Table of Contents

1. Introduction

A weather station can be described as an instrument or device, which provides us with the information of the weather in our neighboring environment. For example, it can provide us with details about the surrounding temperature, barometric pressure, humidity, etc. Hence, this device basically senses the temperature, pressure, humidity, light intensity, rain value. There are various types of sensors present in the prototype, using which all the parameters can be measured. It can be used to monitor the temperature or humidity of a particular room/place. With the help of temperature and humidity we can calculate other data parameters, such as the dew point. In addition to the above-mentioned functionalities, we can monitor the light intensity of the place as well. We have also enabled to monitor the atmospheric pressure of the room.

This project can be applied and developed in the future: usage in the greenhouses for indoor farming, measure the climate information to control the other components in the greenhouses, for instance, with low moisture, the systems will water the plants, or the light automatically turns on when the light intensity is low.

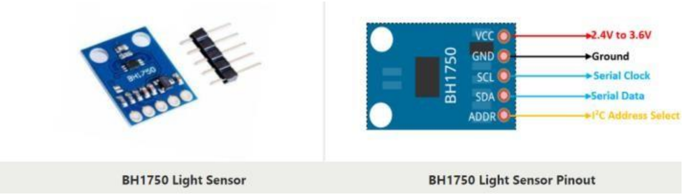
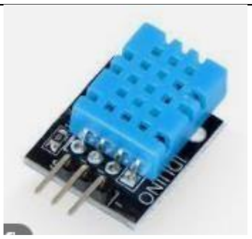
2. Theory description

The plan is to deploy the weather station remotely and read the data from **Blynk app, Thingspeak.com**. In the earlier days, weather parameters like ambient temperature, humidity, and barometric pressure were measured with separate analog instruments: thermometer, hygrometer, and barometer. But today the market is flooded with cheap and efficient **digital sensors** that can be used to measure a variety of environmental parameters. The best examples are sensors like DHT11, DHT 22, BMP180, BMP280, etc.

But in this project, we will use an **ESP8266-12E** Wi-Fi MCU and measuring the weather information from these digital sensors: **BH1750, BM180, DHT11**, we display on the **OLED** screen, read the data from **Blynk IOT** app and display data on **Thingspeak.com** by graphs.

- **BH1750** Is a Digital Light sensor, which is a digital Ambient Light Sensor IC for I2C bus interface. This IC is the most suitable to obtain the ambient light data for adjusting LCD and Keypad backlight power of Mobile phone. It is possible to detect wide range at High resolution. **Connection:** VCC > 3.3 V, GND to GND, SCL to pin D1, SDA to pin D2.
- **BM180** is precision sensor from Bosch is the best low-cost sensing solution for measuring barometric pressure and temperature. Because pressure changes with altitude you can also use it as an altimeter! The sensor is soldered onto a PCB with a 3.3V regulator, I2C level shifter and pull-up resistors on the I2C pins. **Connection:** VCC > 3.3 V, GND to GND, SCL to pin D1, SDA to pin D2.
- **DHT11** Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. The DHT11 spec is only +/- 5% humidity and 2 degrees Celsius so it is not a super accurate sensor. It's also only good for temperatures above freezing. **Connection:** VCC > 3.3 V, GND to GND, DATA to pin D5.
- **Blynk app** is an application for IOT with free Cloud so that we can manage data from our electronics devices from far away.
- **Thingspeak.com** is a website for IOT analytics where you can manage your data by graphs and charts. From this website you can also push your data to **Matlab analysis** or **Matlab Vizualization**.

List of components

Components	Description or Image
ESP8266-12E (MCU)	https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet
BH1750 (Light intensity sensor)	https://components101.com/sensors/bh1750-ambient-light-sensor  BH1750 VCC GND SCL SDA ADDR 2.4V to 3.6V Ground Serial Clock Serial Data I²C Address Select BH1750 Light Sensor BH1750 Light Sensor Pinout
BM180 (Air pressure and)	 Link: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf
OLED SSD1306(0.96 inch)	 Link: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
DHT11 (Humidity temperature sensor)	 Link:https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

3. Measurement and installation

Measurement:

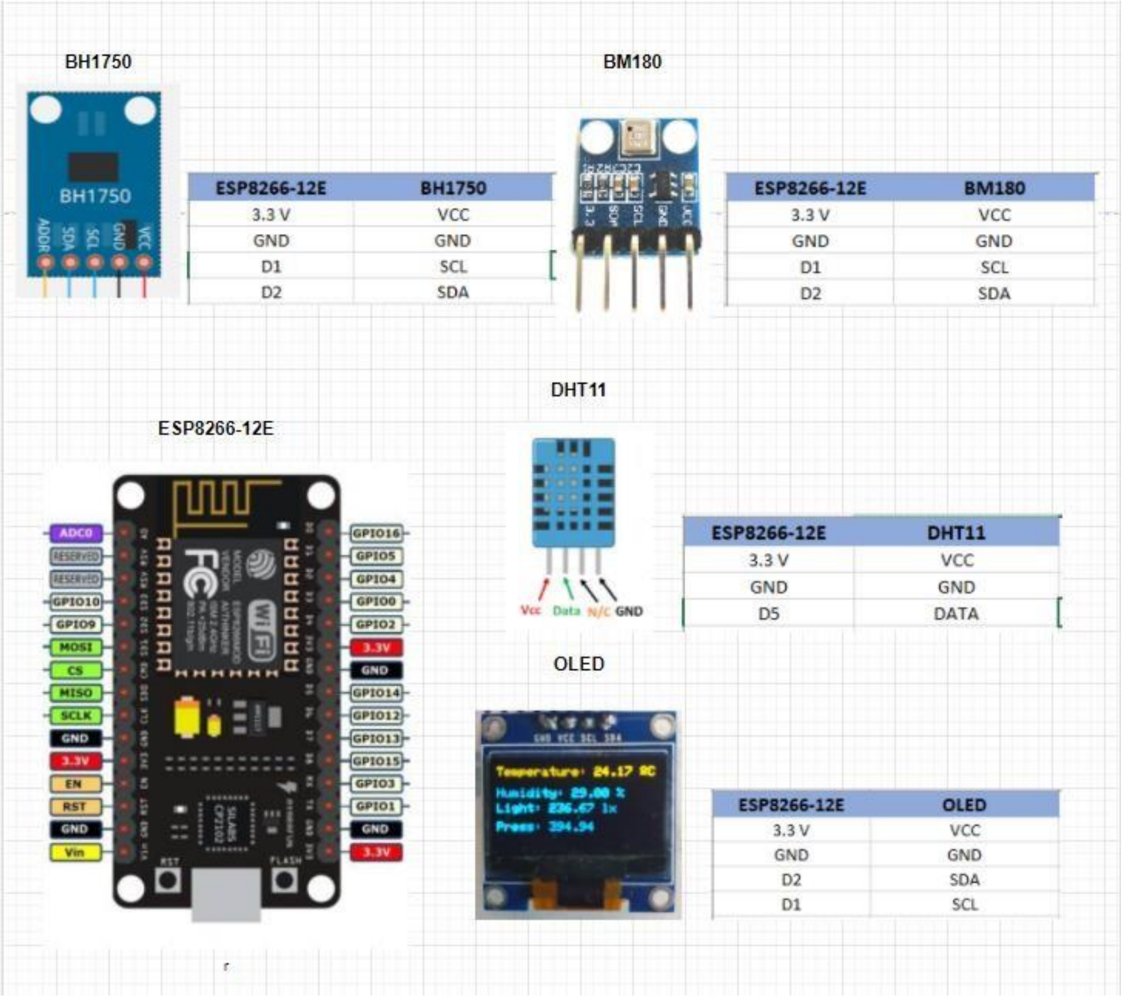
When making **algorithm** for a sensor, usually we need to do variety of testing but all these sensors we used in this project are **digital** sensors with **pre-built** libraries.

