KOLEJ PROFESIONAL MARA BERANANG

FINALE PROJECT SESSION 1 2022/2023

PREPARED FOR:

TUAN ZAKARIA BIN ROSMAN

PREPARED BY:

AISYAH AINA SUFIA BINTI HILMAN

MENTOR'S NAME:

PUAN NOOR FAZLIANI BINTI SHAMSHUDIN

(**Youtube link's video of Finale Project IoT : https://youtu.be/2VsXdPczwvM**)

Scenario

Project Proposal, project costing and IoT System Design Methodology documents have been completed and approved by the respective lecturer from Quantitative Science department (From Phase 1) based on the selected domain developed by the student. The next step is to develop the prototype of an Internet of Things (IoT) application in the selected domain.

Below are the tasks that you need to accomplished in order to develop the new IoT project for MARA Professional College, Beranang (MPCB).

All the tasks details are mention below:

Tasks:

1. Prepare a **Final Prototype** for the developed IoT project.
    - IoT Devices
    - IoT Network Devices
    - IoT Cloud
    - IoT Application
    - IoT Model

2. Prepare a **Final Report in hardcopy** for the developed IoT project.
    - Introduction
    - User Manual
    - Coding
    - Related pictures

3. Present the IoT project by preparing a **demo video** and upload the video in Youtube.
    - Content
    - Quality
    - Creativity
    - Accesibility
    - Voice

TABLE OF CONTENTS PAGES

1.0) ACTUAL & PROPOSE PROJECT JUSTIFICATION

In the first assignment, a proposed project was created with the goal of successfully getting the project to work, organising it, and schematicizing it in accordance with the plans. However, there are some components that are inexplicably not functioning, and in order to keep the project in a suitable state, it is necessary to address these problems.

- Discarded Items :

I. IoT Cloud Bolt

This cloud has an incredible features . Unfortunately, it's only works if only used their IoT product range, which is Bolt Wifi Module. And it's also very expensive item which is the price is around RM187.90 that import from India . Plus, IoT Bolt Cloud is integrated with Twillio and Mailgun. So those both software can't be used.

II. Temperature sensor

Temperature sensor is a good tools to make prediction of flood. But, it didn't worth it when it comes to small project. Because it has low capability to sense flood which in other words inaccurate is high . This will cost a lot of power supply needed and money in order to make it function.

III. LCD Display

LCD is a good screen display data in order for people to get alert with it . Unfortunately, in order to save cost instead using the ThingView App to monitor the flood level compare to LCD. The app required zero cost and save more time and power supply usage.

IV. Phyton IDE

Using Phyton will be an advantage to the project. But I have fewer knowledge about phyton code and could not identify where the error is and also there are not many phyton code suitable with Arduino

V. Arduino Uno R3

Most Arduino boards do not come with Bluetooth or WiFi compared to nodeMCU ESP 8266 . The function of NodeMCU ESP 8266 alone is already functioning that can be replaced by Arduino Uno R3. WiFi and Bluetooth chips typically add $10-$20 USD dollar to the cost of a board.

- Added Items:

VI. NodeMCU ESP 8266

NodeMCU is a microcontroller development board has with wifi capability. It uses an ESP8266 microcontroller chip. Whereas Arduino UNO uses an ATMega328P microcontroller. Besides the chip, it contains other elements such as crystal oscillator, voltage regulator, etc.

2.0) INTRODUCTION

Flood Monitoring System Using ESP8266, Ultrasonic Sensor, and Thingspeak IoT Platform on IoT Platform Based on NodeMCU ESP8266. As we all know, flooding is a huge natural disaster, especially in Malaysia where it is particularly well-known. It has a devastating effect on the environment and the lives of all living things. It is crucial in these situations to have access to real-time water level warnings.

By doing this project in Kolej Profesional MARA Beranang (KPMB), it helps to sense the water level in the guardhouse since it was the lowest point of the college and also the car parking for students that are parked infront of the college by checking if they are in normal condition. If they reach beyond the limit, then it alerts people and students through LED signals with buzzer sound as well as internet applications. So the students able to save their cars and parking inside the college before their cars strucked by flood and muddy mudwet that could damage the car engine. Here, we are using the Ultrasonic HC-SR04 sensor to sense the water level and NodeMCU ESP8266 Microcontroller for data processing. The processed data will be uploaded to the ThingSpeak IoT Cloud Platform. Using which flood levels can be monitored graphically from anywhere in the world.

KPMB's guardhouse and other neighbouring locations, such as student parking in front of the college, are monitored by an Ultrasonic sensor. Raw data from the ultrasonic sensor is transmitted through NodeMCU to ThingSpeack, where graphical monitoring and critical alerts can be generated. To warn of a flooded area, a red LED and buzzer are utilized. Flood warning LED use red LEDs to warn of potential dangers, while green LEDs show that all is normal.

College’s field was strucked by heavy thunderstorm & flood at 17 June 2022. This could lead to any new pontential disaster once the flooded go down to the guardhouse and could harm the other people.

3.0) USER MANUAL

In this project , the User-Manuals are extremely important since they contain instructions on how to utilise the product. This should be sent to end users with the goal of providing adequate usage guidance and reducing the chance of the equipment becoming unusable.

3.1) Hardware Devices

As previously stated in the justification, the project has go through several alterations to the plan were made. Some of the devices are retained, while some of others are replaced or discarded. So, these are the most recent devices that are presently being used for the actual project.:

I. Ultrasonic Sensor HC-SR04 HC SR 04 Ultrasound Range Finder Distance Measure Measurement Module

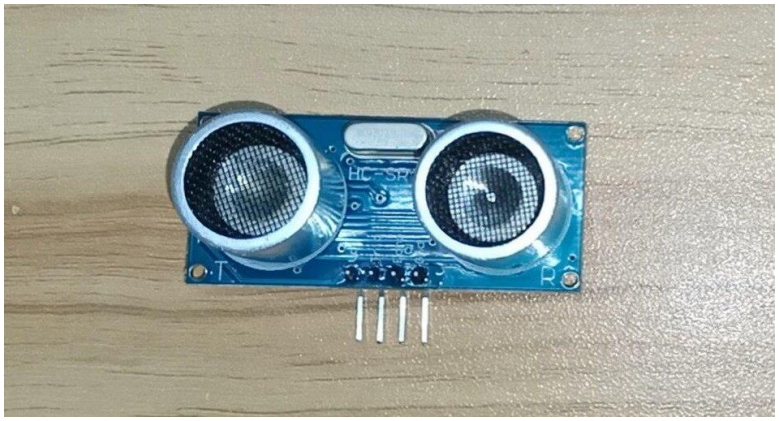


Figure above shows that Ultrasonic Sensor HC-SR04 HC SR 04 in order to measure measurement distance of water level

II. NodeMCU ESP8266 Basic IoT Kit

Figure above shows that NodeMCU ESP 8266 in order to connect objects and let data transfer using the Wi-Fi protocol.

III. Passive Buzzer Module Kit

Figure above shows that Passive Buzzer Module Kit as it's generate sound signals of different frequencies

IV. MB102 Solderless Mini Medium Large Breadboard 170 400 830 Holes Dupont Jumper Wire EE Components Experiment Donut Board

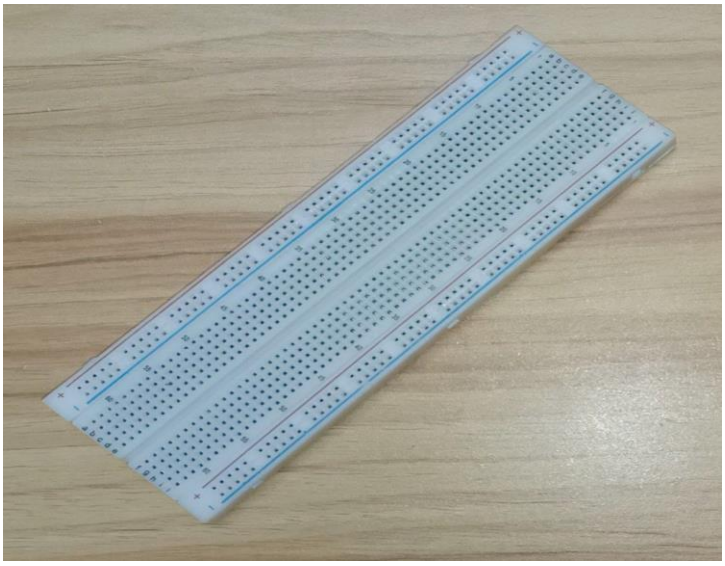


Figure above shows that Breadboard which allows for the rapid and easy development of temporary electrical circuits or circuit design experiments.

V. LED Light 5mm Red & Green

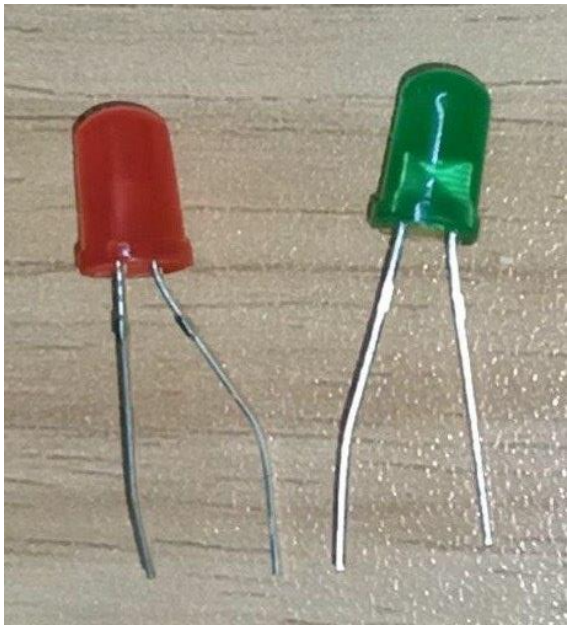


Figure above shows that LED Light as it's transform electrical energy into light instantly, resulting in efficient light generating with little electricity waste

VI. Male to Male Jumper Wire Cable

Figure above shows that Male to Male Jumper Wire Cable as Male-to-male jumper wires used to plug things into . A Male-to-Male wire is required when connecting two ports on a breadboard

VII. Female to Male Jumper Wire Cable

Figure above shows that Female to Male Jumper Wire Cable as it's Male ends have a protruding pin and may be plugged into items, whereas female ends do not have a prominent pin and are used to plug things into.

VIII. USB Cable Micro B

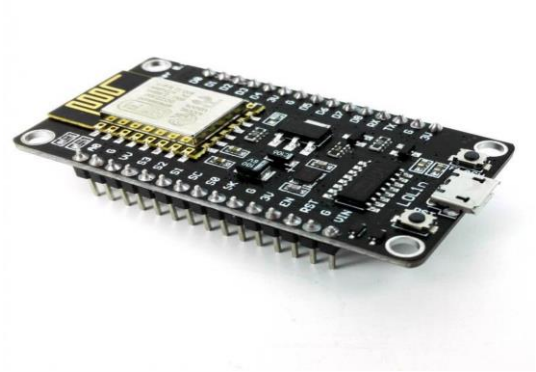
The micro B type connector was used in this project . It has 5 pins for USB OTG, which allows smartphones and other mobile devices to read external discs, digital cameras, and other peripherals in the same way that a PC does.

- **Overall,** this is the latest list of hardware devices for actual finale project:

No.	Hardware Devices
1.	Ultrasonic Sensor HC-SR04 HC SR 04 Ultrasound Range Finder Distance Measure Measurement Module
2.	Passive Buzzer Module Kit
3.	MB102 Solderless Mini Medium Large Breadboard 170 400 830 Holes Dupont Jumper Wire EE Components Experiment Donut Board
4.	LED Light 5mm Red & Green
5.	Male to Male Jumper Wire Cable
6.	Female to Male Jumper Wire Cable
7.	USB cable Micro B

3.2) IoT Network Devices

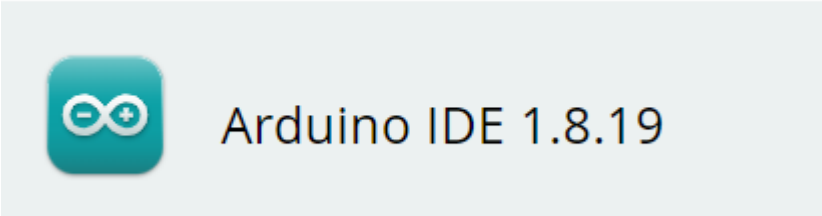
I. NodeMCU ESP8266



The NodeMCU that used for the actual project is a platform that is free and open source IoT platform. It comprises firmware that works on Espressif Systems' ESP8266 Wi-Fi SoC and hardware that is based on the ESP-12 module. By default, the name "NodeMCU" refers to the firmware rather than the development kits.

3.3) Software Devices

I. Arduino IDE



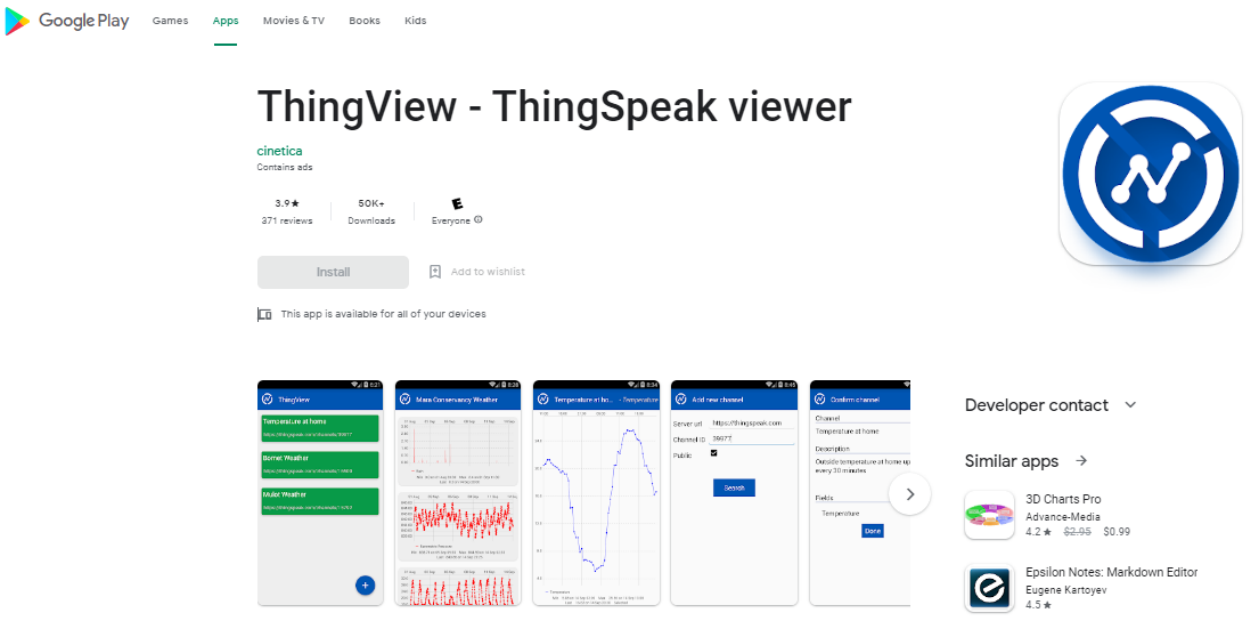
For the actual finale project, writing code and uploading it to the board is simple with the open-source Arduino Software (IDE). Any Arduino board may be used with this software.

3.4) Service & Application Support : CLOUD

I. ThingSpeak IoT Cloud Platform

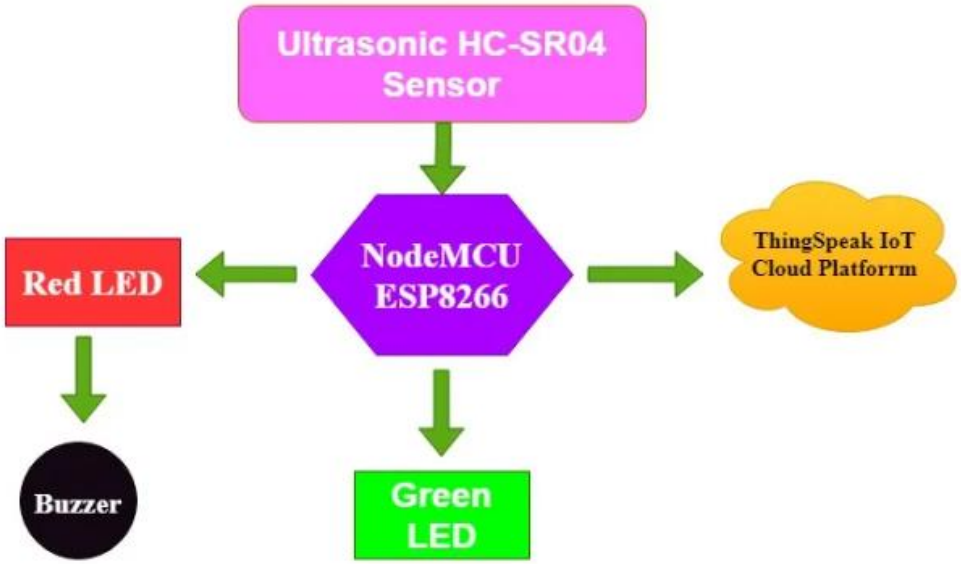
ThingSpeak is a cloud-based IoT analytics tool that lets data gather, visualise, and analyse live data streams. ThingSpeak delivers real-time visualisations of data sent to ThingSpeak by user devices. ThingSpeak is frequently used for IoT system prototype and proof of concept that require analytics

3.5) IoT Application



- **ThingView** makes it simple to see **ThingSpeak** channels by just entering the channel ID, and it's ready to use.
- The programme will preserve the window settings for public channels, including colour, timeframe, chart type, and number of results. Line and column charts are supported in the present version, and spline charts are shown as line charts. The actual finale project will used the public channels.
- But, there is also a private channels, which the data will be displayed using the default settings, as there is no way to read the private windows settings with the api key only.

