the x-y cursor position
}

// Initialize LoRa module:
// -----------------------
void startLoRA(){
  int counter;
  //SPI LoRa pins
  SPI.begin(SCK, MISO, MOSI, SS);
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);
  while (!LoRa.begin(BAND) && counter < 10) {
    Serial.print(".");
    counter++;
    delay(500);
  }
  if (counter == 10) {
    // Increment readingID on every new reading
    Serial.println("Starting LoRa failed!");
  }
  Serial.println("LoRa Initialization OK!");
  display.displayOn();  // Turn Display On
  display.clear();  // Clear Display
  displayscreen(0, 0, 12, 0, "LoRa Initializing OK!"); // Write first line of Text
  through Function
  display.display();  // Write to Display Buffer
  delay(2000);
}

// Function to Connect WiFi:
// -------------------------
void connectWiFi(){
  startWiFi();
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite (WLED, LOW); // set GPIO2 low to keep off LED
    Serial.print(".");
  }
  digitalWrite (WLED, HIGH); // set GPIO2 high to keep on LED
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Read LoRa packet and get the sensor readings:
// ---------------------------------------------
void getLoRaData() {
  Serial.print("Lora packet received: ");
  // Read packet
  while (LoRa.available()) {
    String LoRaData = LoRa.readString();
    // Example LoRaData format: readingID/temperature&soilMoisture#batterylevel
    // String example: 1/27.43&654#95.34
    Serial.println(LoRaData);

    // Get readingID, temperature and soil moisture
    int pos1 = LoRaData.indexOf('/');
    int pos2 = LoRaData.indexOf('&');
    int pos3 = LoRaData.indexOf('#');
    readingID = LoRaData.substring(0, pos1);
    latitude = LoRaData.substring(pos1 +1, pos2);
    longitude = LoRaData.substring(pos2+1, pos3);
    batt = LoRaData.substring(pos3+1, LoRaData.length());
  }
  // Get RSSI
  rssi = LoRa.packetRssi();
  Serial.print(" with RSSI ");
  Serial.println(rssi);
  display.displayOn();  // Turn Display On
  display.clear();  // Clear Display
  displayscreen(0, 0, 10, 0, "        Message Received");
  displayscreen(0, 10, 10, 0, "- RSSI: " + String(rssi) + " dBm");
  displayscreen(0, 20, 10, 0, "- Packet # : " + readingID);
  displayscreen(0, 30, 10, 0, "- Latitude = " + latitude);
  displayscreen(0, 40, 10, 0, "- Longitude = " + longitude);
  displayscreen(0, 50, 10, 0, "- Batt Volts = " + batt);
  display.display();  // Write to Display Buffer
  delay(2000);
  blankscreen();  // Goto Screen Blanking Function
}

// Callback Function:
// ------------------
void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();

  // Feel free to add more if statements to control more GPIOs with MQTT

  // If a message is received on the topic esp32/output, you check if the message is
  either "on" or "off".
  // Changes the output state according to the message
```

```
269     if (String(topic) == "esp32/output") {
270       Serial.print("Changing output to ");
271       if(messageTemp == "on"){
272         Serial.println("on");
273        // digitalWrite(ledPin, HIGH);
274       }
275       else if(messageTemp == "off"){
276         Serial.println("off");
277         //digitalWrite(ledPin, LOW);
278       }
279        if ((char)message[0] == 'O' && (char)message[1] == 'N') //on
280       {
281      // digitalWrite(LED, HIGH);
282       Serial.println("on");
283       client.publish("outTopic", "LED turned ON");
284       }
285      else if ((char)message[0] == 'O' && (char)message[1] == 'F' && (char)message[2] ==
      'F') //off
286       {
287      // digitalWrite(LED, LOW);
288       Serial.println(" off");
289       client.publish("outTopic", "LED turned OFF");
290       }
291     Serial.println();
292     }
293   }
294
295   // Function to Re-Connect:
296   // -----------------------
297   void reconnect() {
298     // Loop until we're reconnected
299     while (!client.connected()) {
300       Serial.print("Attempting MQTT connection...");
301       // Attempt to connect
302       if (client.connect("ESP32_clientID")) {
303         Serial.println("connected");
304         // Subscribe
305         //client.subscribe("esp32/output");
306       } else {
307         Serial.print("failed, rc=");
308         Serial.print(client.state());
309         Serial.println(" try again in 5 seconds");
310         // Wait 5 seconds before retrying
311         delay(5000);
312       }
313     }
314   }
315
316   // MQTT Connection Function:
317   // -------------------------
318   void connectmqtt(){
319     client.connect("ESP32_clientID");  // ESP will connect to mqtt broker with clientID
320     {
321       Serial.println("connected to MQTT");
322       if (!client.connected()) {
323         reconnect();
324       }
325     }
326   }
327
328   // Main Program:
329   // =============
330   void setup() {
331     // Initialize Serial Monitor
332     Serial.begin(115200);
333     pinMode (WLED, OUTPUT);
334     digitalWrite (WLED, LOW); // set GPIO25 low to keep off LED
335     connectWiFi();
336     startOLED();
```

```
337     startLoRA();
338    client.setServer(mqtt_server, 1883);  // Uncomment
339     //client.setCallback(callback);  // Uncomment for suscribe, not publish
340    connectmqtt();
341   }
342
343   void loop() {
344     // Check if there are LoRa packets available
345      if (WiFi.status() != WL_CONNECTED) {
346       digitalWrite (WLED, LOW); // set GPIO25 low to keep off LED
347       if (!wc.autoConnect()) wc.startConfigurationPortal(AP_RESET);
348     }
349     else {
350       digitalWrite (WLED, HIGH); // set GPIO25 high to keep on LED
351     }
352     int packetSize = LoRa.parsePacket();
353     if (packetSize) {
354       getLoRaData();
355     }
356
357     if (!client.connected()) {
358       reconnect();
359     }
360     client.loop();
361     //if(!client.loop()) {
362      // client.connect("ESP32Client");
363    // }
364    readingID.toCharArray(msg_1,readingID.length()+1);  // Convert string to a Character
       Array
365    latitude.toCharArray(msg_2,latitude.length()+1);  // Convert string to a Character Array
366    longitude.toCharArray(msg_3,longitude.length()+1);  // Convert string to a Character
       Array
367    batt.toCharArray(msg_4,batt.length()+1);  // Convert string to a Character Array
368    unsigned long currentMillis = millis(); // Every X number of seconds (interval = 5
       seconds)
369      if (currentMillis - previousMillis >= interval) { // Save the last time a new
         reading was published
370         //client.subscribe("ESP32/Output"); //topic=Demo
371         // Send values to MQTT Broker Server
372         client.publish("esp32/reading", msg_1);
373         client.publish("esp32/latitude", msg_2);
374         client.publish("esp32/longitude", msg_3);
375         client.publish("esp32/batteryvolts", msg_4);
376         previousMillis = currentMillis;
377      }
378   }
```