The libraries we used:

https://drive.google.com/drive/folders/1q6QS1V6Nz_7t4-dbhkIYitpK8rgA-tKI?usp=sharing

Installation:

Diagram and connections:



ESP8266-12E	BH1750
3.3 V	VCC
GND	GND
D1	SCL
D2	SDA

ESP8266-12E	BM180
3.3 V	VCC
GND	GND
D1	SCL
D2	SDA

ESP8266-12E	DHT11
3.3 V	VCC
GND	GND
D5	DATA

ESP8266-12E	OLED
3.3 V	VCC
GND	GND
D2	SDA
D1	SCL

Figure 1: Circuit

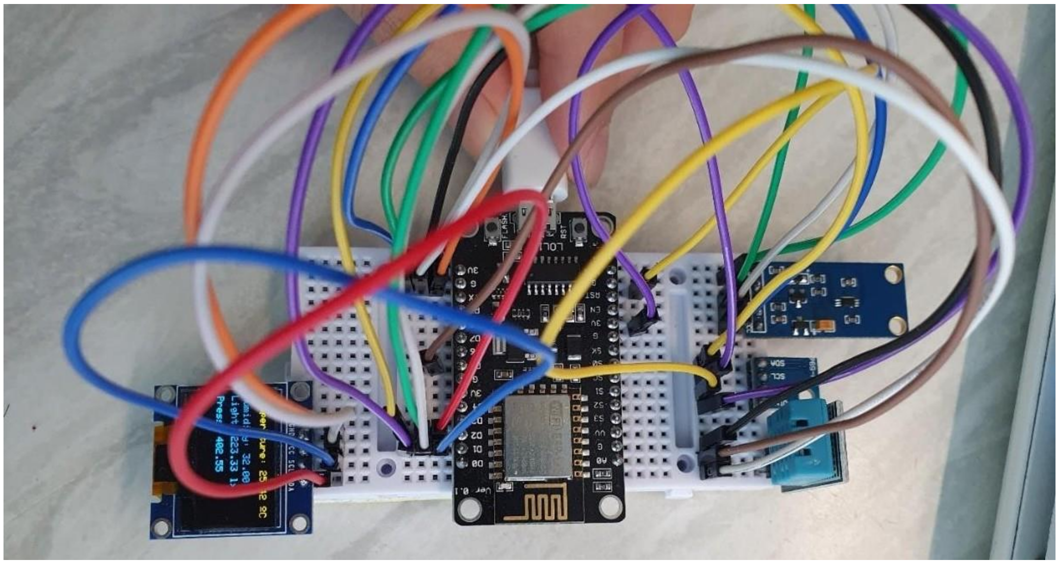


Figure 2: Real-life setup

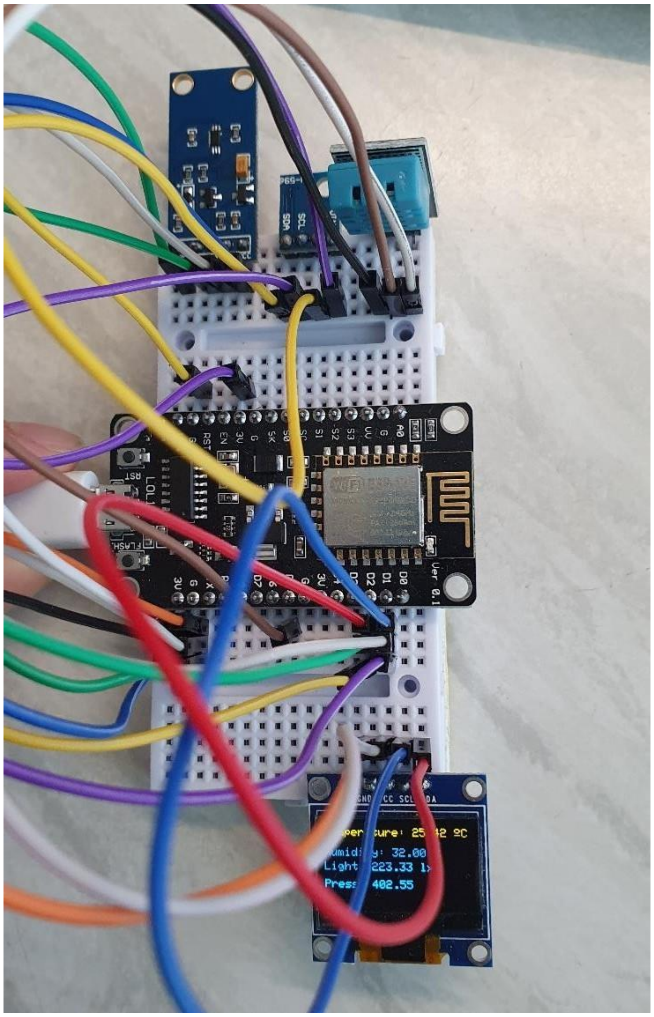
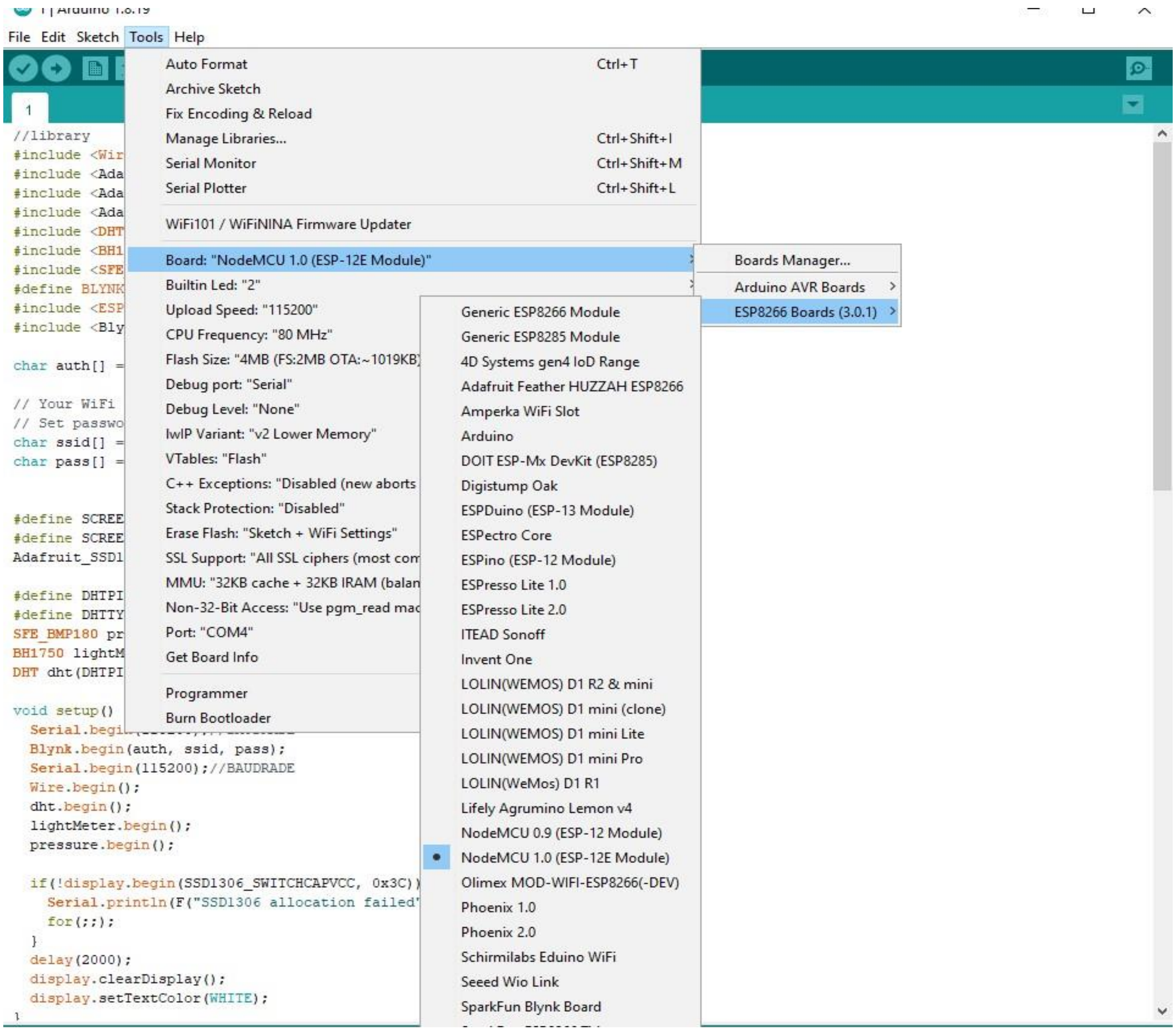


Figure 3: Setup

4. Programming

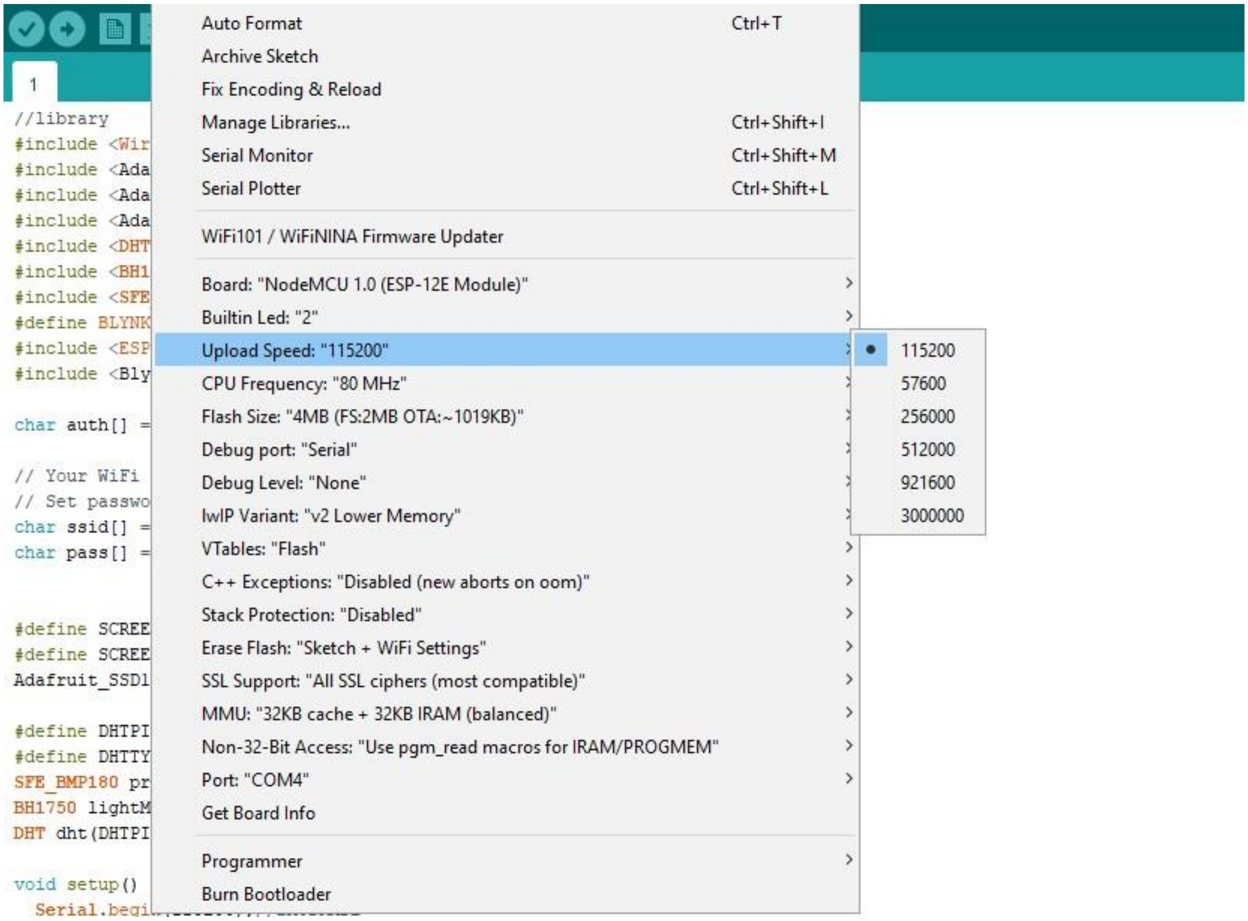
Setup environment Arduino IDE:



Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)
Debug port: "Serial"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts
Stack Protection: "Disabled"
Erase Flash: "Sketch + WiFi Settings"
SSL Support: "All SSL ciphers (most com
MMU: "32KB cache + 32KB IRAM (balan
Non-32-Bit Access: "Use pgm_read mac
Port: "COM4"

```
//library
#include <Wir
#include <Ada
#include <Ada
#include <Ada
#include <DHT
#include <BH1
#include <SFE
#define BLYNK
#include <ESP
#include <Bly

char auth[] =

// Your WiFi
// Set passwo
char ssid[] =
char pass[] =


#define SCREE
#define SCREE
Adafruit_SSD1

#define DHTPI
#define DHTTY
SFE_BMP180 pr
BH1750 lightM
DHT dht(DHTPI

void setup()
  Serial.begi
  Blynk.begin(auth, ssid, pass);
  Serial.begin(115200);//BAUDRADE
  Wire.begin();
  dht.begin();
  lightMeter.begin();
  pressure.begin();

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)
    Serial.println(F("SSD1306 allocation failed"
    for(;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
}
```

Figure 4: Choose the right Board



Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Serial"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Sketch + WiFi Settings"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Port: "COM4"
Get Board Info
Programmer
Burn Bootloader

● 115200
57600
256000
512000
921600
3000000

Figure 5: Upload speed

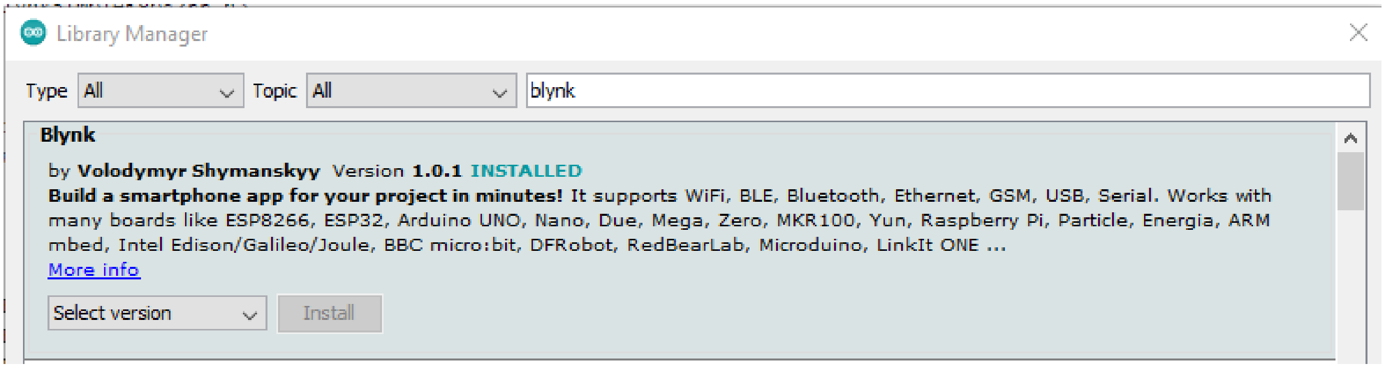
Coding:

```
//library
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <BH1750.h>
#include <SFE_BMP180.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char auth[] = "zPhTRw1mgdVNjbg4i6gi1OWqa2XS_dIq";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "B62";
char pass[] = "0869023151";

//Thingspeak
#include "ThingSpeak.h"
```

```
WiFiClient  client;
//unsigned long myChannelNumber = "1697736";
const char * myWriteAPIKey = "G6D1RICPDASOXPVU";

#define SCREEN_WIDTH 128 // OLED display width
#define SCREEN_HEIGHT 64 // OLED display height
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
#define DHTPIN 14
#define DHTTYPE    DHT11
SFE_BMP180 pressure;
BH1750 lightMeter;
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);//BAUDRADE
  Blynk.begin(auth, ssid, pass);
  ThingSpeak.begin(client);
  Serial.begin(115200);//BAUDRADE
  Wire.begin();
  dht.begin();
  lightMeter.begin();
  pressure.begin();

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
}

void loop() {
  Blynk.run();
  //read LIGHT INTENSITY lux
  float lux = lightMeter.readLightLevel();
  //read Temperature and Humidity
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
  }
  float pres = 0;
  float temp = 0;
  double T, P, p0, a;
  pressure.startTemperature();
  pressure.getTemperature(T);
  pressure.startPressure(3);
  pressure.getPressure(P, T);
  pres = P;
  temp = T;
  // clear display
  display.clearDisplay();

  // display temperature
