MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ BÖLÜMÜ

ALGILAYICI AĞLAR

IoT based Flood Monitoring System Using NodeMCU

Contents:

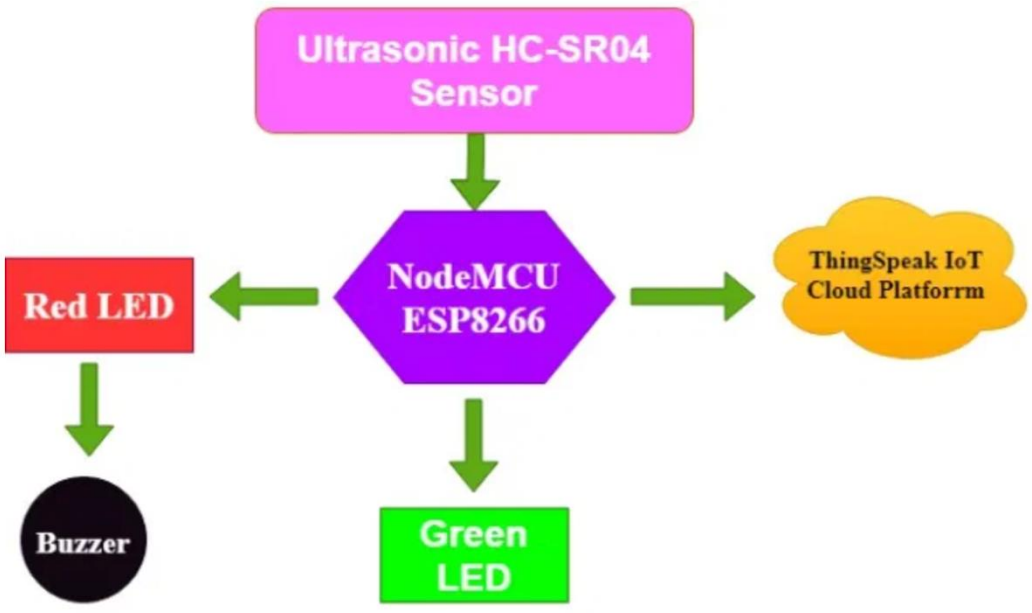
Today in this project we are going to make IoT Based Flood Monitoring System Using NodeMCU(ESP 32S), Ultrasonic HC-SR04 Sensor & Thingspeak & Database IoT Platform. As we all know that Flood is one of the major well Known Natural disasters. It causes a huge amount of loss to our environment and living beings as well. So in these cases, it is very important to get emergency alerts of the water level situation in different conditions in the river bed.

Required Components of Flood Monitoring System:

These are the components required to make IoT Based Flood Monitoring System Using NodeMCU & ThingSpeak:

- **NodeMCU ESP32S Development Board**
- **Ultrasonic HC-SR04 Sensor**
- **Red and Green LEDs**
- **Jumper wire**
- **Power Supply**
- **LocalHost (if you have a domain name it would be better)**
- Resistance
- Buzzer (if you want)

How this Flood Monitoring System Works?



We have used ESP8266 NodeMCU to build many IoT projects before. The block diagram above shows the working of this IoT based flood monitoring system using the NodeMCU and IoT Platform. Here, the Ultrasonic sensor is used to detect river water levels. The raw data from the ultrasonic sensor is fed to the NodeMCU, where it is processed and sent to ThingSpeack for graphical monitoring and critical alerts. Here, the red LED and Buzzer is used to send an alert in a flooded condition. While Green LED is used for indicating Normal condition. are used to be alert in severe flood conditions, and green LEDs are used to indicate normal conditions.

Ultrasonic HC-SR04 Sensor

Ultrasonic sensors work on the principle of ultrasound waves which are used to determine the distance for an object. An Ultrasonic sensor generates high-frequency sound waves. When this ultrasound hits the object, it reflects as the echo that the receiver sense. We can calculate the distance to the target object using the time required to reach the receiver for Echo.