3.6) Works Flow for IoT Based Flood Monitoring System

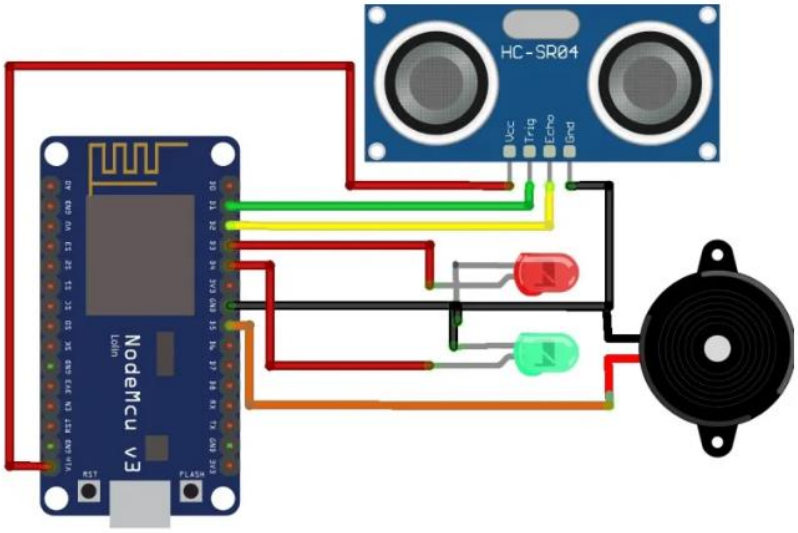


Latest Alteration of Working Flow of Flood Monitoring System For Actual Project

The NodeMCU ESP 8266 and IoT Platform are used in this IoT-based flood monitoring system, as shown in the block diagram above. To detect river water levels, an ultrasonic sensor is used. The ultrasonic sensor's raw data is supplied to the NodeMCU, which processes it before sending it to ThingSpeak for graphical monitoring and critical alerts. In this case, a red LED and a buzzer are utilised to transmit an alert in a flooded situation, and they are used to stay alert in extreme flood situations. Normal is shown by a green LED, but abnormal is indicated by a red LED.

3.7) Schematic of IoT Based Flood Monitoring System

As shown in the circuit design, NodeMCU ESP8266, Ultrasonic HC-SR04 Sensor, Buzzer, and LEDs (Red and Green) are interconnected:

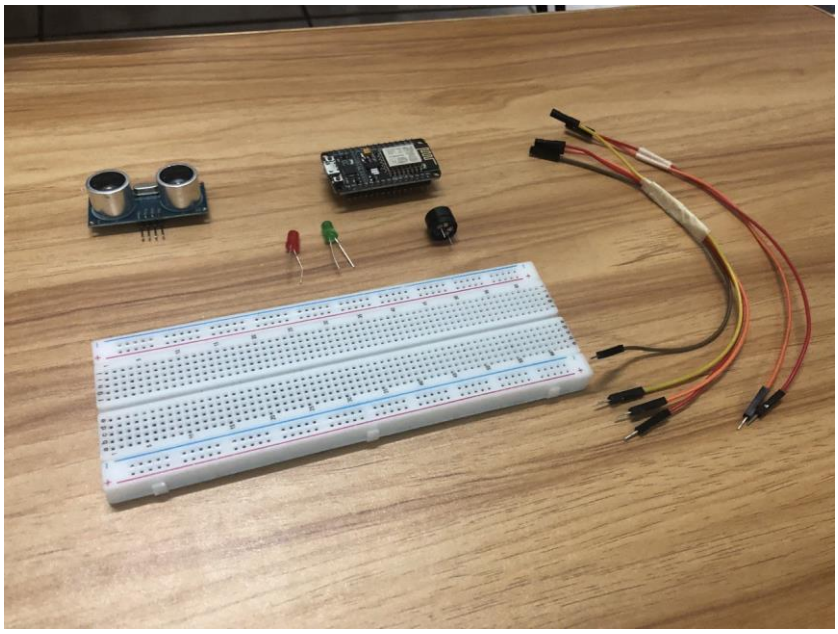


Circuit Diagram of IoT Flood Monitoring System

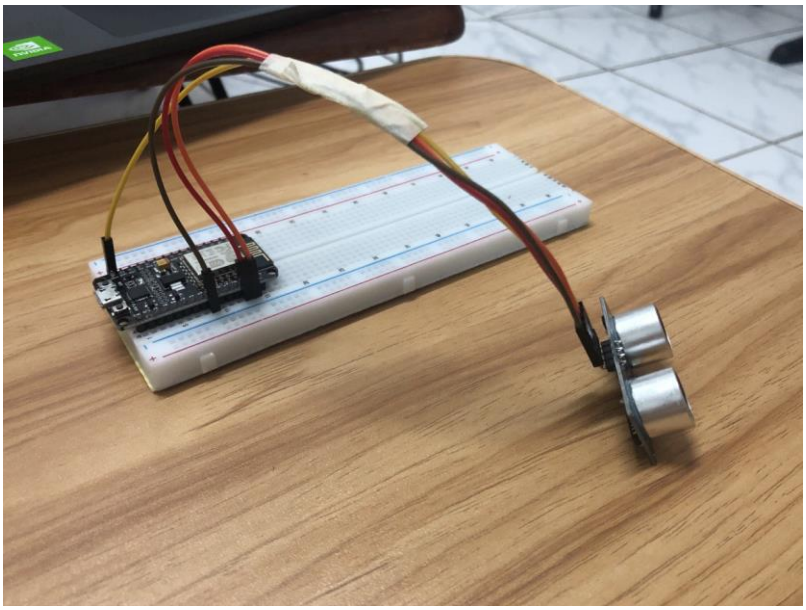
3.8) Hardware Setup

The most important aspect of the process is hardware setup, which focuses on how to manage the hardware components so that everything runs smoothly. Configuration settings on all hardware components can have an influence on overall performance and system functionality. The project will not work properly without the hardware.

This is me configuring the hardware setup



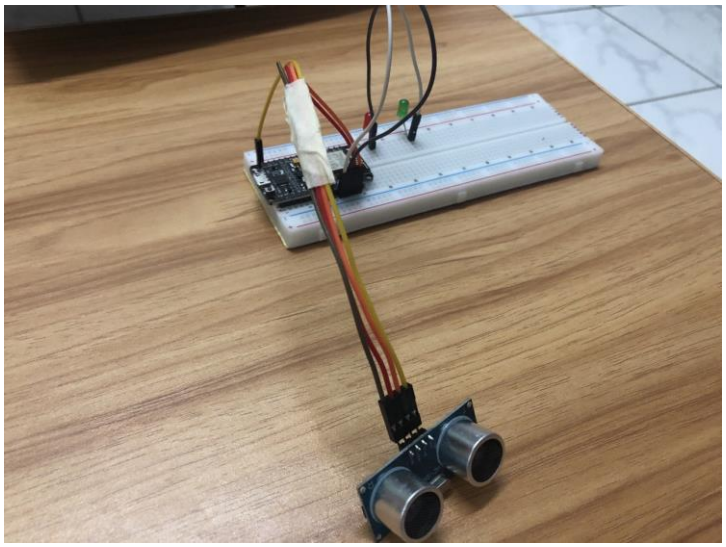
STEP 1: Prepared All the items aboves which is ultrasonic sensor , LED Lights, passive buzzer, breadboard, wire jumper male to male & female to male and NodeMCU ESP 8266



STEP 2: Connect NodeMCU and Ultrasonic Sensor :

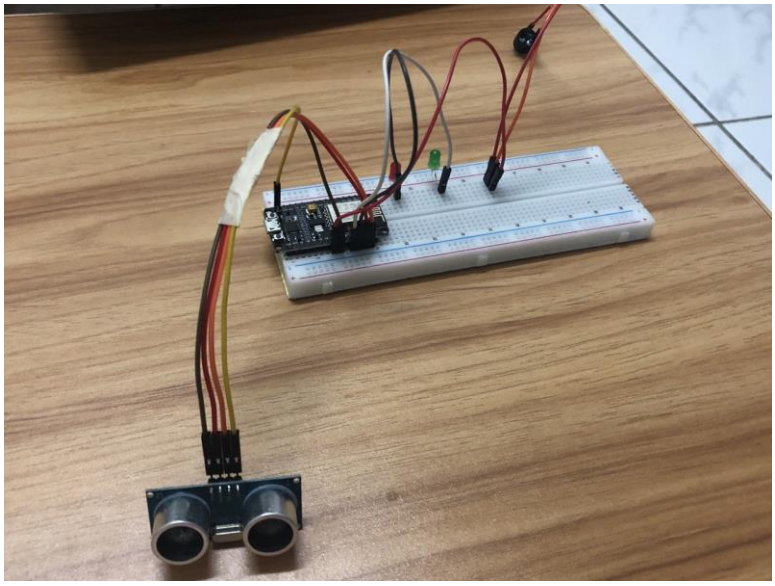
- Place the NodeMCU at the end of the breadboard
- pin the VCC of Ultrasonic to pin VIN NodeMCU
- pin the Trig of Ultrasonic to pin D1 NodeMCU
- pin the Echo of Ultrasonic to pin D2 NodeMCU

- pin the GND of Ultrasonic to pin GND NodeMCU



STEP 3: Connect LEDs

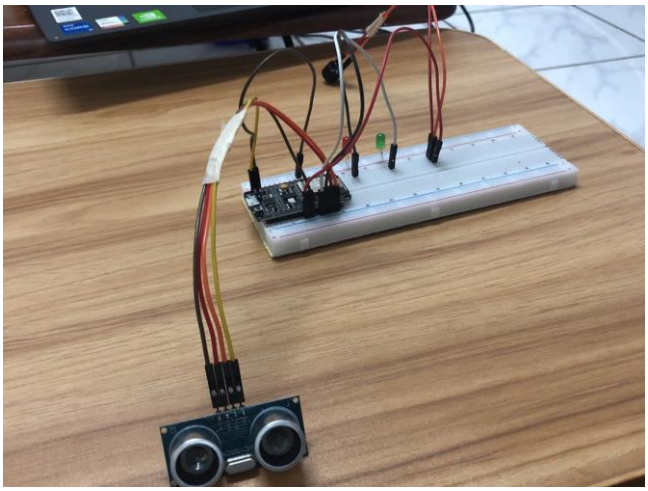
- red LED (cathode) to 22+ & (anode) to 22j
- green LED (cathode) to 30+ & (anode) to 30j
- pin D3 of NodeMCU to pin 22h breadboard
- pin D4 of NodeMCU to pin 30h breadboard



STEP 4: Connecting buzzer

- pin D5 of NodeMCU to pin 41f breadboard
- pin negative of buzzer to pin 41j breadboard

- pin positive of buzzer to pin 40+ breadboard

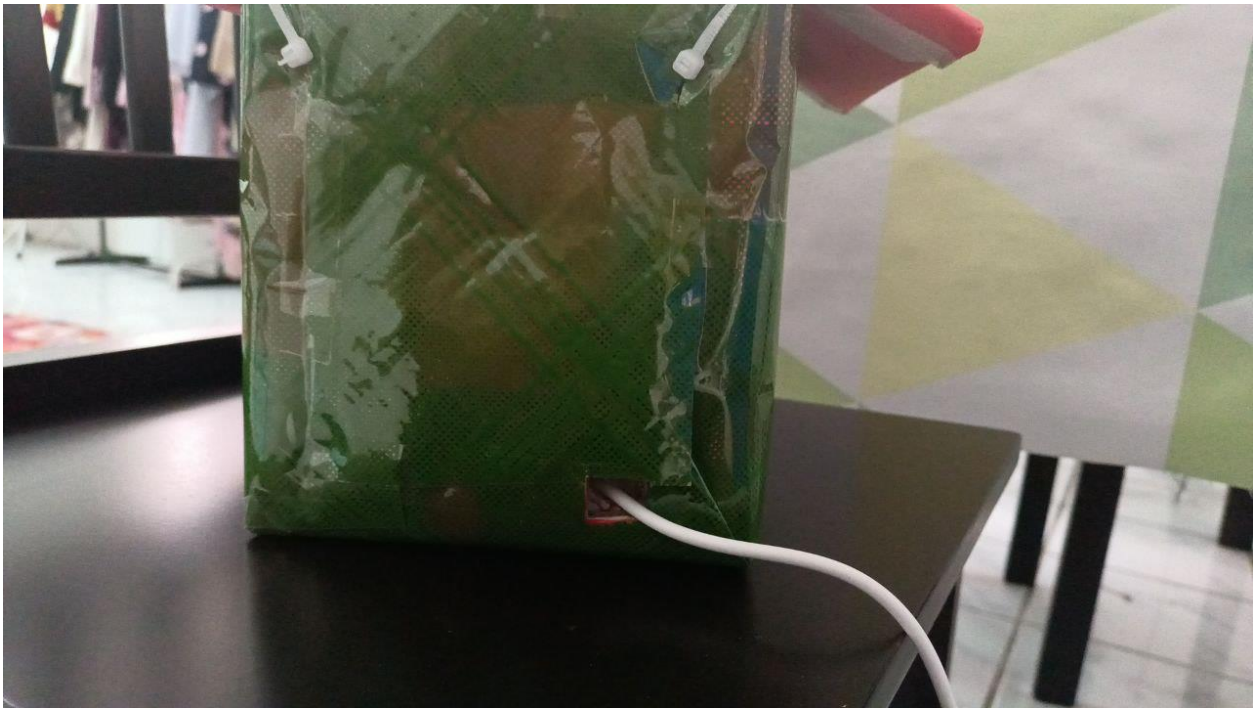


STEP 5: Provide GND

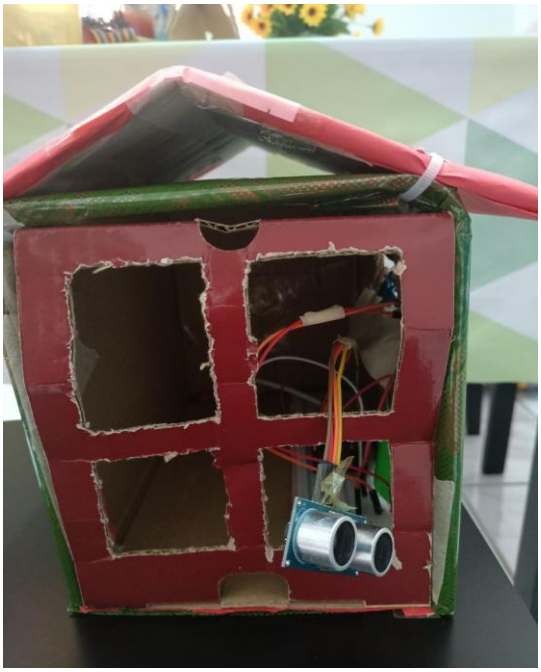
- pin GND of NodeMCU to pin 11+ breadboard

STEP 6: Poke the hole on the prototype and design the prototype before putting the hardware inside the prototype name ‘guardhouse KPMB’

Soldier toys, colour paper, printed pictures and etc to design the prototype



Poke the back hole for the cable micro b to connect to laptop/power bank in order to make hardware function



STEP 7: Put all the hardware into the prototype ‘guardhouse’ through the gate that looks like window