  display.setTextSize(1);
  display.setCursor(0,3);
  display.print("Temperature: ");
  display.setTextSize(1);
  display.print(temp);
  display.print(" ");
  display.setTextSize(1);
  display.cp437(true);
  display.write(167);// Celsius
  display.setTextSize(1);
  display.print("C");
  Blynk.virtualWrite(V0, temp);
ThingSpeak.writeField(1697736, 1, temp, myWriteAPIKey);

  // display humidity
  display.setTextSize(1);
  display.setCursor(0,18);
  display.print("Humidity: ");
  display.setTextSize(1);
  display.print(h);
  display.print(" %");
  Blynk.virtualWrite(V1, h);
  ThingSpeak.writeField(1697736, 2, h, myWriteAPIKey);

  // display light intensity
  display.setTextSize(1);
  display.setCursor(0,30);
  display.print("Light: ");
  display.print(lux);
  display.print(" lux");
  Blynk.virtualWrite(V3, lux);
  ThingSpeak.writeField(1697736, 3, lux, myWriteAPIKey);

   // display pressure
  display.setTextSize(1);
  display.setCursor(0,45);
  display.print("Air Pressure: ");
  display.print(pres);
  Blynk.virtualWrite(V2, pres);
ThingSpeak.writeField(1697736, 4, pres, myWriteAPIKey);

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

5. Blynk app

- Download Blynk app to your phone.
- Install Blynk library in Arduino IDE.



Library Manager

Type All Topic All blynk

Blynk

by **Volodymyr Shymanskyy** Version **1.0.1** **INSTALLED**
Build a smartphone app for your project in minutes! It supports WiFi, BLE, Bluetooth, Ethernet, GSM, USB, Serial. Works with many boards like ESP8266, ESP32, Arduino UNO, Nano, Due, Mega, Zero, MKR100, Yun, Raspberry Pi, Particle, Energia, ARM mbed, Intel Edison/Galileo/Joule, BBC micro:bit, DFRobot, RedBearLab, Microduino, LinkIt ONE ...
More info

Select version Install

- Settings and create Dashboard for your Weather station.

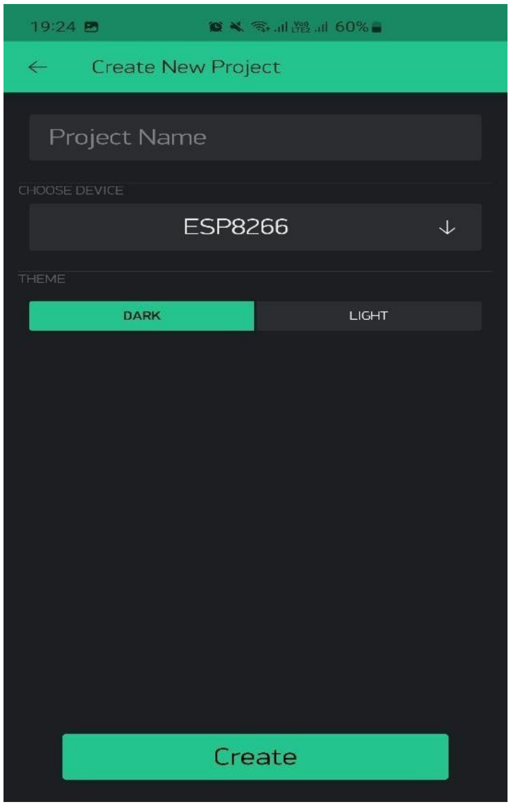


Figure 6: Create project

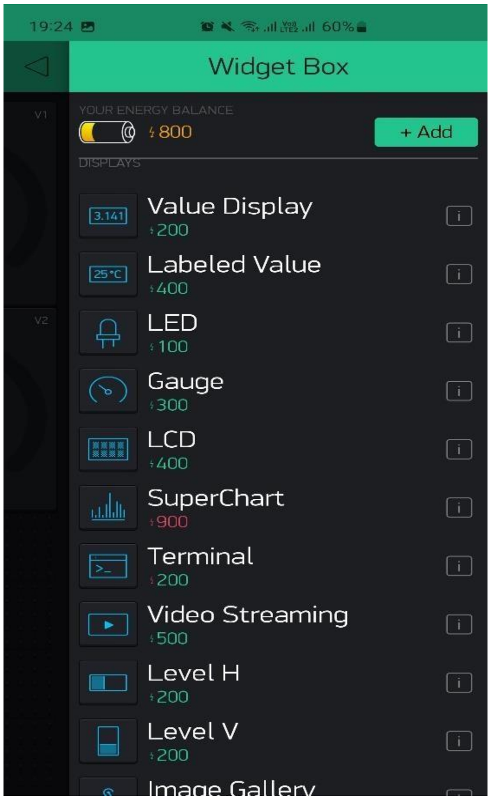


Figure 7: Select 4 Gauge for 4 data sensors

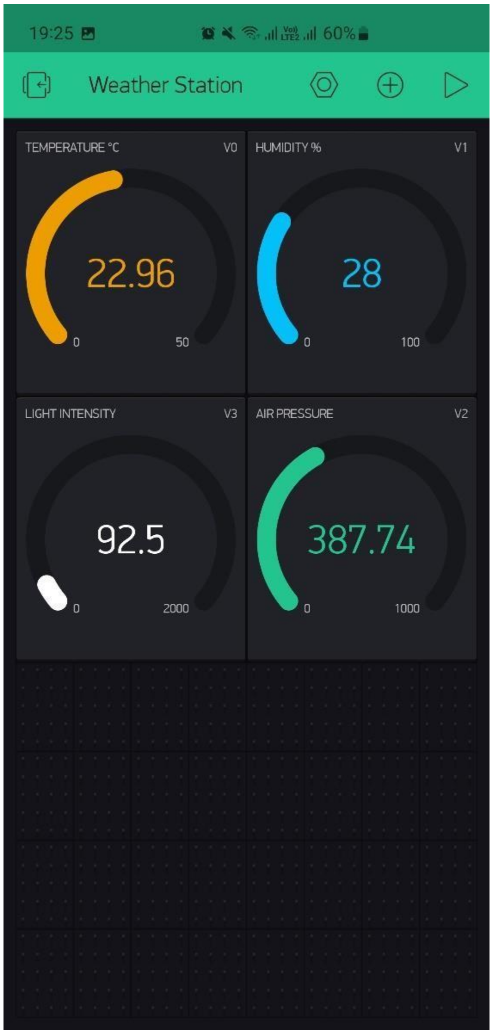
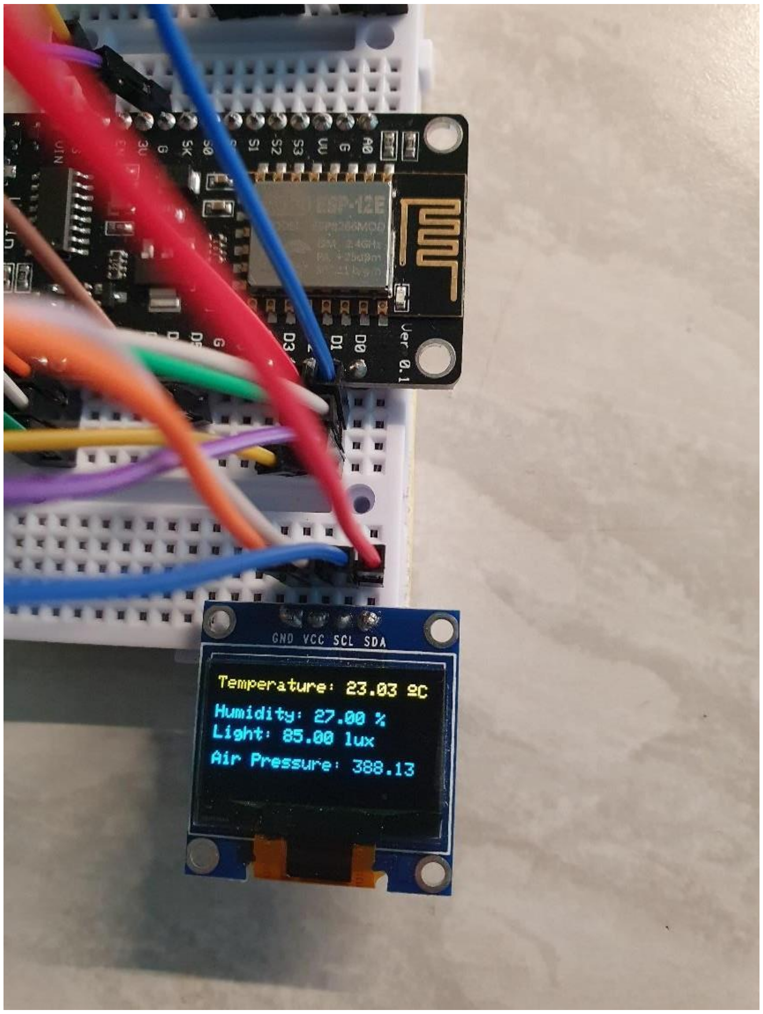


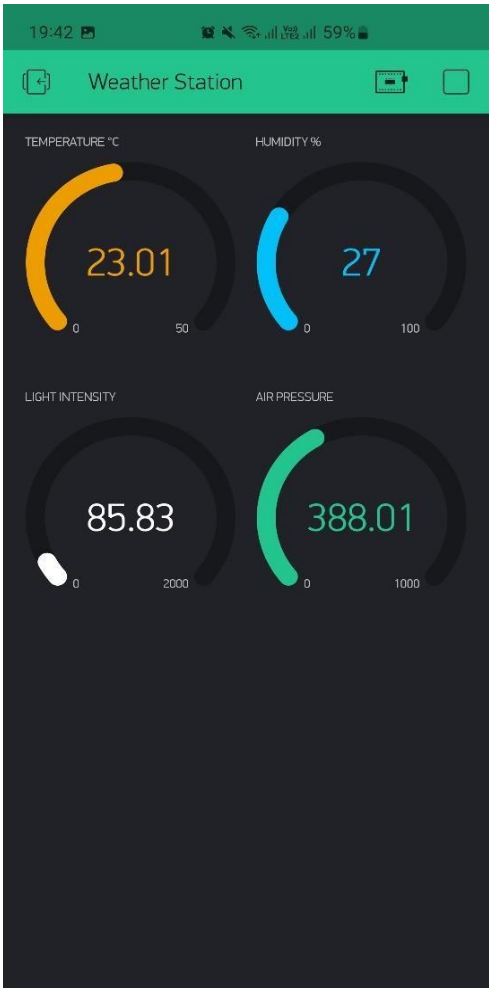
Figure 8: The dashboard

- The Dashboard is consisting of different widgets. To add widgets, follow the steps below:
- Click “Create” to enter the main Dashboard screen.
- Next, press “+” again to get the “Widget Box”
- Then drag 4 Gauges.
- Click on the graphs, it will pop up a settings menu as shown above.
- You have to change the name "Temperature", Select the Virtual Pin V1, then change the range from 0 -50. Similarly, do for other parameters.
- Finally, drag a graph and repeat the same procedure as in gauge settings. The final dashboard picture is shown in the above picture.
- You can change the color also by clicking the circle icon on the right side of the Name.

So now we have our data display from the OLED and from Blynk app:



```
Temperature: 23.03 ºC
Humidity: 27.00 %
Light: 85.00 lux
Air Pressure: 388.13
```



As we can see here, the room **temperature** of a member is **23.01** Celsius Degree. The **humidity** accounts for **27%** while **the air pressure** is **388.01** Pa. Additionally, the **light intensity** in the room is **85.83** lux.

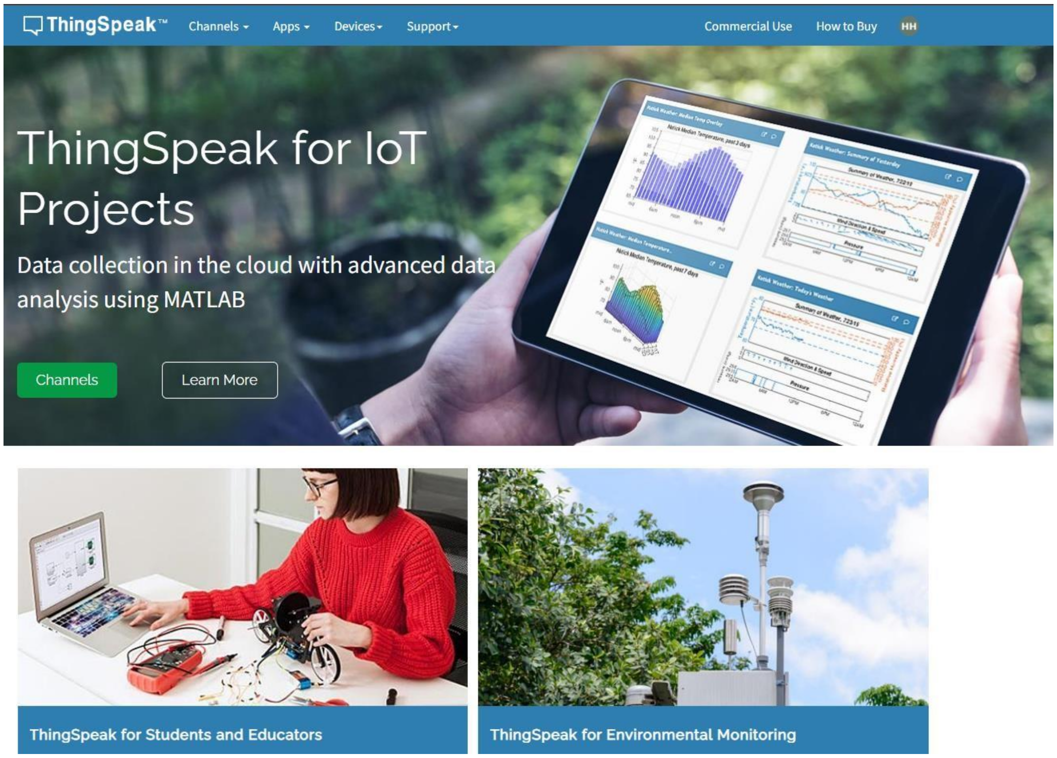
<u>Code part for upload data to Blynk:</u>