Formula to calculate:
Distance= (Time x Speed of Sound in Air (340 m/s))/2

Ultrasonic distance sensors are of two ultrasonic transducers. One of them acts as a transmitter that converts the electrical pulse of the microcontroller into an ultrasonic sound pulse and is received by the receiver for transmitted pulses. If it receives them, then it produces an

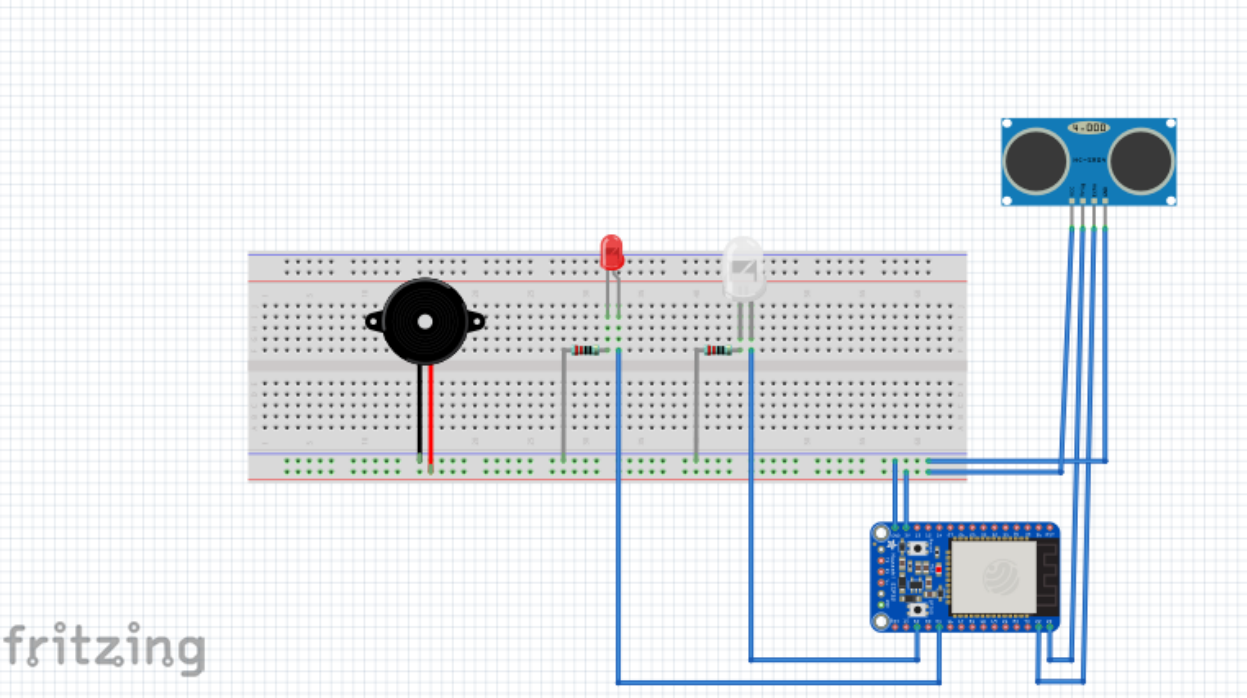
output pulse whose time period is used to determine the distance from the object.

Features:

- Working voltage: 5V
- Current working: 15mA
- Working frequency: 40HZ
- Measurement distance: 2cm - 4m
- Measuring Angle: 15 Degree.
- Trigging input pulse: 10uS

Schematic of IoT Based Flood Monitoring System

Interfacing NodeMCU ESP8266, Ultrasonic HC-SR04 Sensor, and LEDs (Red and Green) as shown in the following circuit diagram:



Interfacing HC-SR04 Sensor to NodeMCU

Ultrasonic HC-SR04	ESP8266
Vcc Pin	Vin Pin
Trig Pin	(GPIO 23)
Echo Pin	(GPIO 22)
GND Pin	GND