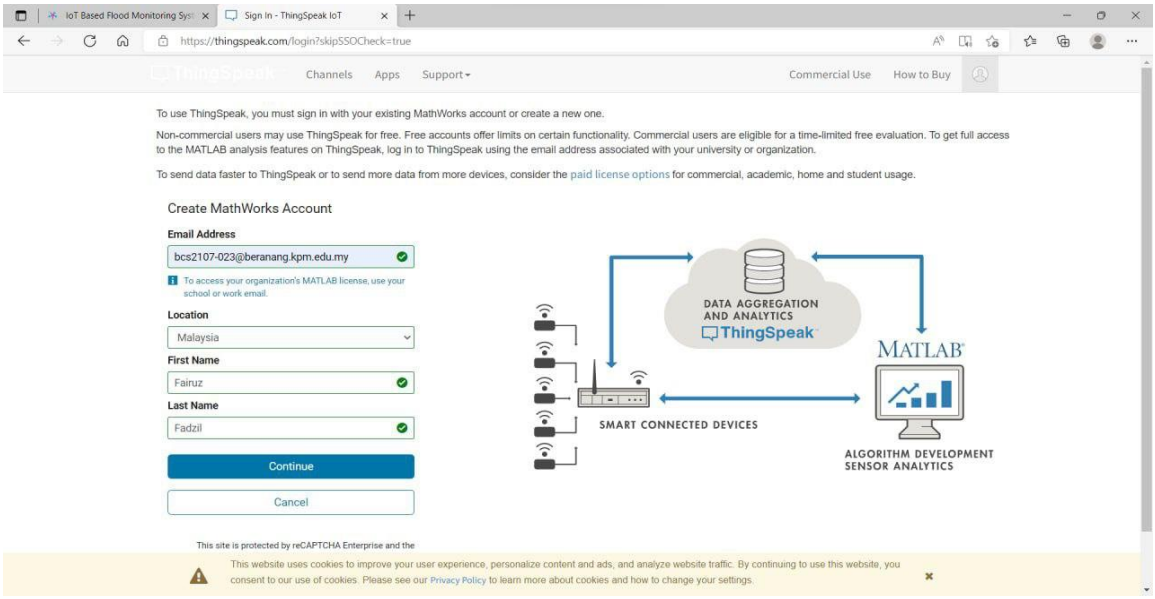
STEP 8 : Final result of the project was successfully made

3.9) Cloud Setup

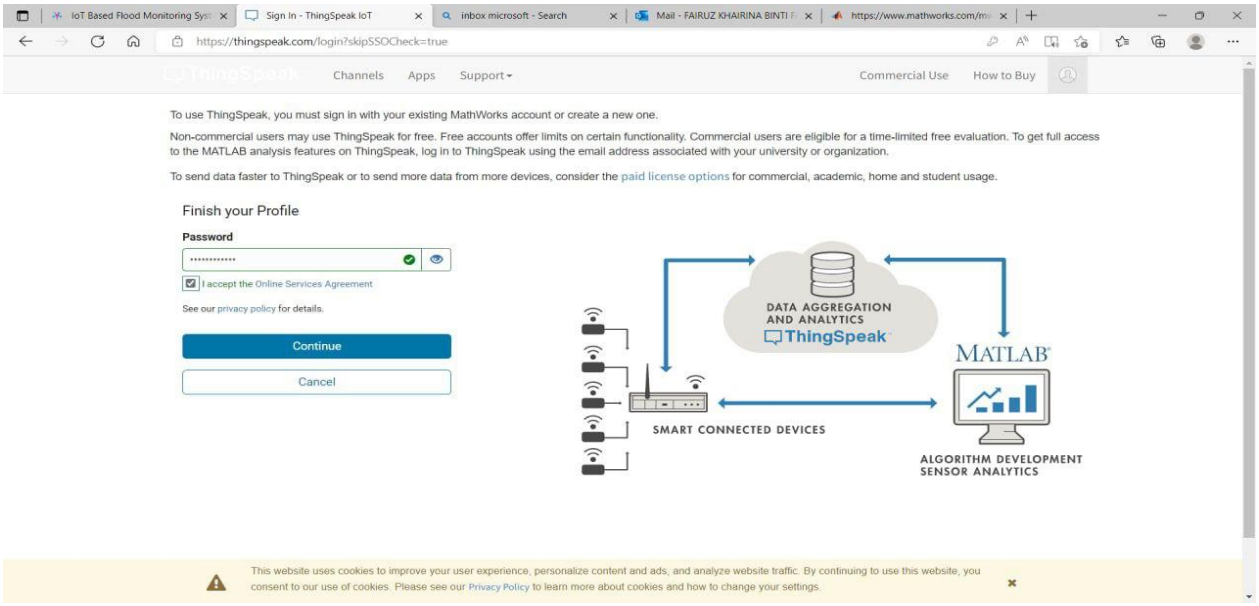
Set up a ThingSpeak account for Flood Monitoring & Alerting System

Step 1: Sign up for ThingSpeak

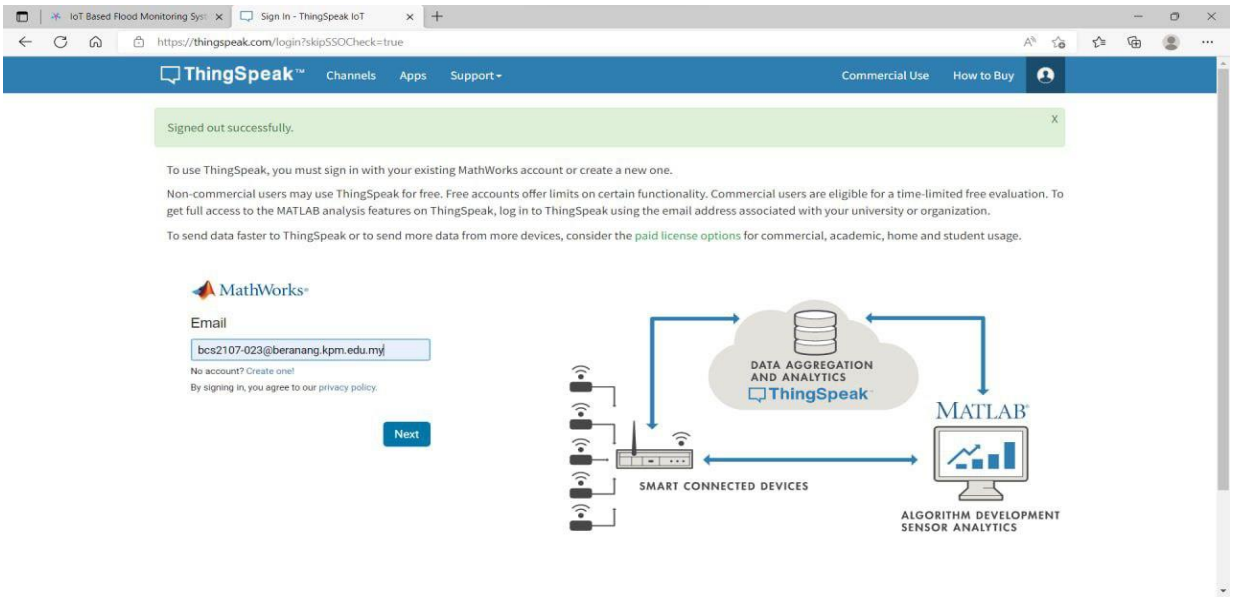
If do not have a MathWorks account, the first step is to go to ThingSpeak and set up a new free MathWorks account there.



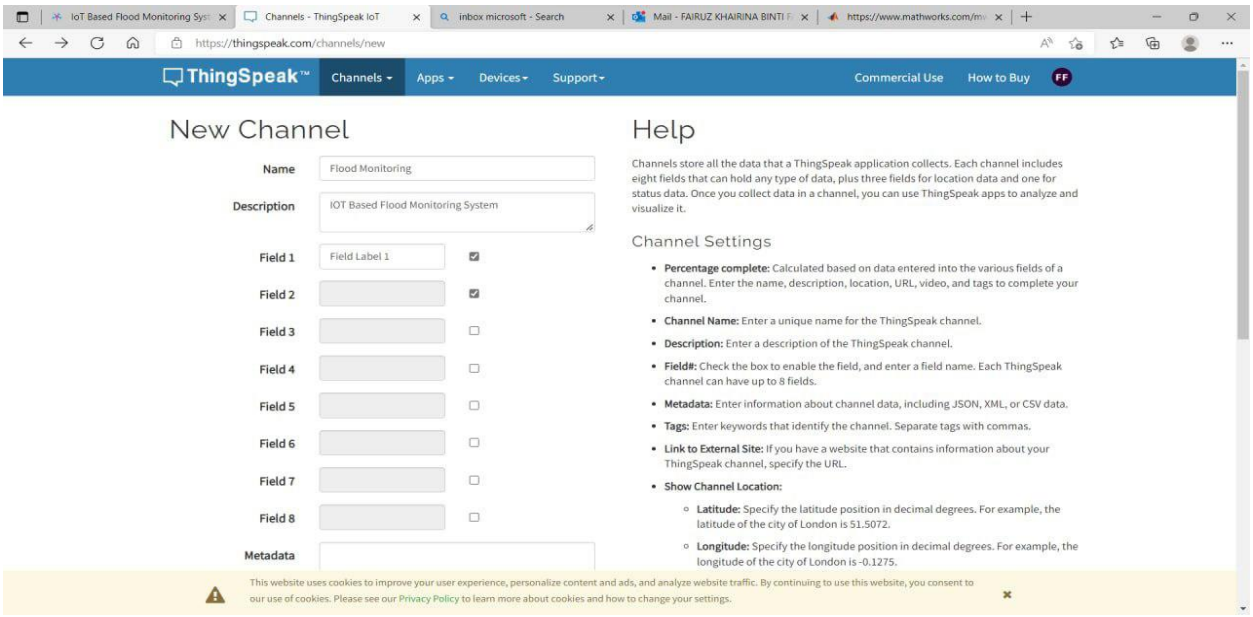
Sign up details at MathWorks to get access into cloud



Create an account password for the cloud account

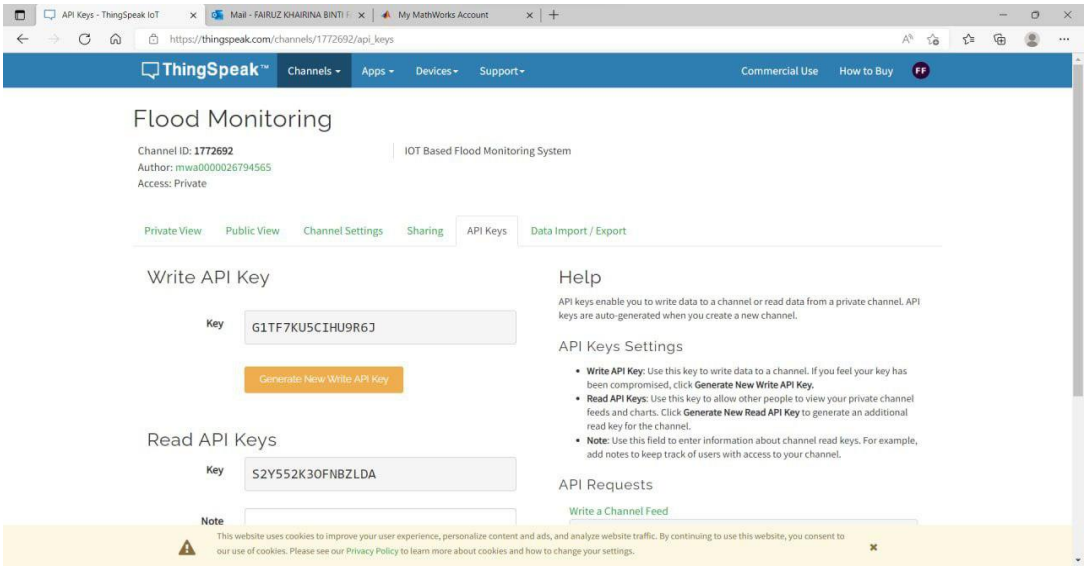


Step 2: Sign in to ThingSpeak



Step 3: Create “New Channel“.

Now, fill in the specifics of the project, such as its name and the names of the fields. At this point, we need to come up with titles for the fields, such as water level and status. After that, click the "Save Channel" button.



Step 4: Record the credentials

Select the created channel and record the following credentials.

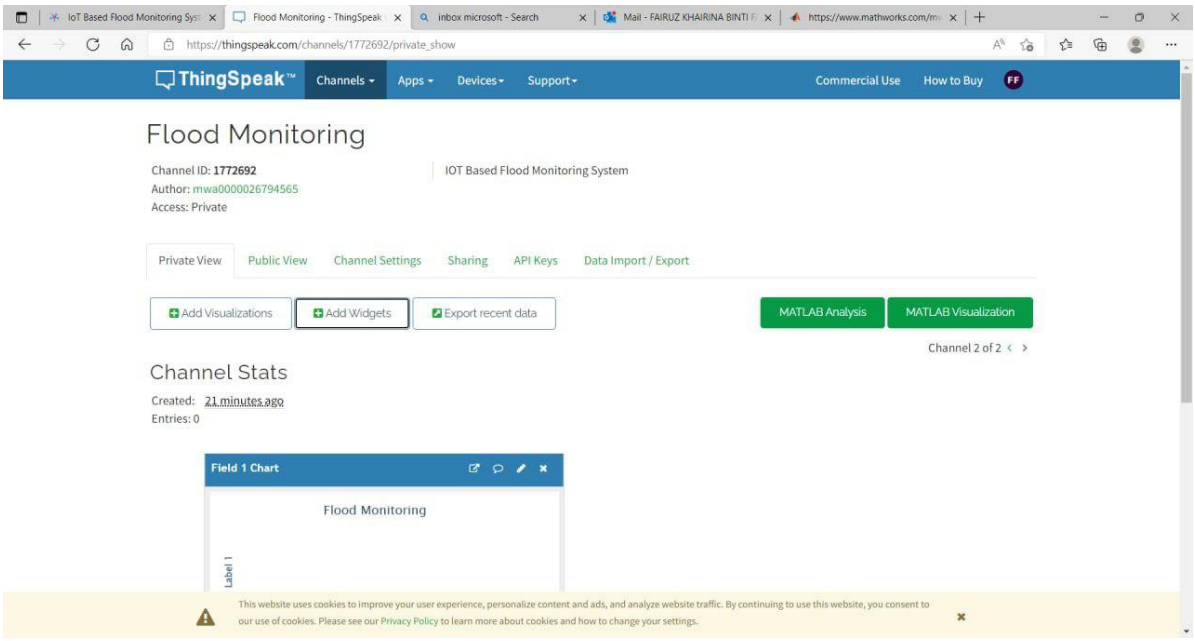
Channel ID, which is at the top of the channel view. API key, which can be found in the API Key tab of your channel view.