```
//Blynk
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "zPhTRw1mgdVNjbg4i6gi1OWqa2XS_dIq";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Bin";
char pass[] = "888888888";
```

```
Blynk.virtualWrite(V0, temp);
Blynk.virtualWrite(V1, h);
Blynk.virtualWrite(V3, lux);
Blynk.virtualWrite(V2, pres);
```

6. Thingspeak.com

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

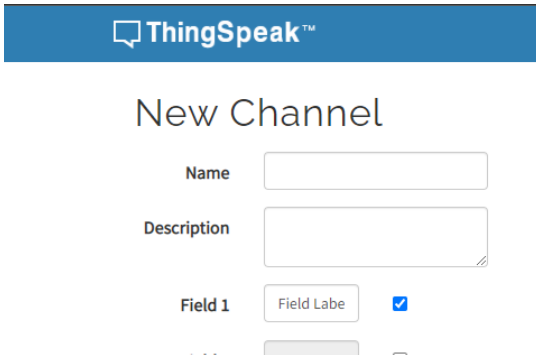


ThingSpeak for IoT Projects

Data collection in the cloud with advanced data analysis using MATLAB

Figure 9: Sign up

Visit ThingSpeak.com and Sign Up for an account. This will just take a minute and user accounts are free. Once you have a user account, you need to create a channel. ThingSpeak channels are where data gets stored. Create a new channel by selecting Channels, My Channels, and then New Channel. Name the channel, “Weather Station” and name Field 1, “Temperature”, Field 2 “Humidity” … Click “Save Channel” at the bottom to finish the process.



ThingSpeak™

New Channel

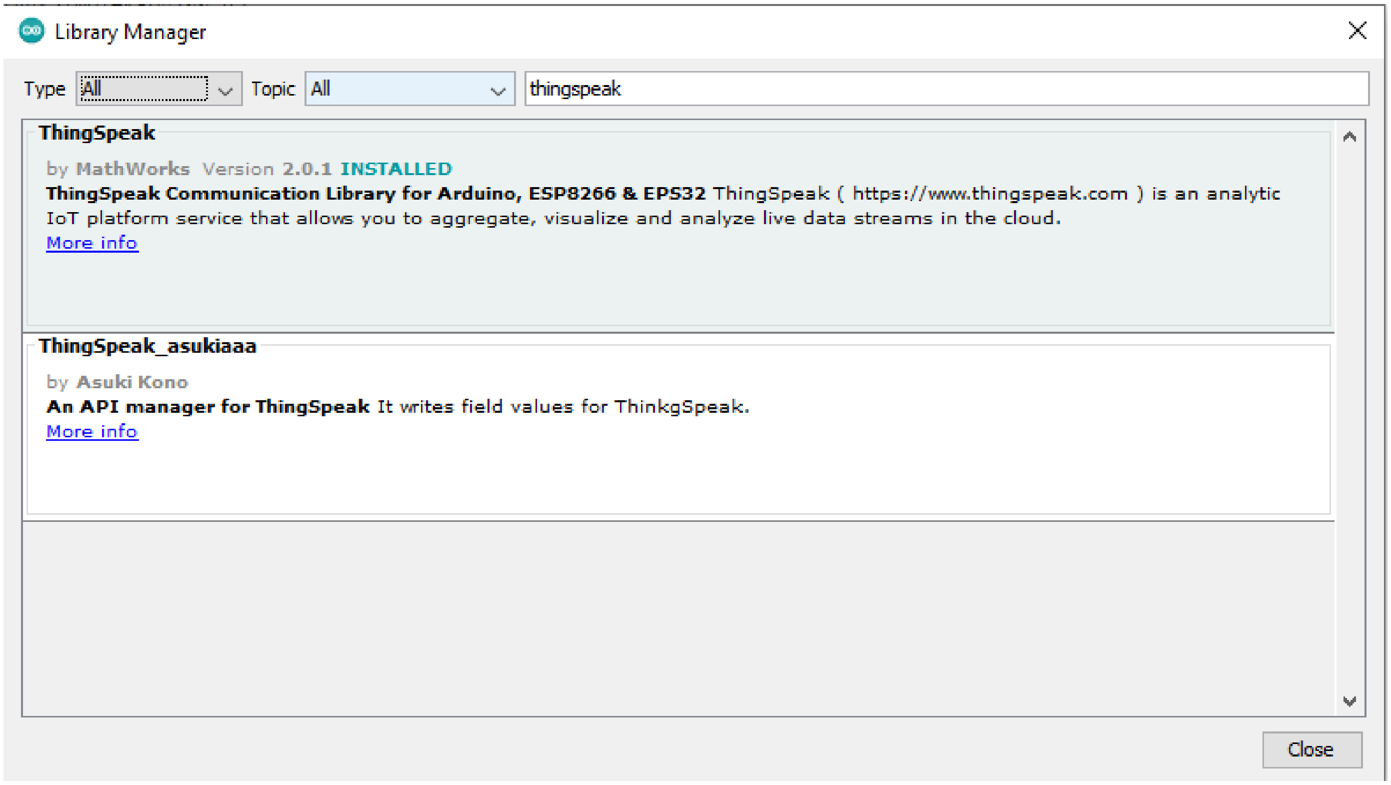
Name

Description

Field 1 Field Labe ☑

Figure 10: Create channel

Install Thingspeak library on Arduino IDE:



Type All Topic All thingspeak

ThingSpeak

by **MathWorks** Version **2.0.1** **INSTALLED**

ThingSpeak Communication Library for Arduino, ESP8266 & EPS32 ThingSpeak (https://www.thingspeak.com) is an analytic IoT platform service that allows you to aggregate, visualize and analyze live data streams in the cloud.