Red led -> gnd and GPIO 2
Green led -> gnd and GPIO 4
Don't forget the residence

// Buzzer
Buzzer -> gnd, 15 GPIO and vcc

Set up a ThingSpeak account for Flood Monitoring & Alerting System

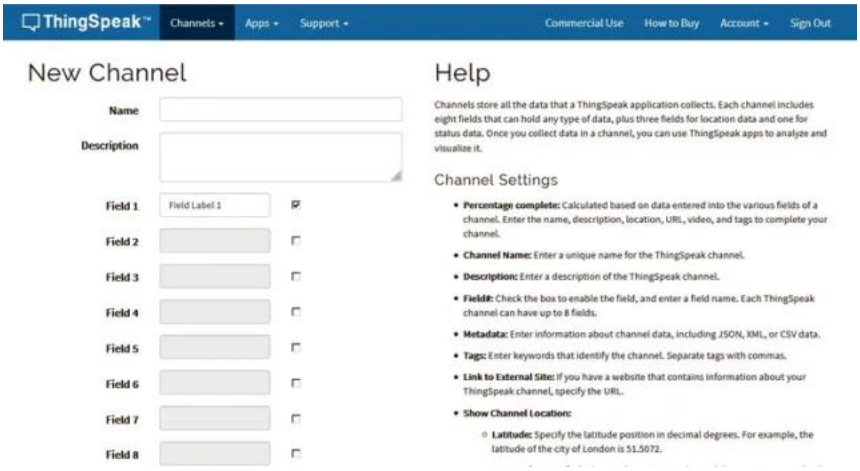
After the successful interface of the hardware parts according to the circuit diagram above. Now its time to set up the IoT platform, where data can be stored for online monitoring. Here we are using ThingSpeak to store data. ThingSpeak is a very popular IoT cloud platform that is used to store, monitor, and process data online.

Step 1: Sign up for ThingSpeak

First go to ThingSpeak and create a new free MathWorks account if you don't already have a MathWorks account.

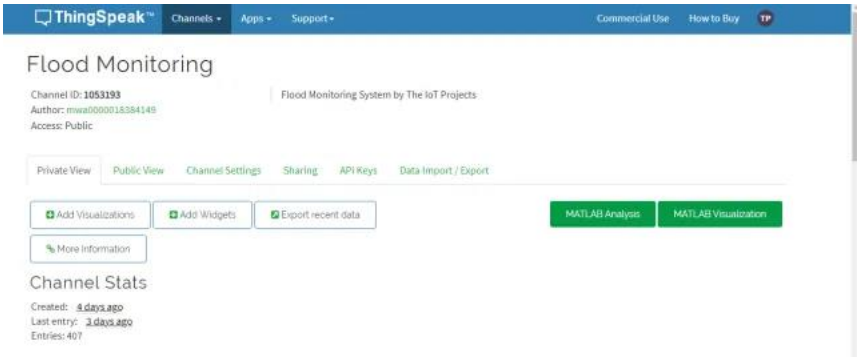
Step 2: Sign in to ThingSpeak

Sign in to ThingSpeak using your credentials and create "New Channel". Now fill the project details like name, field names, etc. Here we need to create three field area names such as Flood Live Monitoring, and Flood Status. Then click "Save Channel".

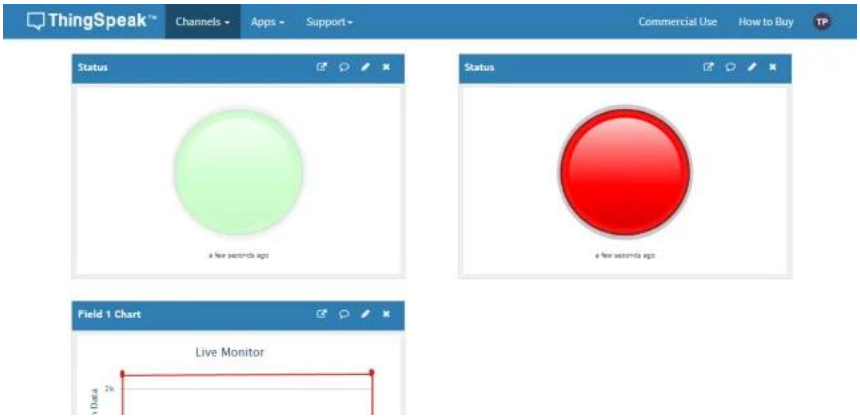


Step 3: Record the credentials
Select the created channel and record the following credentials.