Channel ID: 1772692

Author: mwa0000026794565

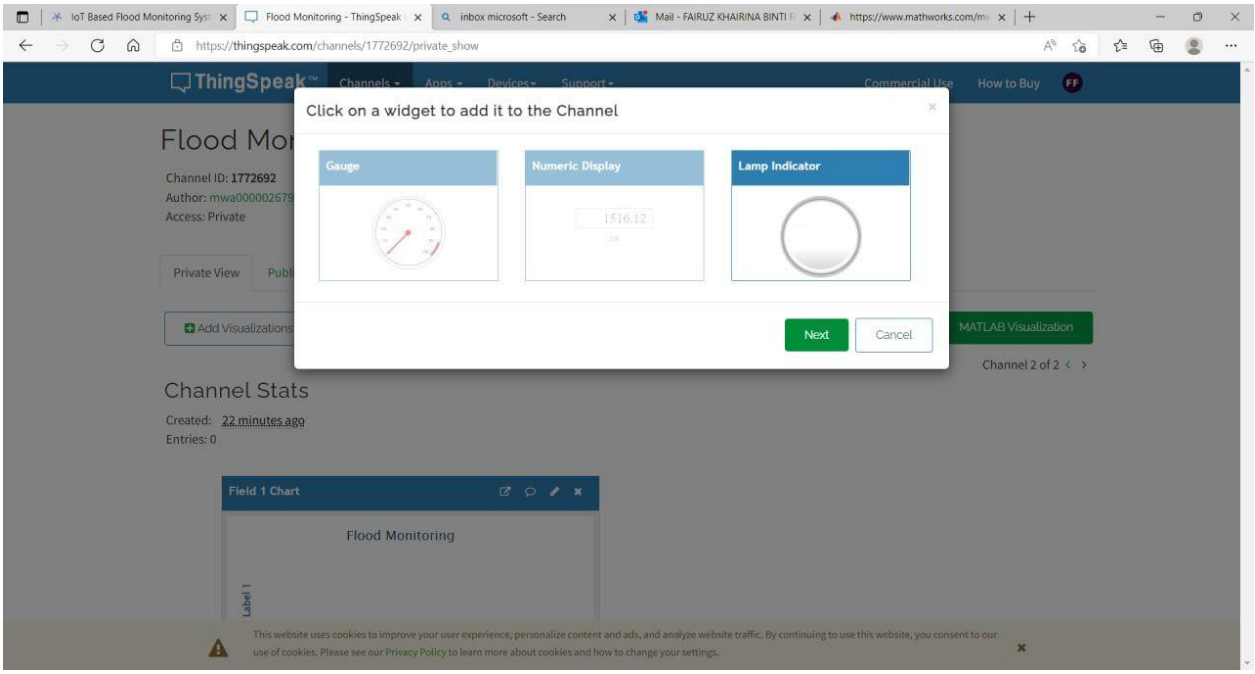
write api key: G1TF7KU5CIHU9R6J

read api key: S2Y552K3OFNBZLDA

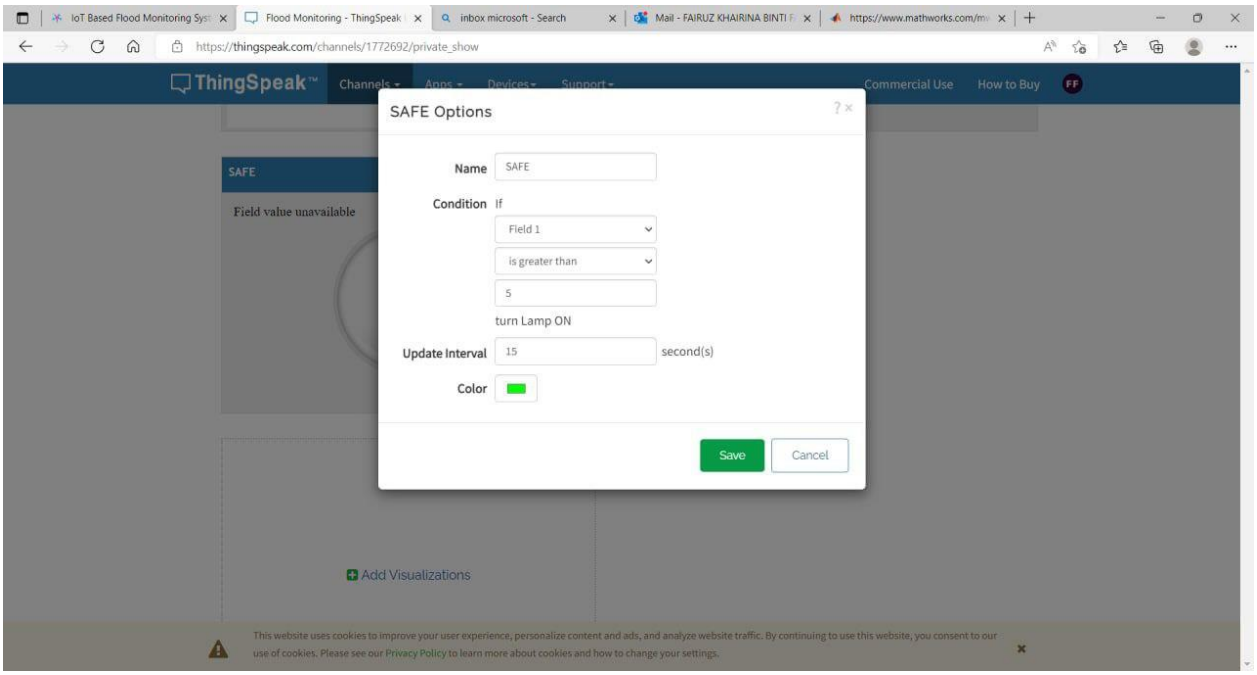


Step 5: Add widgets to your GUI

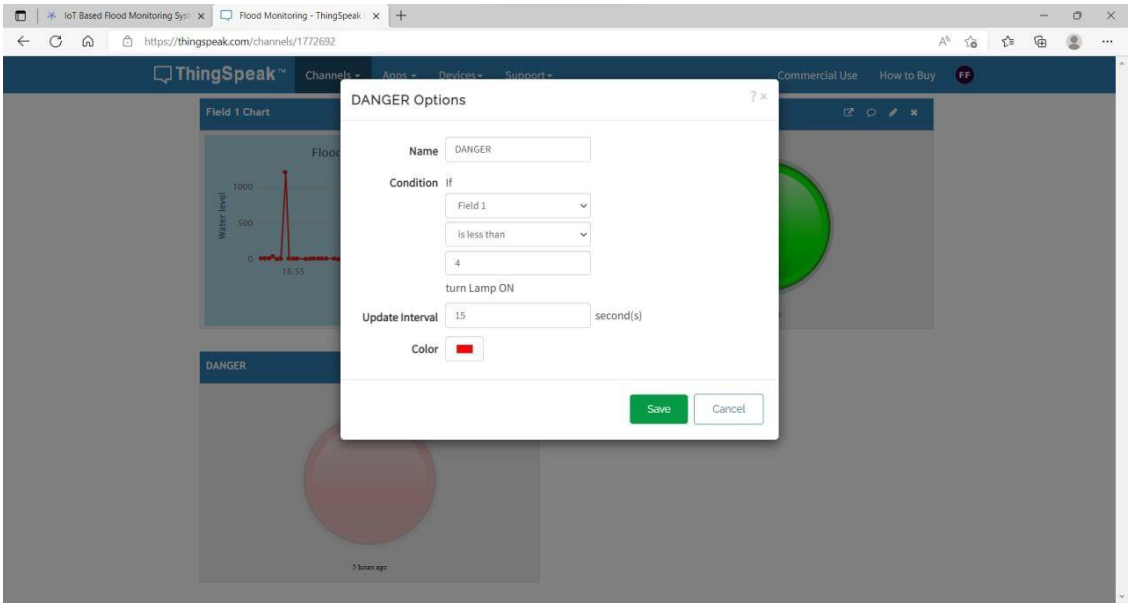
Click “Add Widgets”



Add two relevant Indicator widgets to the widget. In this situation,the indicator of was taken



The flood indicator setting for safe zone which represent as green LED



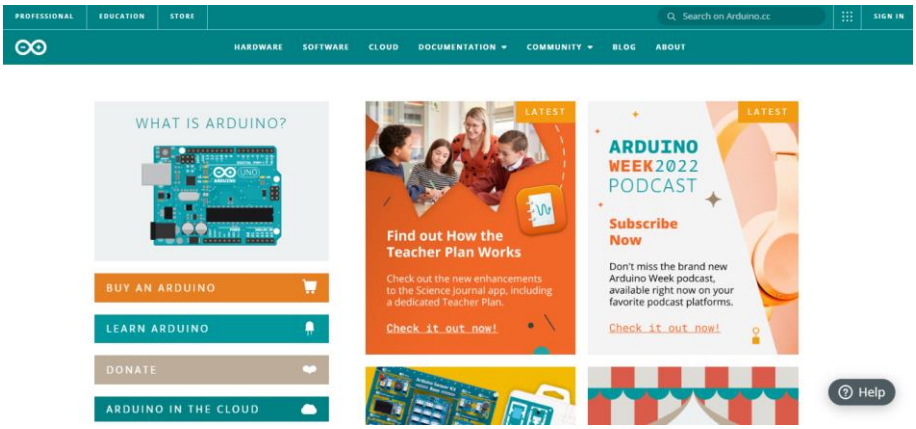
Choose the correct names for each of the widget's fields. as an example, SAFE would be used for green LED, whereas DANGEROUS would be used for red LED

3.10) Software Setup

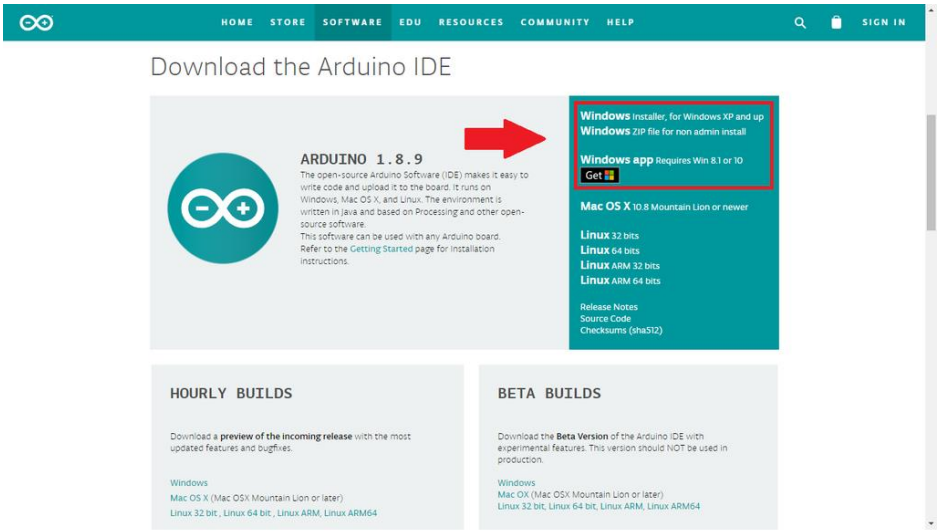
It is important to setup the software in order to keep systems software fully up to date. Otherwise, the systems may be vulnerable to newly discovered security flaws. So, these are the software that is going to setup for the actual finale project:

- **Download Arduino IDE**

STEP 1: First and foremost, go to Arduino website https://www.arduino.cc/ and click at the navigation bar that written “software”



STEP 2: Download File Arduino IDE

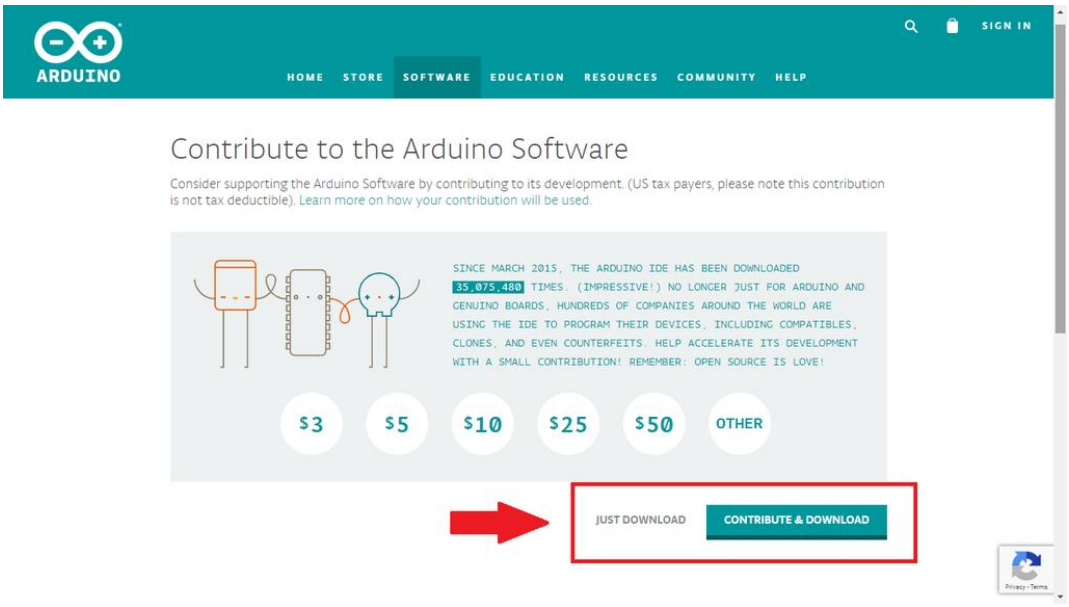


Once after click at the software page , there are 3 download options for Windows.

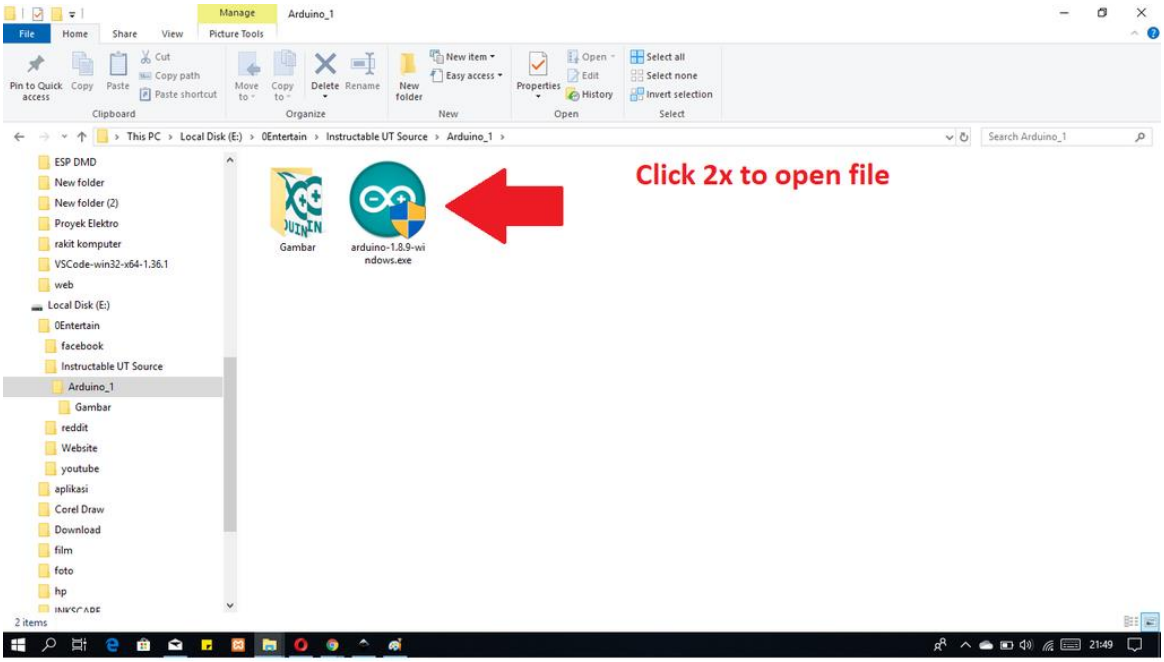
i. Windows Installer : The software will be installed in Windows operating system and required admin access.
ii. Windows Zip file : To make a portable installation.
iii. Windows App : for Windows 8.1 or 10.

Highly recommended to go for the first option . It is because it directly install everything needed and latest features that need to use the Arduino IDE software and include the drivers for the Arduino board. Choose the zip file if need to install the driver manually.

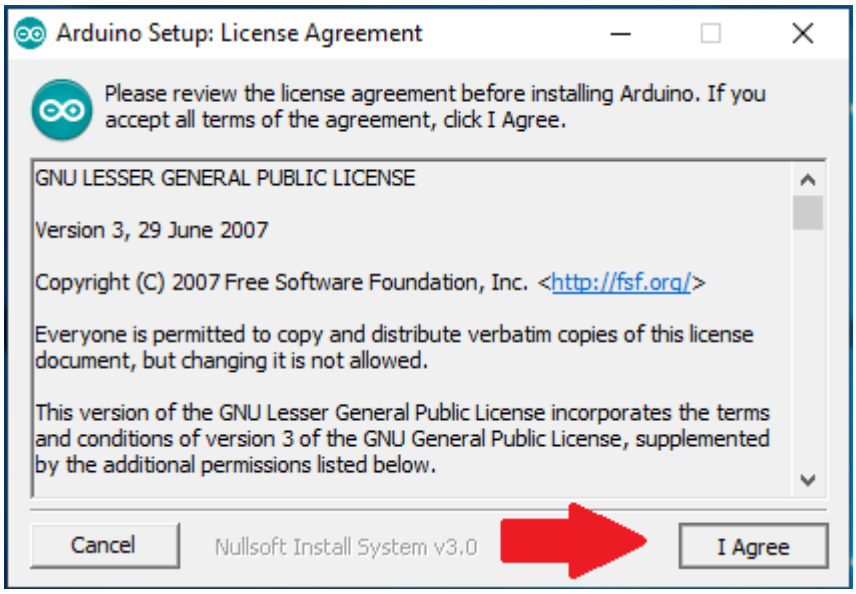
STEP 3: Click Windows Installer, then click "*just download*" for free version or "*Contribute & Download*" which may contribute for their development of the software.



STEP 4: After the download is complete, open the installer file and start installing.

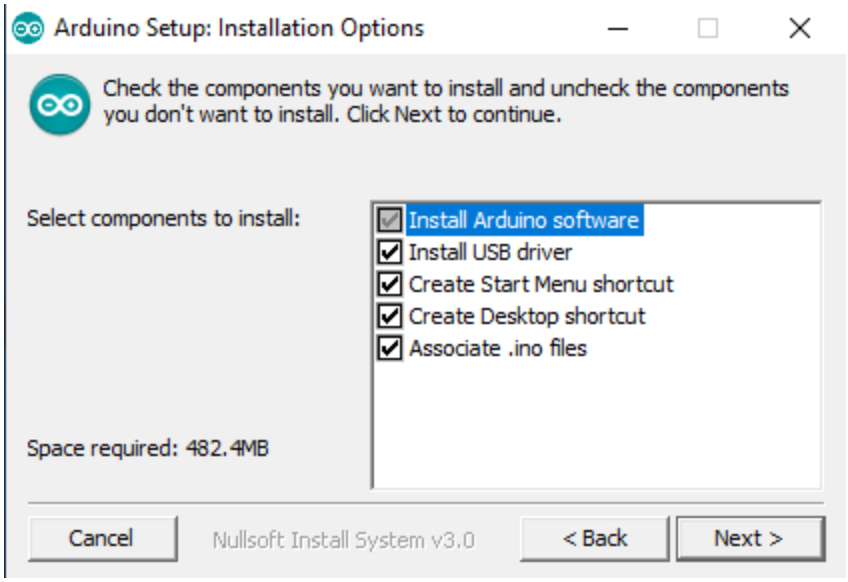


STEP 5 : Licence Agreement.