More info

ThingSpeak_asukiaaa

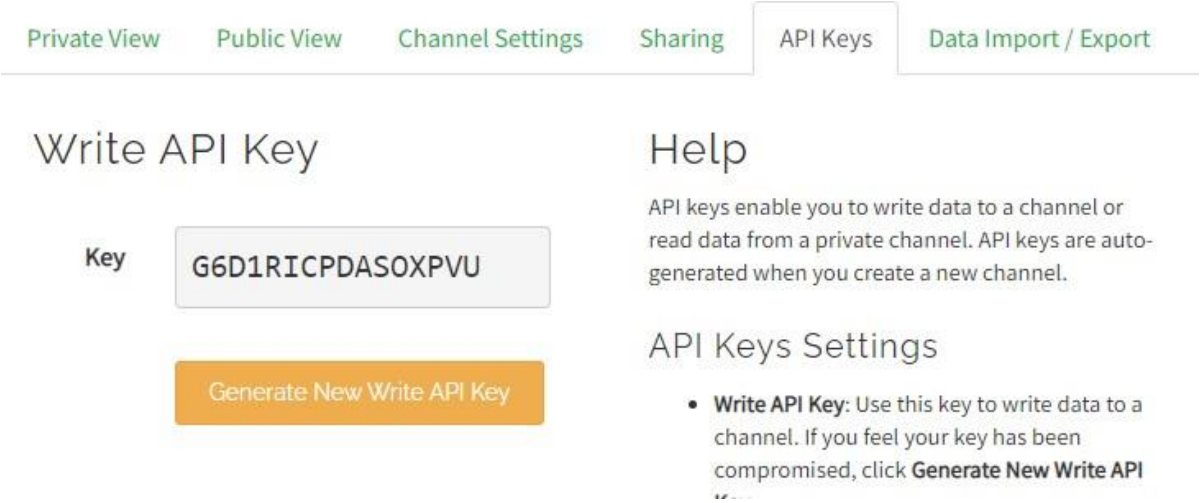
by **Asuki Kono**

An API manager for ThingSpeak It writes field values for ThinkgSpeak.

More info

Figure 11: Thingspeak library

Copy the API and start coding:



Write API Key

Key G6D1RICPDASOXPVU

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API**

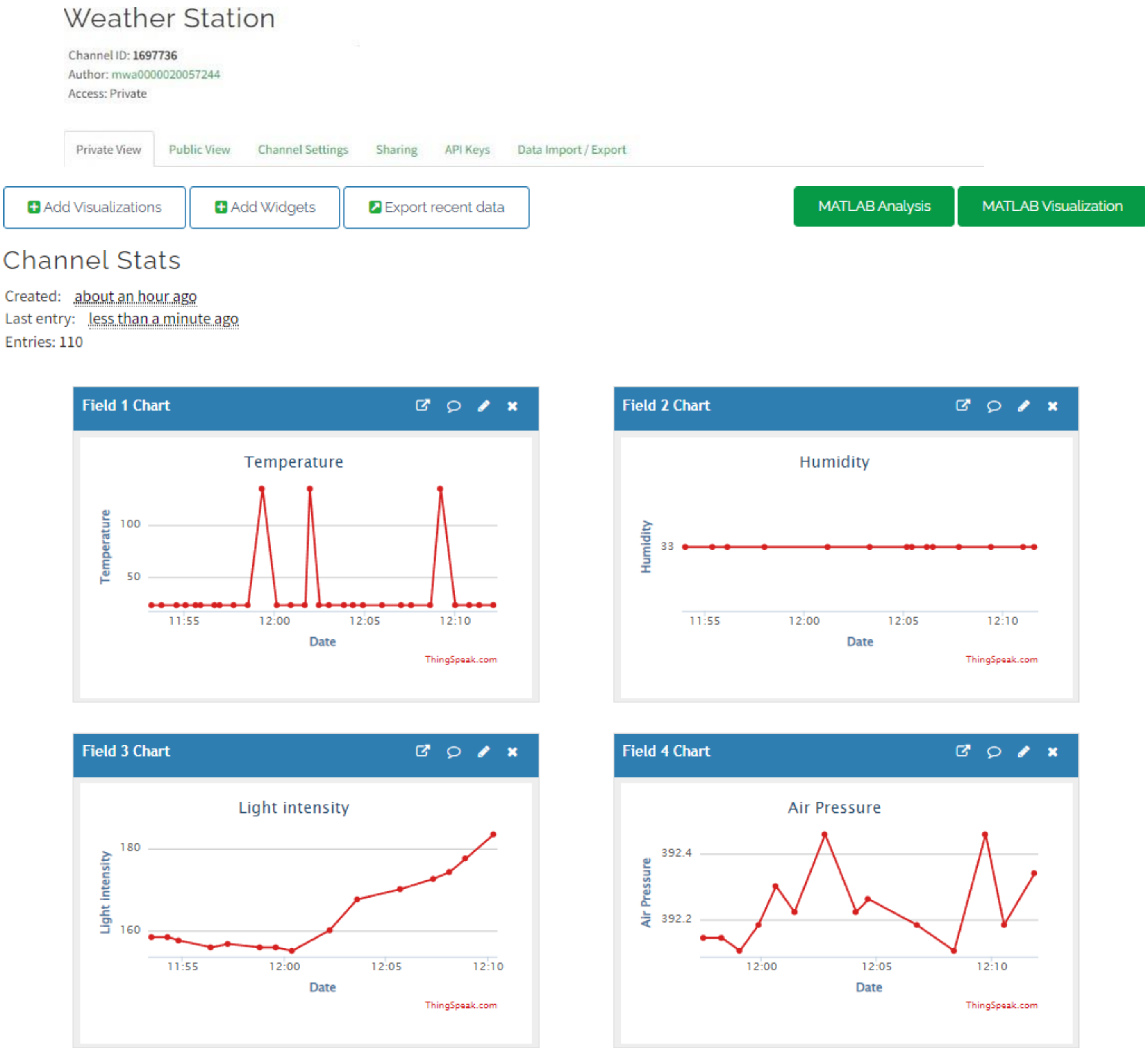
Figure 12: API key

Coding part:

```
#include "ThingSpeak.h"
WiFiClient  client;
//unsigned long myChannelNumber = "1697736";
const char * myWriteAPIKey = "G6D1RICPDASOXPVU";

ThingSpeak.begin(client);
ThingSpeak.writeField(1697736, 1, temp, myWriteAPIKey);
ThingSpeak.writeField(1697736, 2, h, myWriteAPIKey);
ThingSpeak.writeField(1697736, 3, lux, myWriteAPIKey);
ThingSpeak.writeField(1697736, 4, pres, myWriteAPIKey);
```

Result on Thingspeak:

We have all the data information and display in graphs in real-time, this way helps you to manage and analys your data effectively:



Weather Station

Channel ID: 1697736
Author: mwa0000020057244
Access: Private

Channel Stats

Created: about an hour ago
Last entry: less than a minute ago
Entries: 110

Figure 13: Thingspeak DashBoard

7. Conclusion

In conclusion, this small system can be applied in any location which is useful for measuring the temperature, light intensity, air humidity and pressure at that location. This project can be developed further in the future.