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.



Step 4: Add widgets to your GUI
Click “Add Widgets” and add two appropriate Indicator widgets. In my case, I have taken an indicator of flooding. Choose the appropriate field names for each widget.



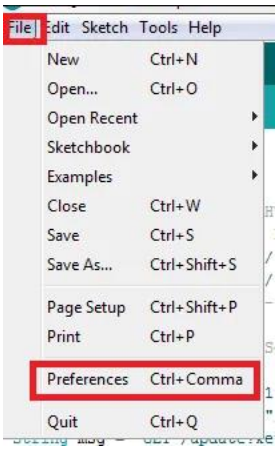
Red Indicator Indicates Flooded Condition

Setting up Arduino IDE for NodeMCU Board

After the successful completion of the hardware setup. Now its time to program ESP8266 NodeMCU.

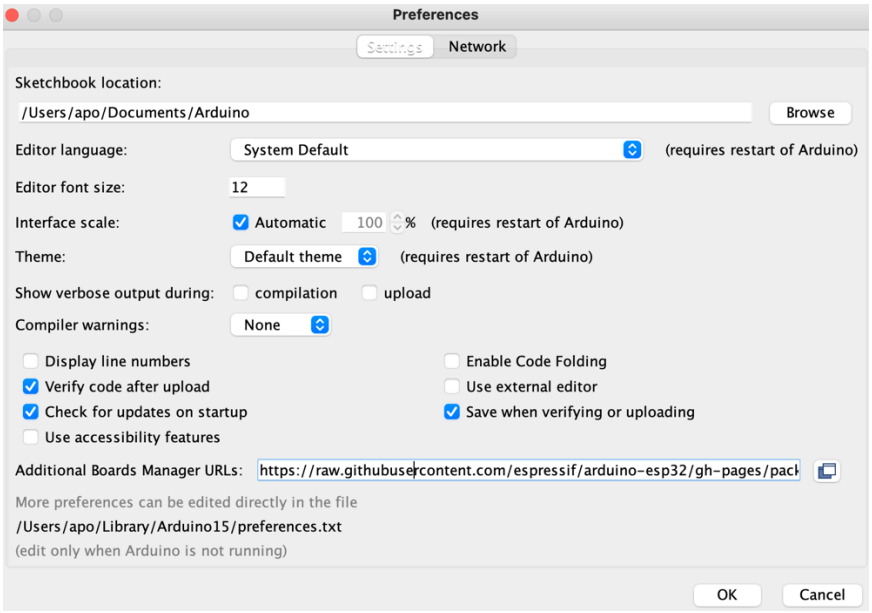
To upload the code to NodeMCU using the Arduino IDE, follow the steps below:

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



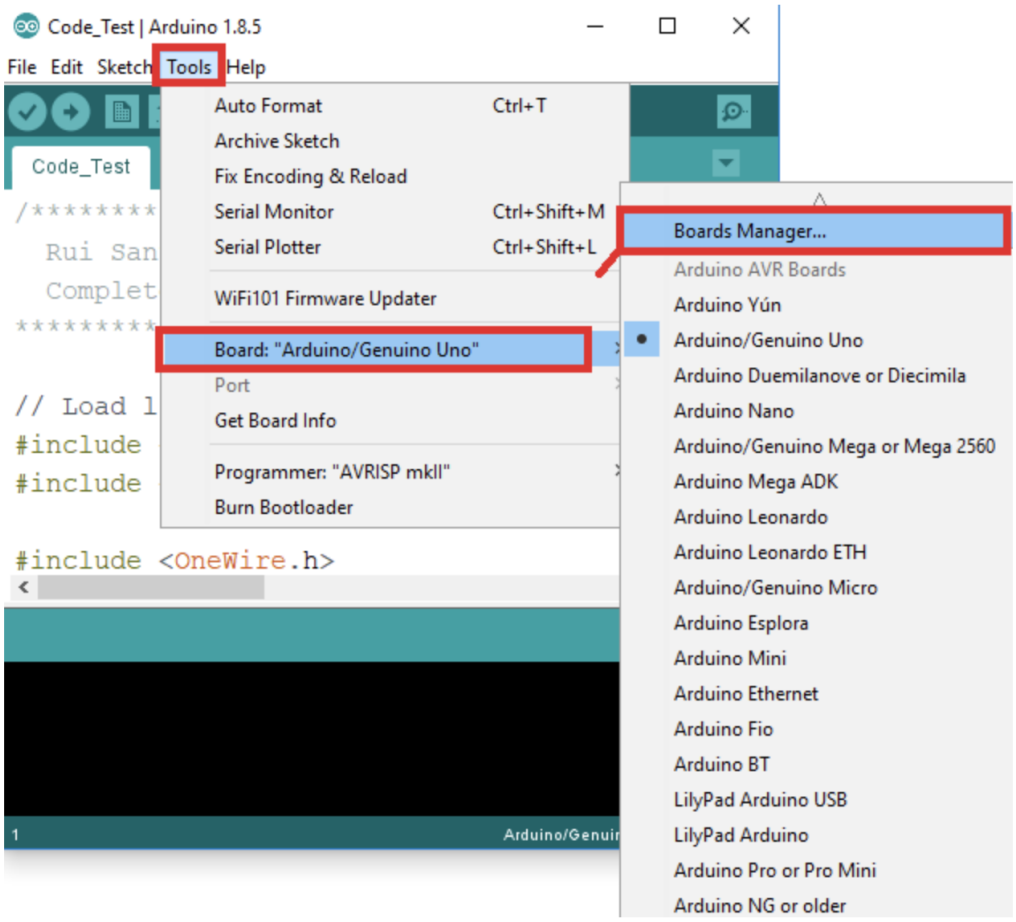
Paste the URL: **https://dl.espressif.com/dl/package_esp32_index.json**

2. in the 'Additional Board Manager URL' field and click 'Ok'.