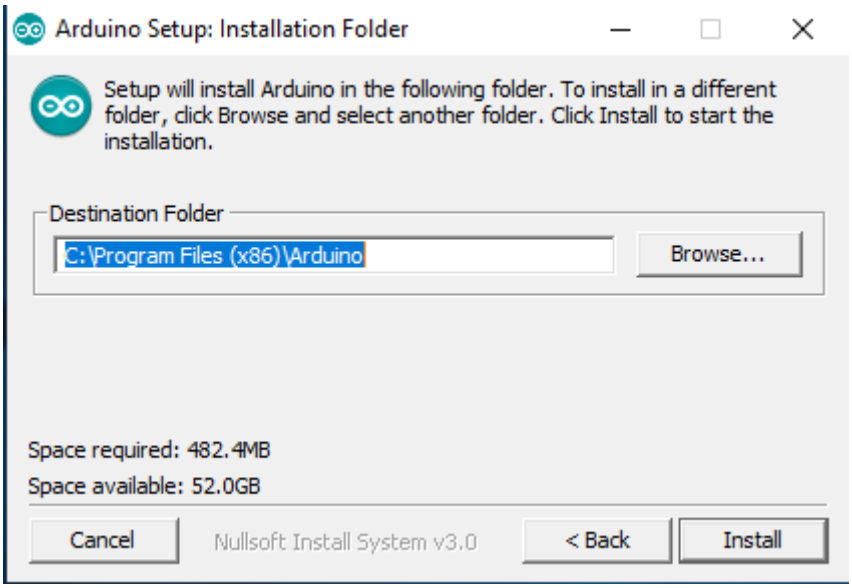
After the file is run, the "License Agreement" page will apper. User can read it, then click " *I Agree*" to continue.

STEP 6: Installation Option



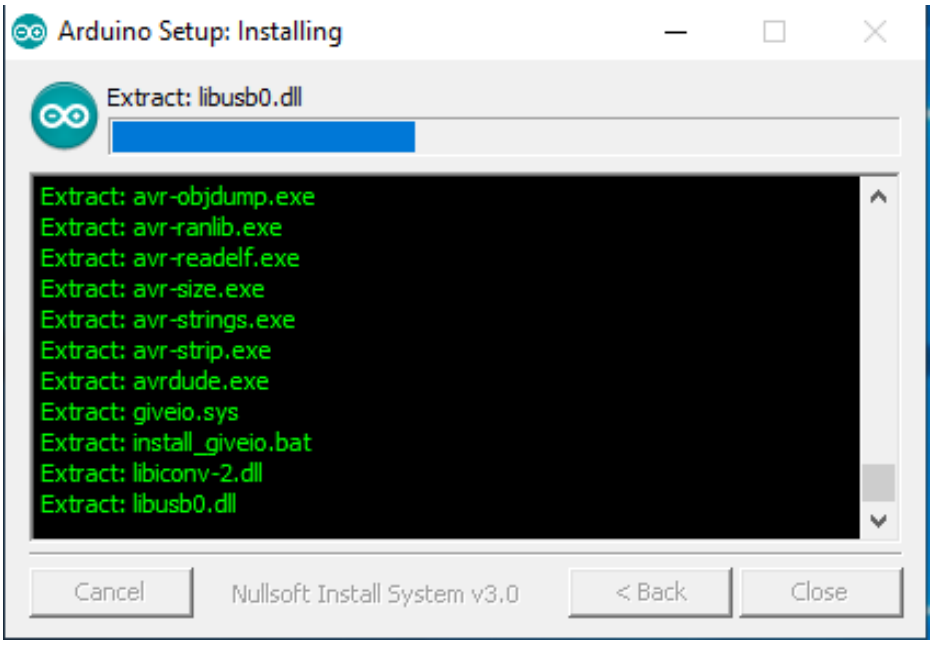
Check the component that want to install and uncheck the components that don't want to install. It was more suggested to installing all component provided. Click "*next*" to continue.

STEP 7 : Installation Folder



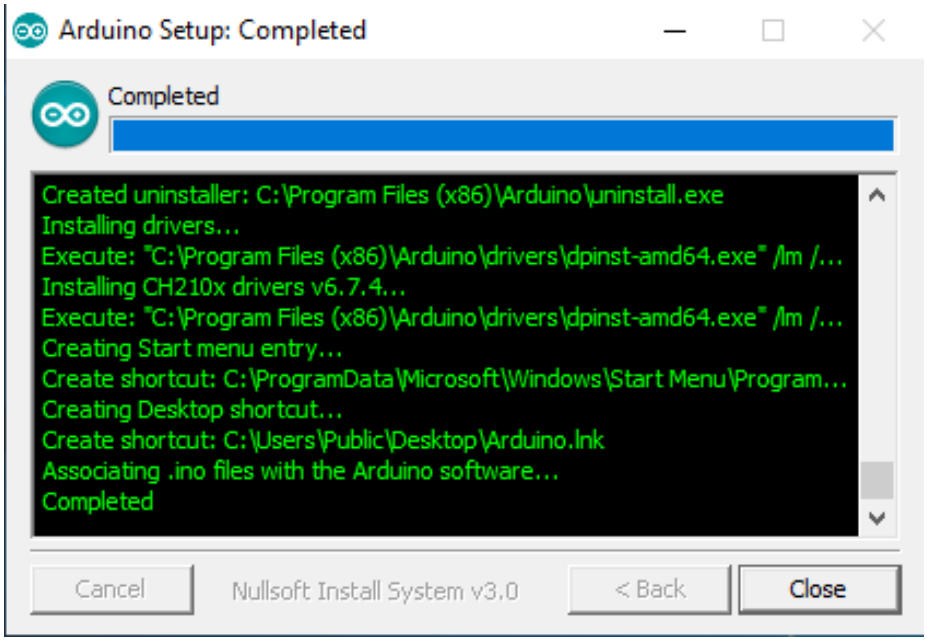
Arduino will automatically installed in "**C:\Program Files (x86)\Arduino**". If user want to change the folder, click "*Browse*" and select the desired folder. Click install to start the installation.

STEP 8: Installing Process



The installation process is ongoing.

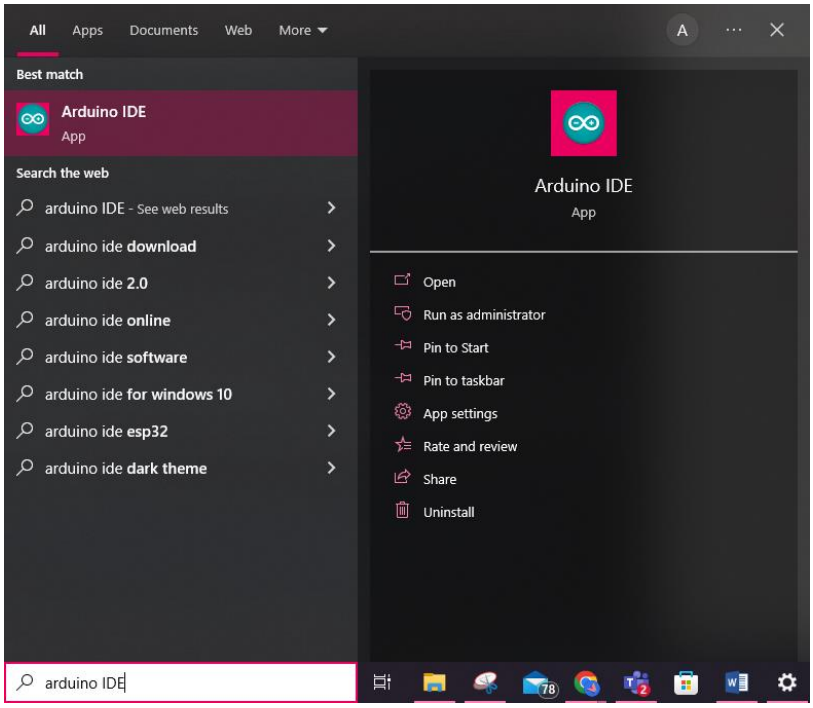
STEP 9 : Installation Complete



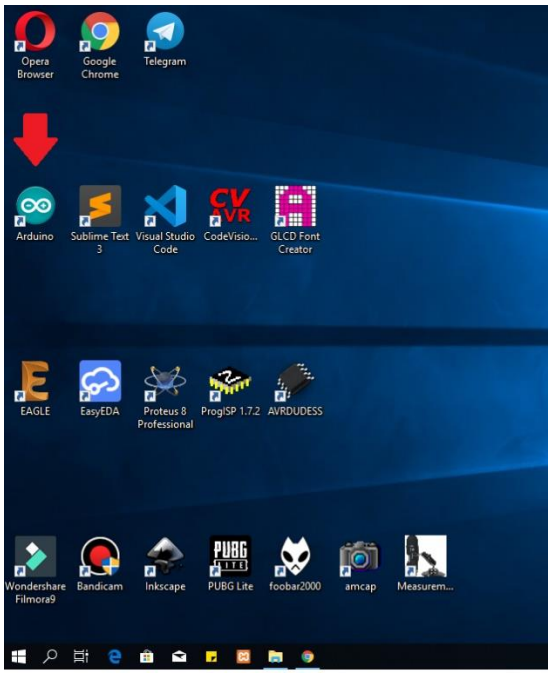
If there is written "*complete*", it means that the isntallation process is complete. click "*Close*".

STEP 10 : Open Arduino IDE

After the installation process is complete, there will be an Arduino icon on the Desktop. Or check on the search icon and write "arduino". If have found the arduino icon, run the application.



Arduino IDE at the search icon



Arduino IDE on Desktop

STEP 11: Display Arduino IDE

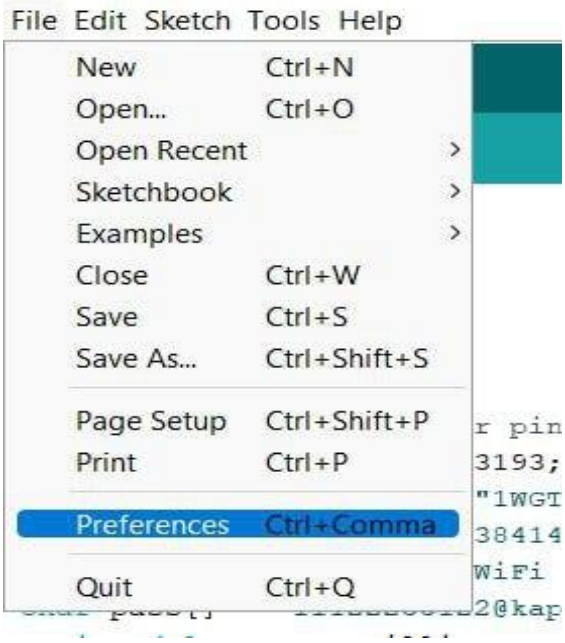
Lastly, This is a display of the Arduino IDE Software. The application is ready to be used to create an amazing projects.

- **Setting Arduino IDE for NodeMCU Board**

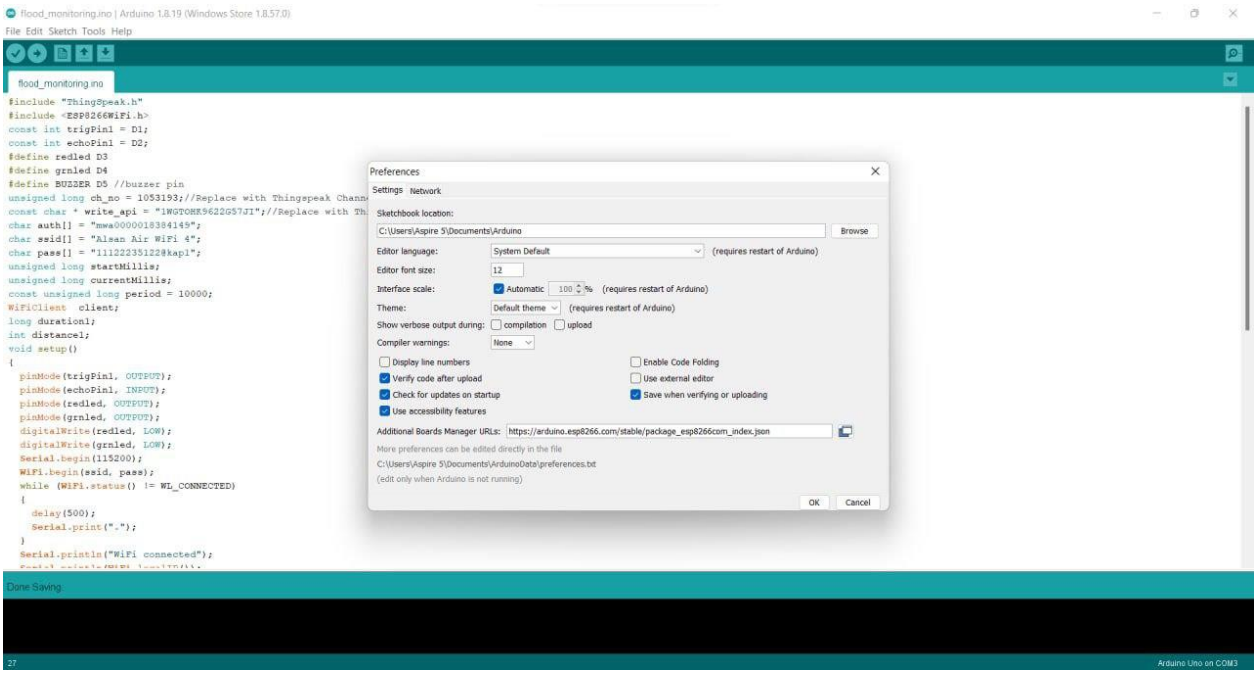
After the hardware installation has been completed successfully. Writing code for the ESP8266 NodeMCU is the next stage.

The following steps need to be taken in order to upload the code to NodeMCU using the Arduino IDE:

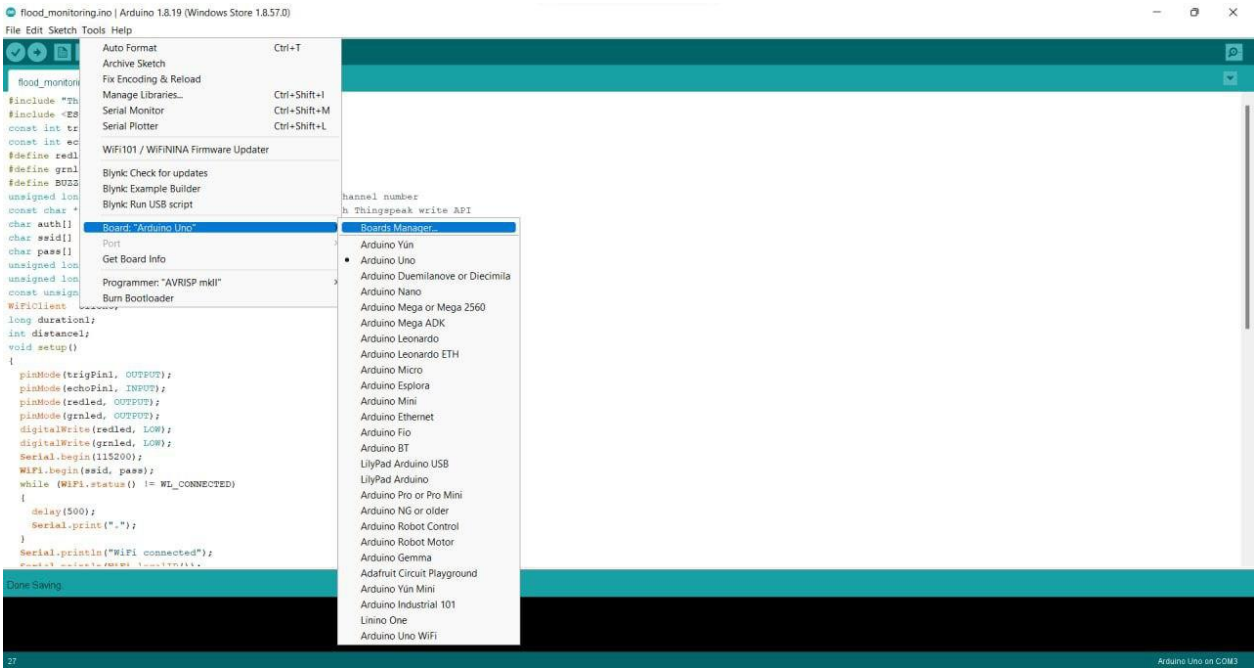
STEP 1 : Open the Ardino IDE, then go to File> Preferences> Settings.