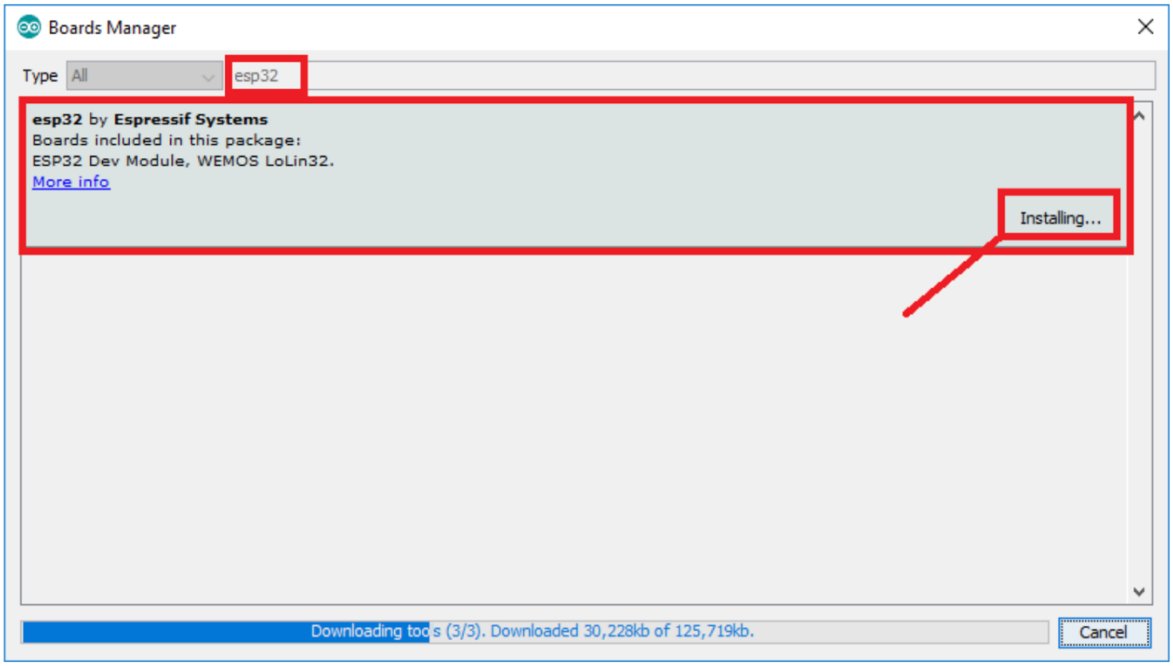


Now. Now go to **Tools> Board> Board Manager.** In the Boards Manager window, type **ESP32 in the search box**, select the new version of the board and click Install.

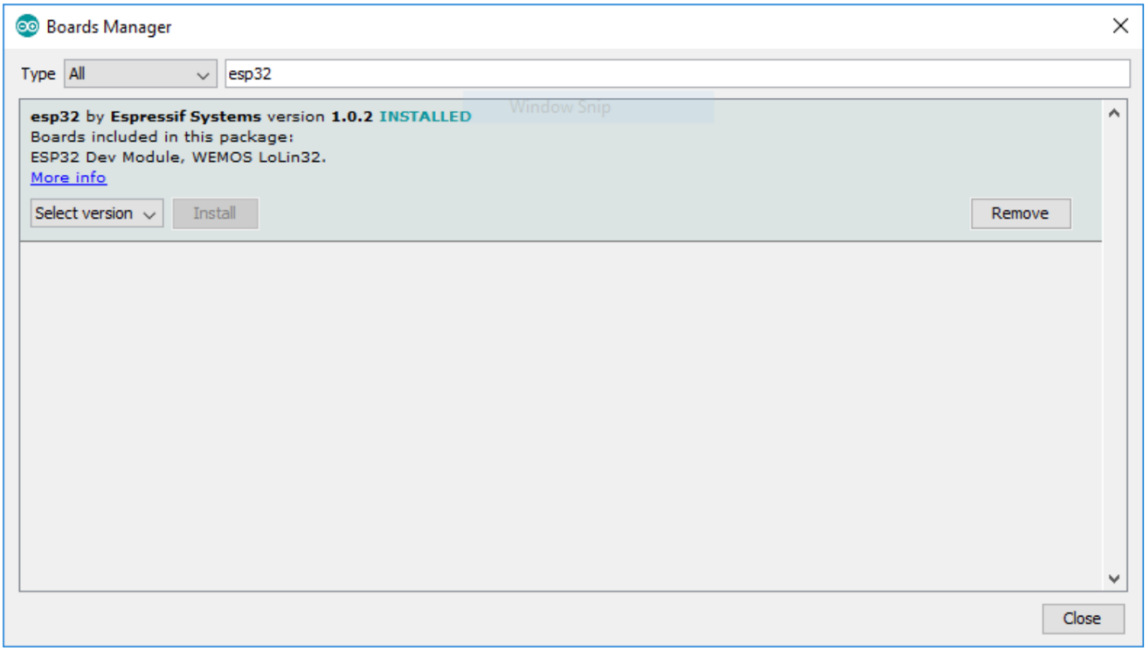
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager...**