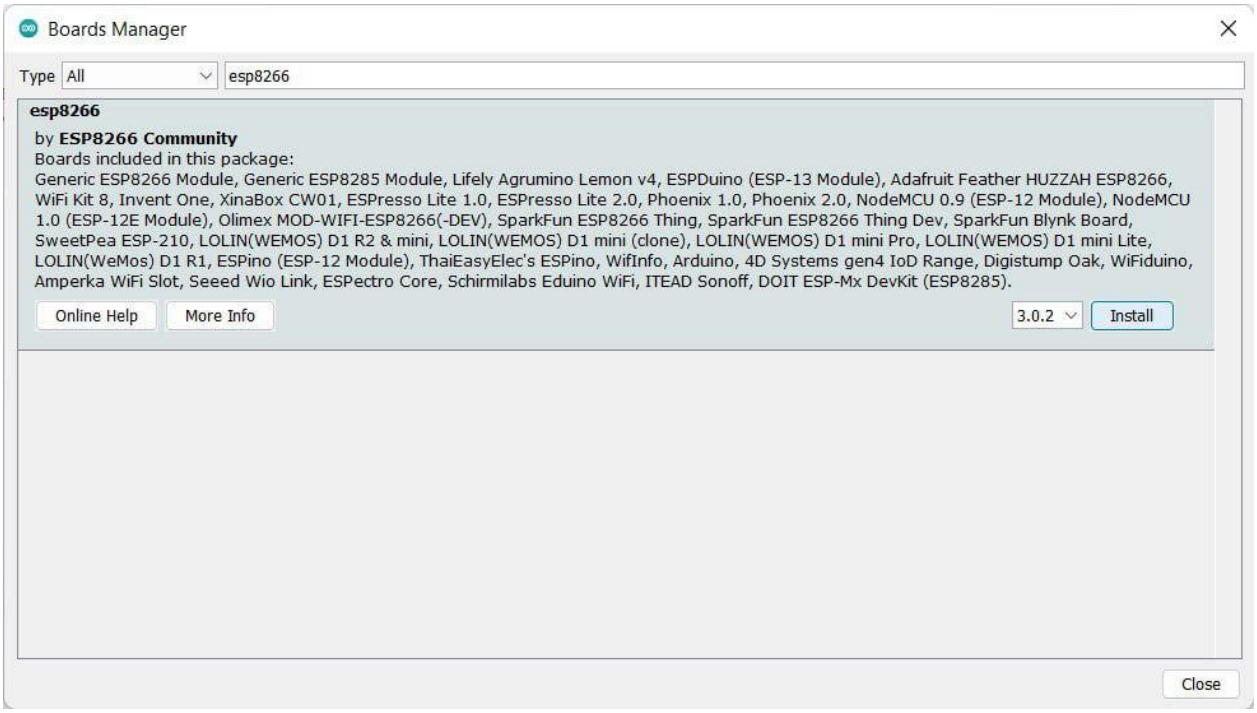
STEP 2: Paste the URL: https://arduino.esp8266.com/stable/package_esp8266com_index.json the 'Additional Board Manager URL' field and click 'Ok'.



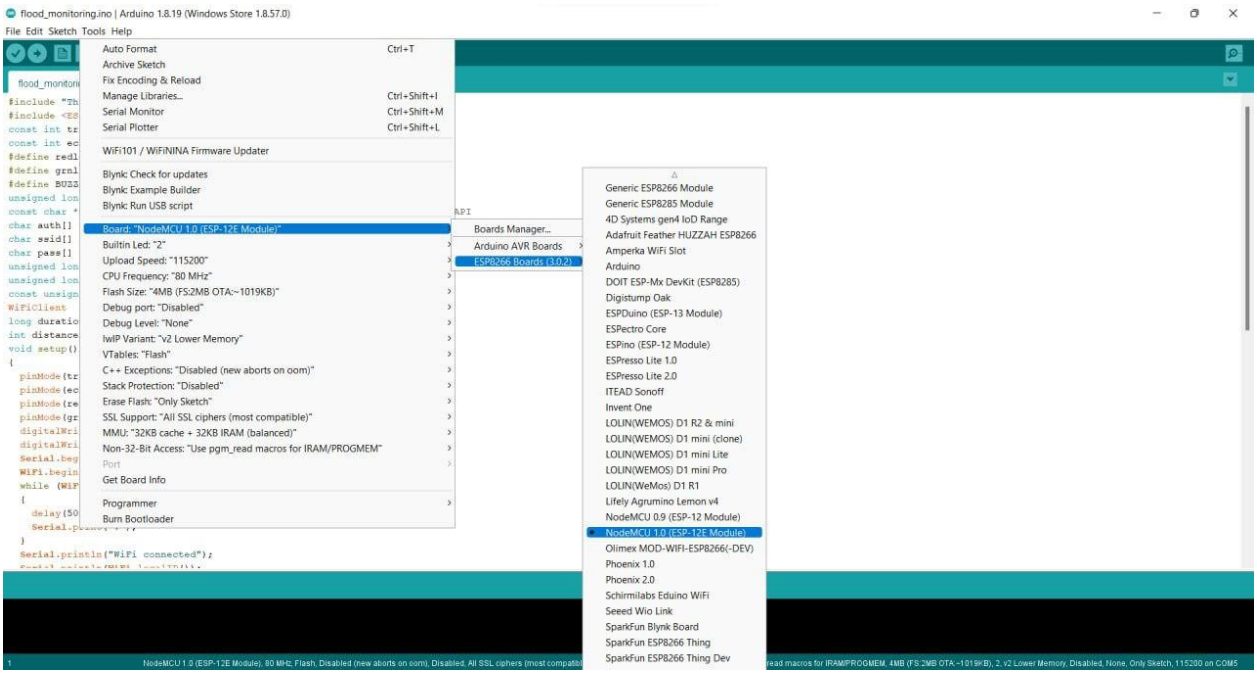
Step 3: Now. Now go to Tools> Board> Board Manager.



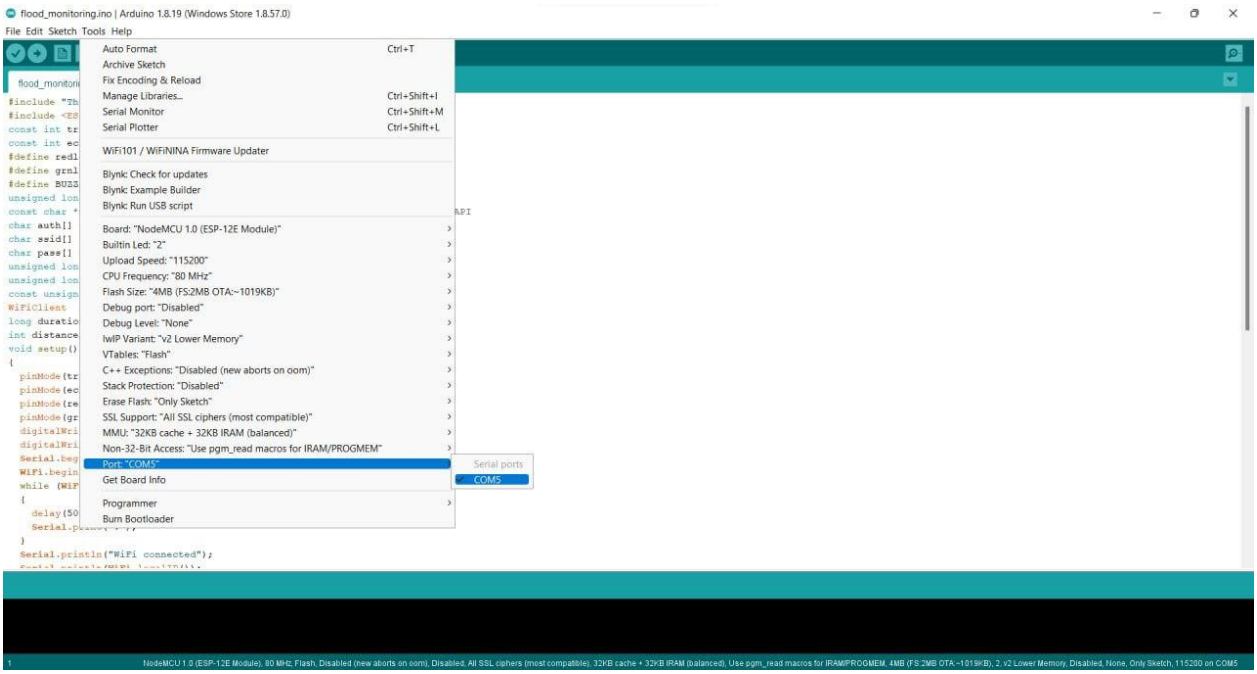
STEP 4: In the Boards Manager window, type ESP8266 in the search box, select the new version of the board and click Install.



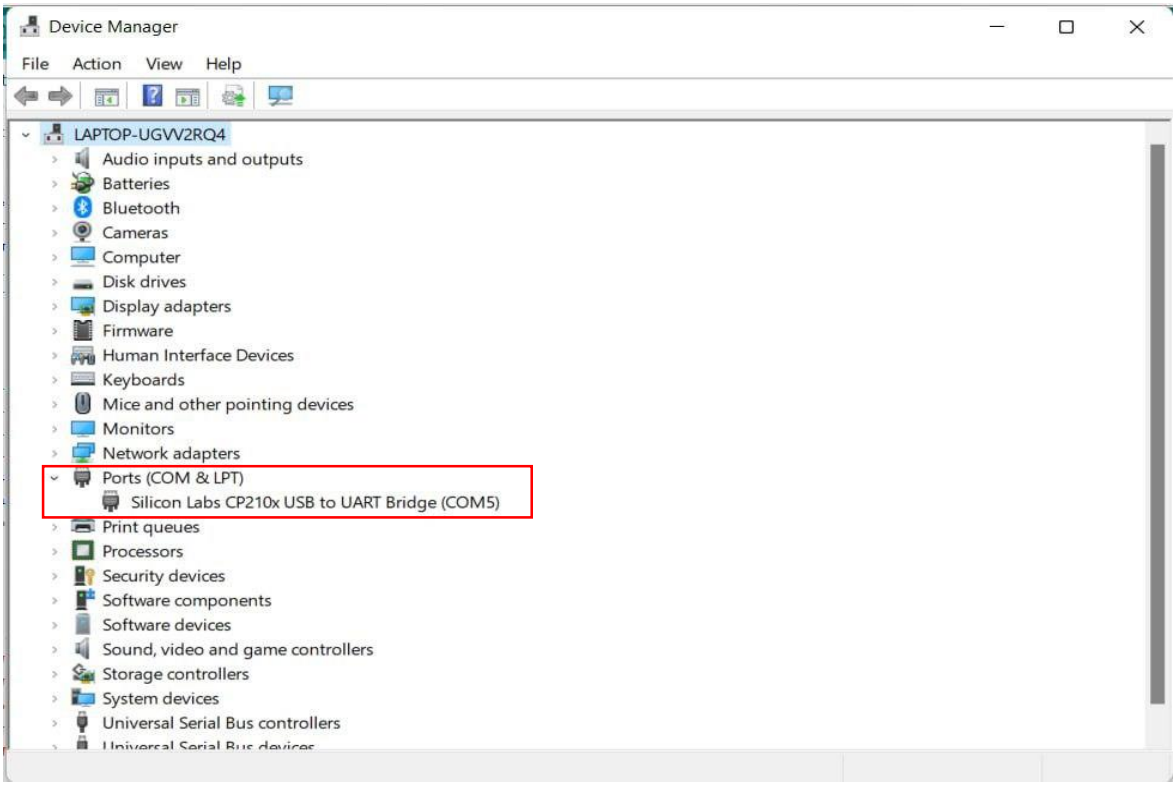
STEP 5: After successful installation, go to Tools -> Board -> and select NodeMCU 1.0 (ESP-12E Module). Now you can program NodeMCU with Ardino IDE.



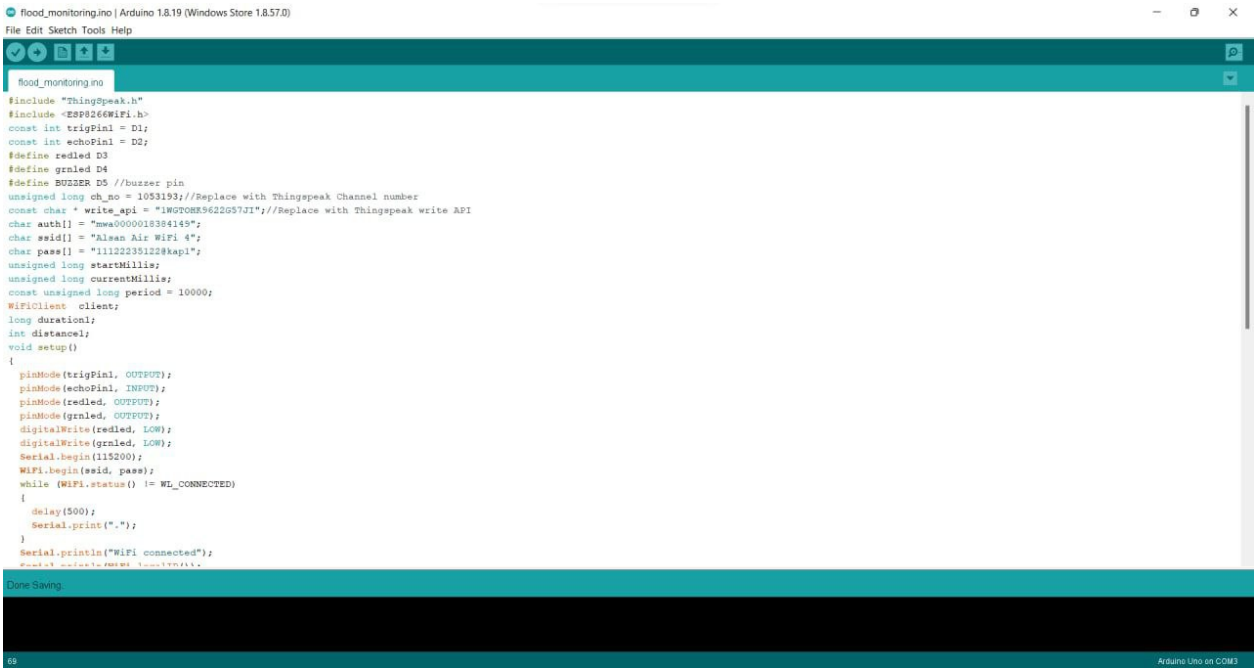
Step 6: click tools>port and select your port



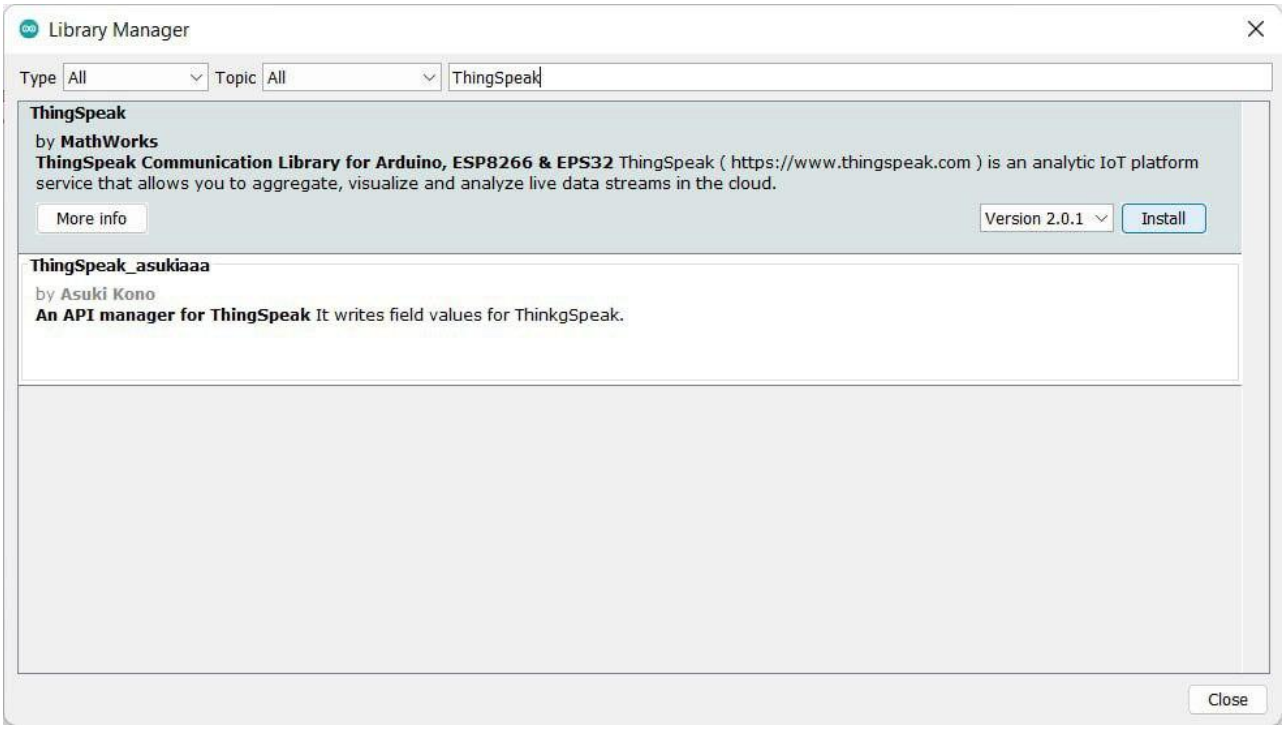
Step 7: Check the port in device manager . For example , for certain user laptop it will appear com1 , com2 , com3, com4 ,com5 and etc.



STEP 8: After the above setup for programming NodeMCU. User can upload the complete code to ESP8266 NodeMCU. The step-by-step explanation of the full code is provided below.



- **Install Thingspeak library in Arduino IDE**



Begin by adding all of the essential library files into the ESP8266 code.

The ThingSpeak platform makes use of the ThingSpeak.h library. The following steps can be used to add it to the Arduino IDE:

Step 1: Select Sketch/Include Library/Manage Libraries in the Arduino IDE.

Step 2: Click the ThingSpeak Library from the list

Step 3: click button install.

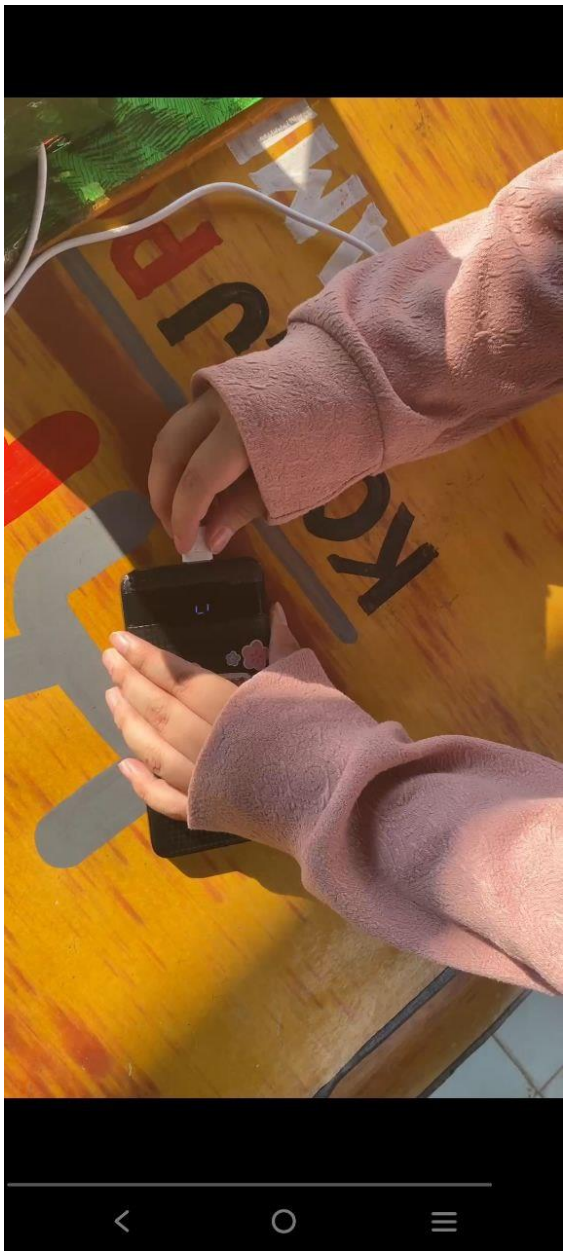
(Disclaimer : The Arduino IDE coding will be show at table of contents number 4.0)

3.11) Demonstration on How Flood Monitoring Works

STEP 1 : Upload the Arduino coding from Arduino IDE into the NodeMCU ESP 8266

Make sure use the nearest WI-FI in order to connect with the hardware

STEP 2 : After upload the code, connect the the USB Cable Micro B with power bank in order to make it wireless



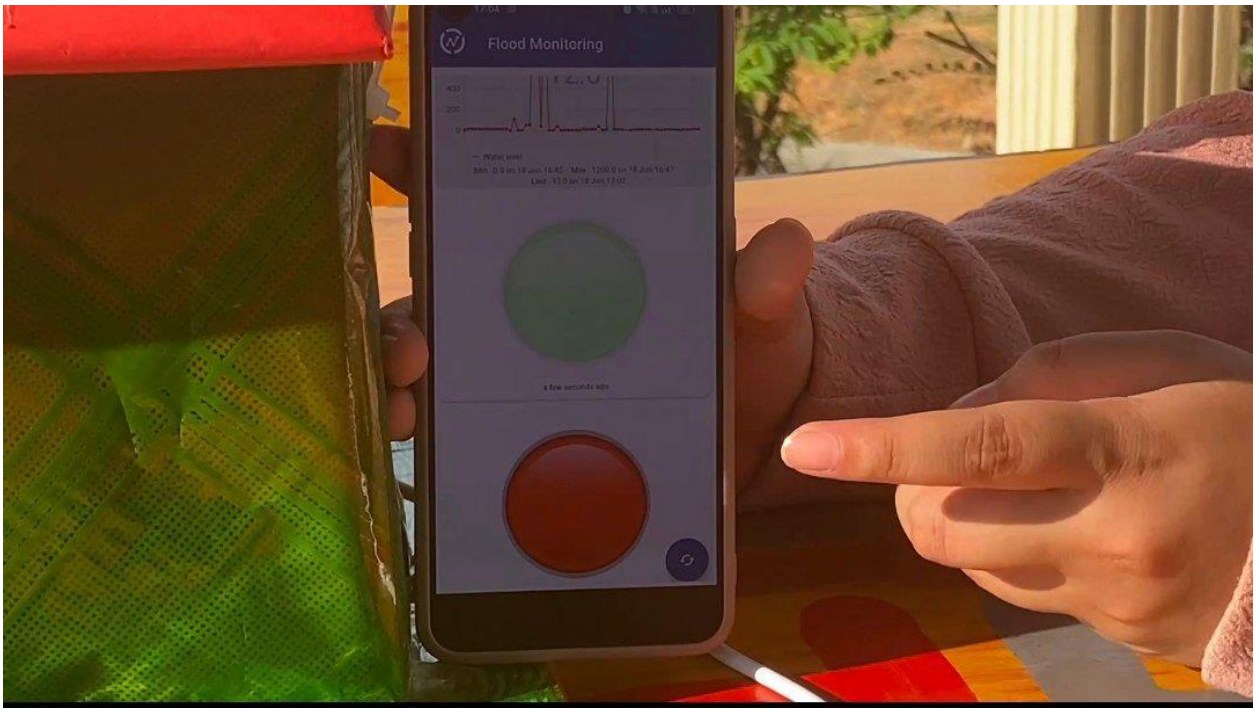
STEP 3 : Pour the amount of water that indicates the safe zone level

The water level shows green LED indicates its still safe

User able to monitor the flood level by using the application

STEP 4: Pour more water into the prototype bucket until it indicates the dangerous zone

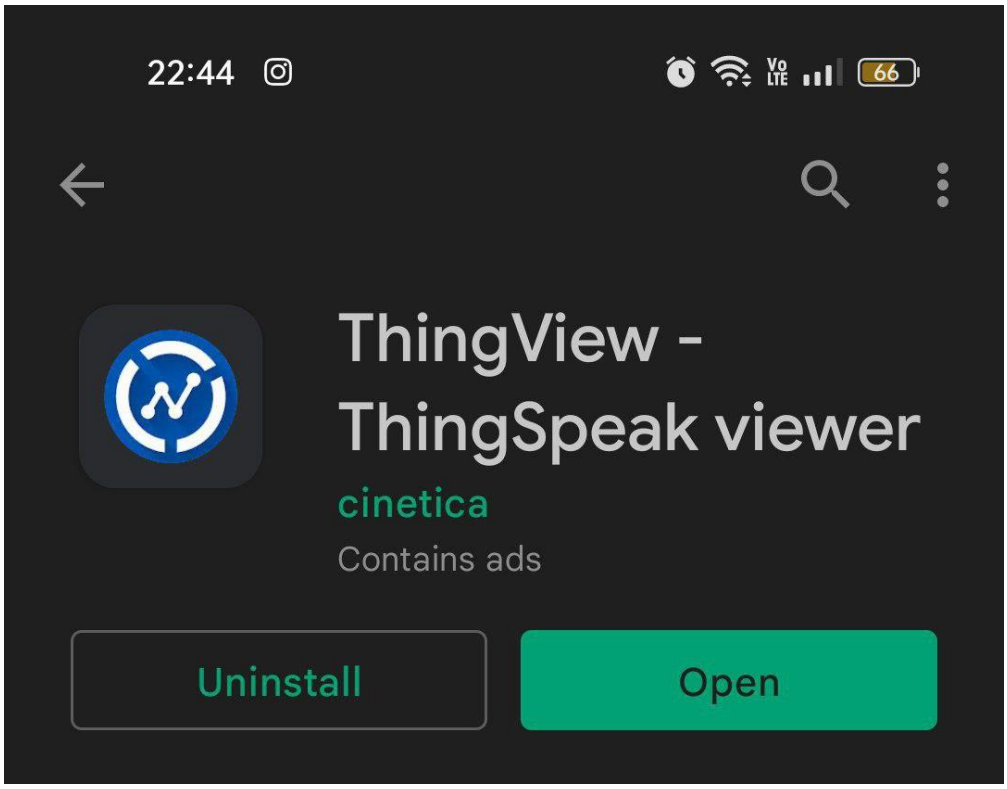
The red LED light indicates it was a dangerous zone . which means the distance between water level and ultrasonic sensor is really closed.



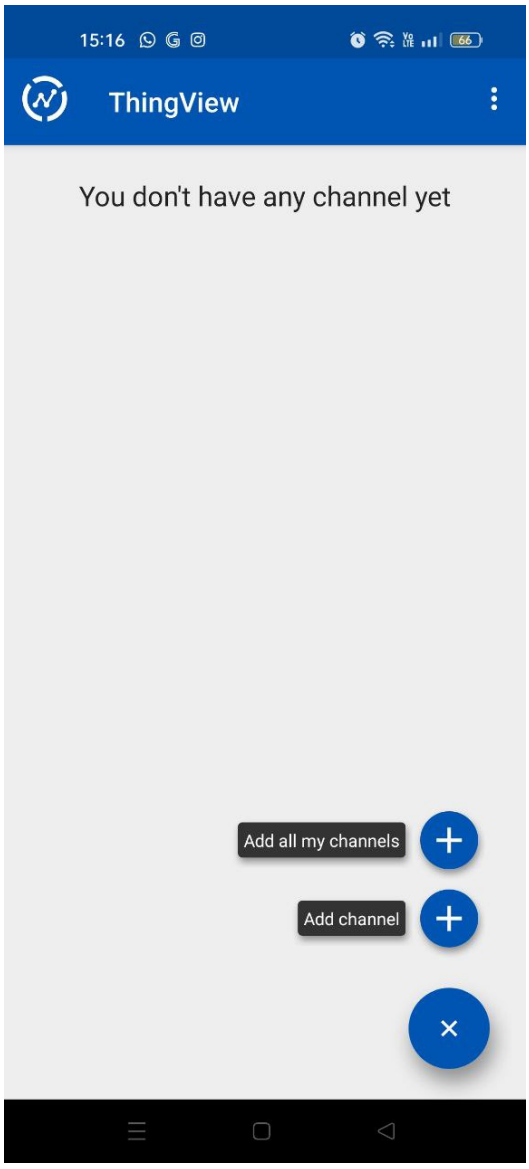
Once has reached to the dangerous zone, the buzzer will buzz loudly and the application shows that the red button has more pungent colour compare to green button colour,

3.12) How To Use The App

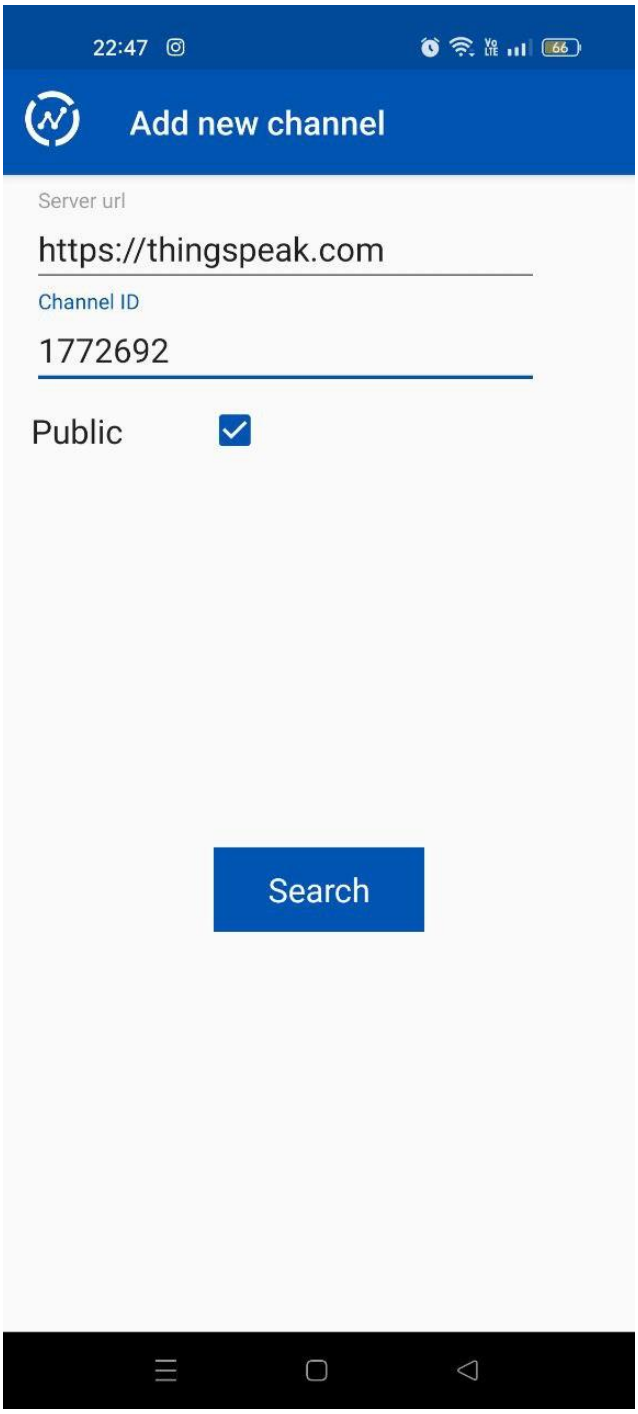
Step 1: Download ‘ThingView - ThingSpeak viewer’ at the Playstore



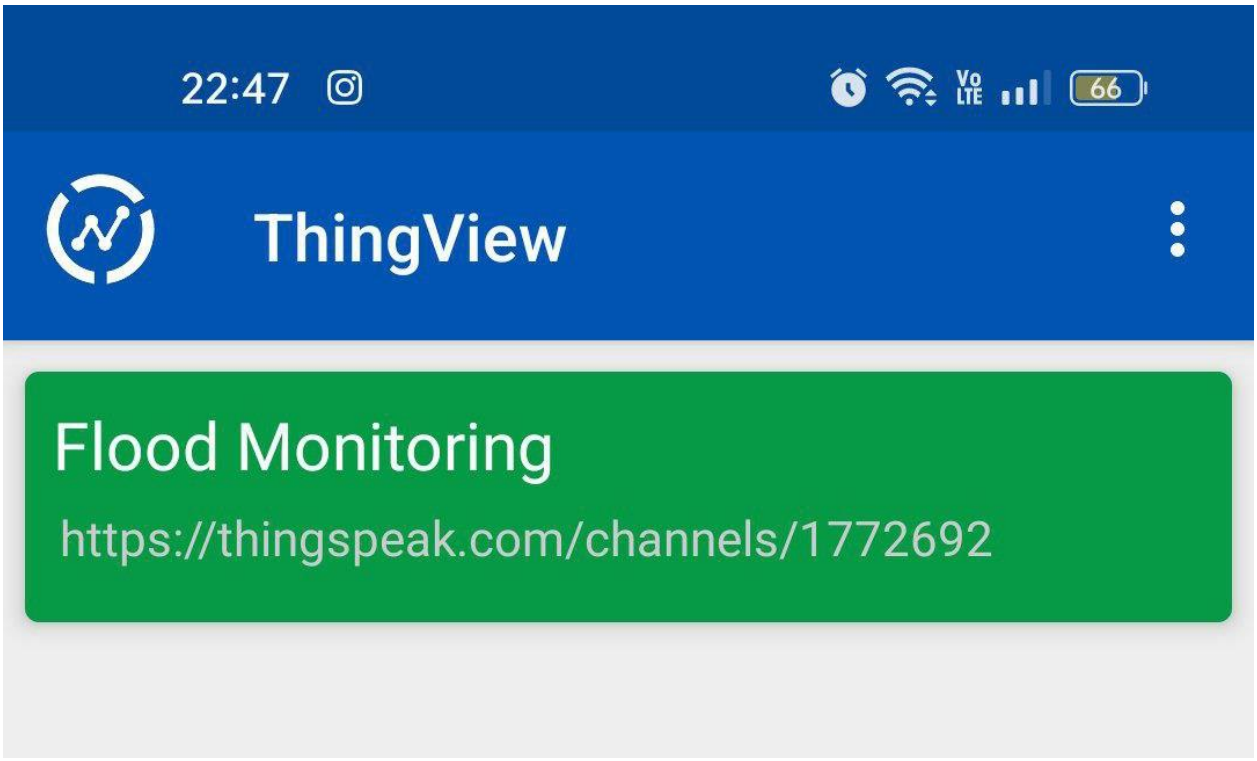
Step 2: Press add channel button at bottom



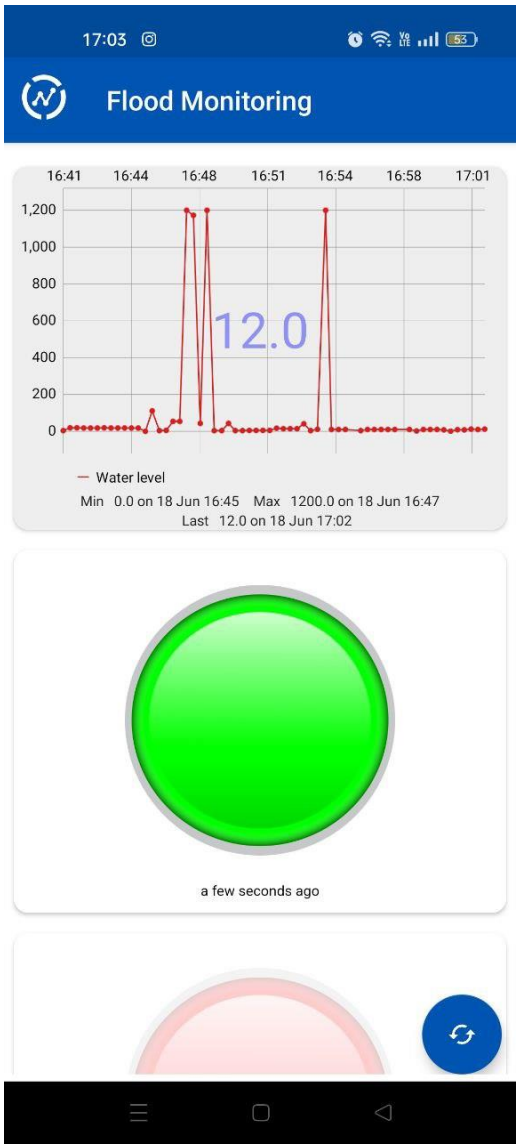
Step 3: Enter the channel ID and hit the search button.



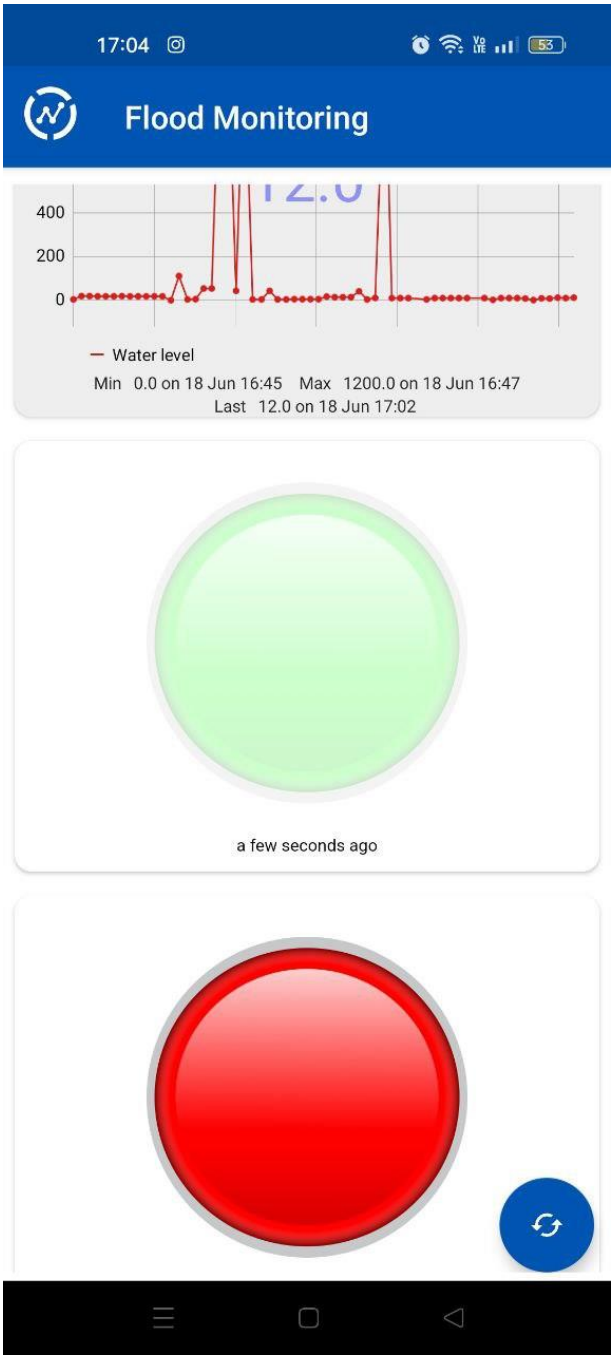
STEP 4 : Choose flood monitoring option.



STEP 5 : While there is no flood, ThingSpeak will display a large green indication. and a Red Indicator in the event of a Flood, as depicted below:

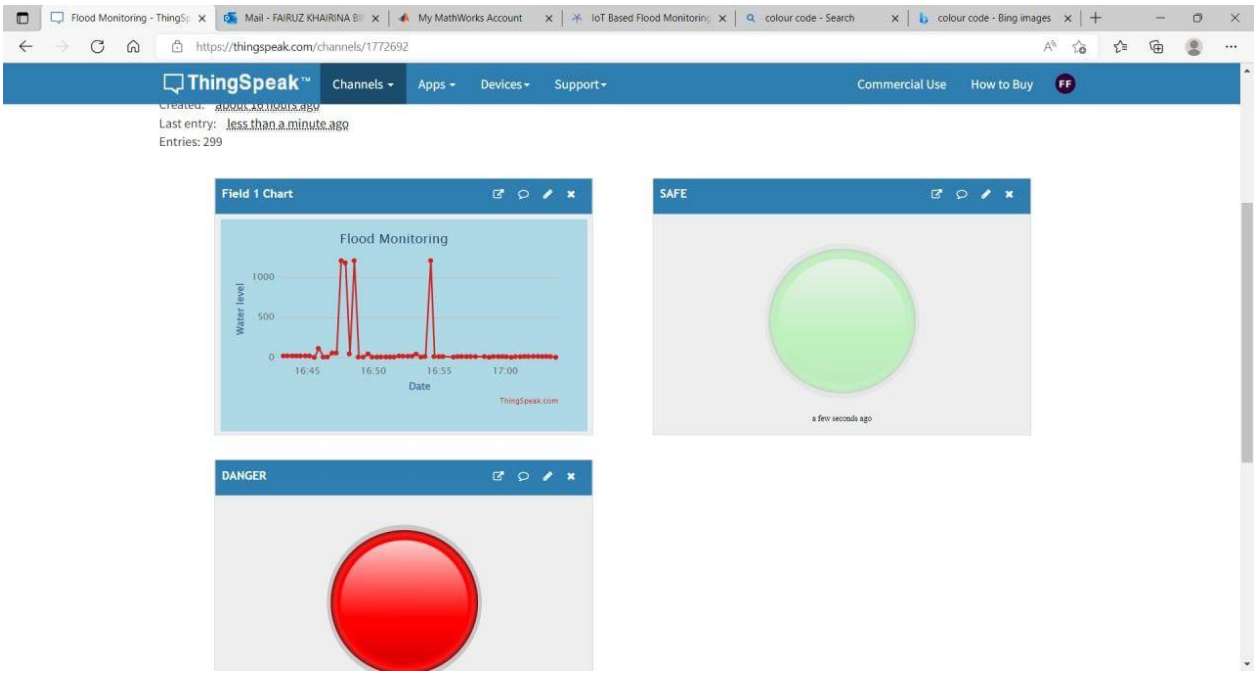


Green button indicate there is no flood occur or too minimal

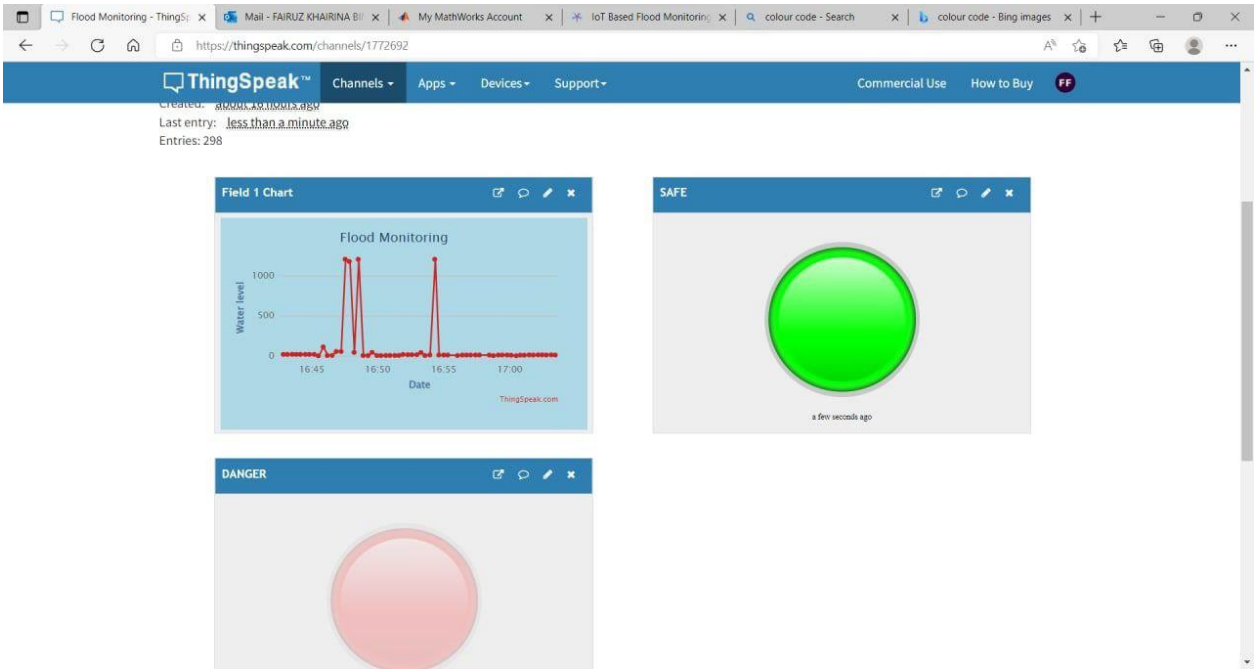


Red button indicate there was flood occur and people surrounding should be alert for any changes at that situation

- Other than apps, these are the interface when user used on PC:



Red button indicate there was flood occur and people surrounding should be alert for any changes at that situation



Green button indicate there is no flood occur or too minimal

The real-time graph at the top-left side to display the water level . It tells the minimum level that had ever recorded is 19 June at 05:20 P.M . Meawhile , The maximum level ever recorded is 20 Jun at 12:12 A.M

4.0) CODING IN ARDUINO IDE

The code for ESP8266 boards should begin by incorporating all of the essential library files. The ThingSpeak platform relies on the ThingSpeak.h library for its functionality. The following are the steps to add it to the Arduino IDE:

- In the Arduino IDE, select Sketch/Include Library/Manage Libraries from the list
- The ThingSpeak Library can be installed by clicking the Install option on the ThingSpeak Library.

STEP 1

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
```

STEP 2: The Ultrasonic sensor, Buzzer, and LEDs all have pins that must be defined next.

```
#define redled D3
#define grnled D4
#define BUZZER D5 //buzzer pin
```

STEP 3: Now, Enter the network credentials- i.e. SSID and password of user WiFi Network to connect the NodeMCU with the internet. Then the ThingSpeak account credentials: channel number, API Key, and Author Key. These all credentials were recorded while setting ThingSpeak IoT Platform. Hence, make sure, user have edited these credentials in place of these variables.

```
unsigned long ch_no = 1772692;//Replace with Thingspeak Channel number
const char * write_api = "G1TF7KU5CIHU9R6J";//Replace with Thingspeak write API
char auth[] = "mwa0000026794565";
char ssid[] = "MUHD06@unifi";
char pass[] = "6bdk@0323";
```

STEP 4: The variables are defined for timing purposes.

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 10000;
WiFiClient  client;
long duration1;
int distance1;
```

STEP 5: Basically, to connect NodeMCU to the internet, it calls **WiFi.begin** function. To Check for the successful network connection ***WiFi.status()*** is used. Finally, after a successful connection, we print a message on the Serial Monitor with the NodeMCU IP address.

```
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());
```

STEP 6: Now toconnect to the **ThingSpeak IoT Platform** using Provided credentials. User need to use *ThingSpeak.begin* function.

```
ThingSpeak.begin(client);
```

STEP 7: For calculating the distance, an input pulse is given to the sensor through the trig pin of the HC-SR04 sensor. Here, a 2-microsecond pulse is given, then from the echo pin, the output pulse of the sensor is read. Finally, the distance is calculated in cm.

```
digitalWrite(trigPin1, LOW);
delayMicroseconds(2);
digitalWrite(trigPin1, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin1, LOW);
duration1 = pulseIn(echoPin1, HIGH);
distance1 = duration1 * 0.034 / 2;
```

STEP 8: Then, an if-else condition is defined for the LED indications and Buzzer state for both Normal and Flood conditions. Here as per setup, I have taken 4 cm as reference. But, user can change it as per user requirements.

```
if (distance1 <= 4)
{
  digitalWrite(D3, HIGH);
  tone(BUZZER, 300);
  digitalWrite(D4, LOW);
  delay(500);
  noTone(BUZZER);
}
else
{
  digitalWrite(D4, HIGH);
  digitalWrite(D3, LOW);
}
```

STEP 9: Finally, the river water level is uploaded to the ThingSpeak channel in interval of 10 seconds.

```
currentMillis = millis();
if (currentMillis - startMillis >= period)
{
  ThingSpeak.setField(1, distance1);
  ThingSpeak.writeFields(ch_no, write_api);
  startMillis = currentMillis;
}
```

4.1) Print Screen of the coding


```
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());
  ThingSpeak.begin(client);
  startMillis = millis();  //initial start time
}
void loop()
{
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = duration1 * 0.034 / 2;
  Serial.println(distance1);
  if (distance1 <= 4)
  {
    digitalWrite(D3, HIGH);
    tone(BUZZER, 300);
    digitalWrite(D4, LOW);
    delay(500);
    noTone(BUZZER);
  }
```

flood_monitoring.ino | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help

flood_monitoring.ino

```
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = duration1 * 0.034 / 2;
  Serial.println(distance1);
  if (distance1 <= 4)
  {
    digitalWrite(D3, HIGH);
    tone(BUZZER, 300);
    digitalWrite(D4, LOW);
    delay(500);
    noTone(BUZZER);
  }
  else
  {
    digitalWrite(D4, HIGH);
    digitalWrite(D3, LOW);
  }
  currentMillis = millis();
  if (currentMillis - startMillis >= period)
  {
    ThingSpeak.setField(1, distance1);
    ThingSpeak.writeFields(ch_no, write_api);
    startMillis = currentMillis;
  }
}
```

```
Invalid library found in C:\Users\user\Documents\Arduino\libraries\BlynkUsbScript: no headers files (.h) found in C:\Users\user\Documents\Arduino\libraries\BlynkUsbScript
Invalid library found in C:\Users\user\Documents\Arduino\libraries\libraries: no headers files (.h) found in C:\Users\user\Documents\Arduino\libraries\libraries
Invalid library found in C:\Users\user\Documents\Arduino\libraries\tools: no headers files (.h) found in C:\Users\user\Documents\Arduino\libraries\tools
```

4.2) Real Coding

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
const int trigPin1 = D1;
const int echoPin1 = D2;
#define redled D3
#define grnled D4
#define BUZZER D5 //buzzer pin
unsigned long ch_no = 1772692;//Replace with Thingspeak Channel number
const char * write_api = "G1TF7KU5CIHU9R6J";//Replace with Thingspeak write API
char auth[] = "mwa0000026794565";
char ssid[] = "MUHD06@unifi";
char pass[] = "6bdk@0323";
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 10000;
WiFiClient  client;
long duration1;
int distance1;
void setup()
{
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(redled, OUTPUT);
  pinMode(grnled, OUTPUT);
  digitalWrite(redled, LOW);
  digitalWrite(grnled, LOW);
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
```

```
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());
  ThingSpeak.begin(client);
  startMillis = millis();  //initial start time
}
void loop()
{
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin1, LOW);
  duration1 = pulseIn(echoPin1, HIGH);
  distance1 = duration1 * 0.034 / 2;
  Serial.println(distance1);
  if (distance1 <= 4)
  {
    digitalWrite(D3, HIGH);
    tone(BUZZER, 300);
    digitalWrite(D4, LOW);
    delay(500);
    noTone(BUZZER);
  }
  else
  {
    digitalWrite(D4, HIGH);
    digitalWrite(D3, LOW);
  }
  currentMillis = millis();
  if (currentMillis - startMillis >= period)
  {
    ThingSpeak.setField(1, distance1);
    ThingSpeak.writeFields(ch_no, write_api);
    startMillis = currentMillis;
  }
}
```