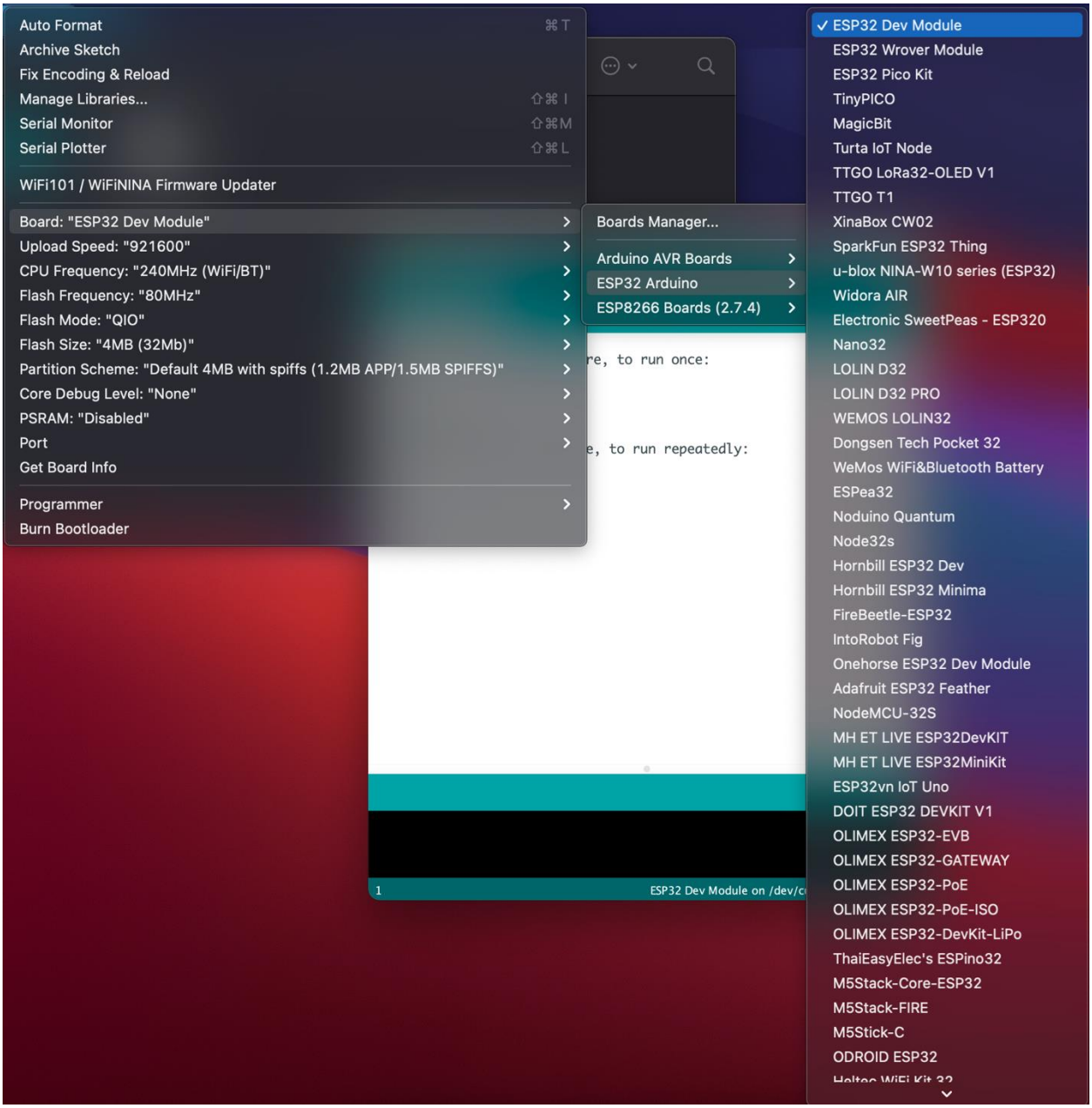
Search for **ESP32** and press install button for the “**ESP32 by Espressif Systems**“:



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



After successful installation, go to Tools -> Board -> and select (ESP32 Dev Module). Now you can program NodeMCU with Ardino IDE.

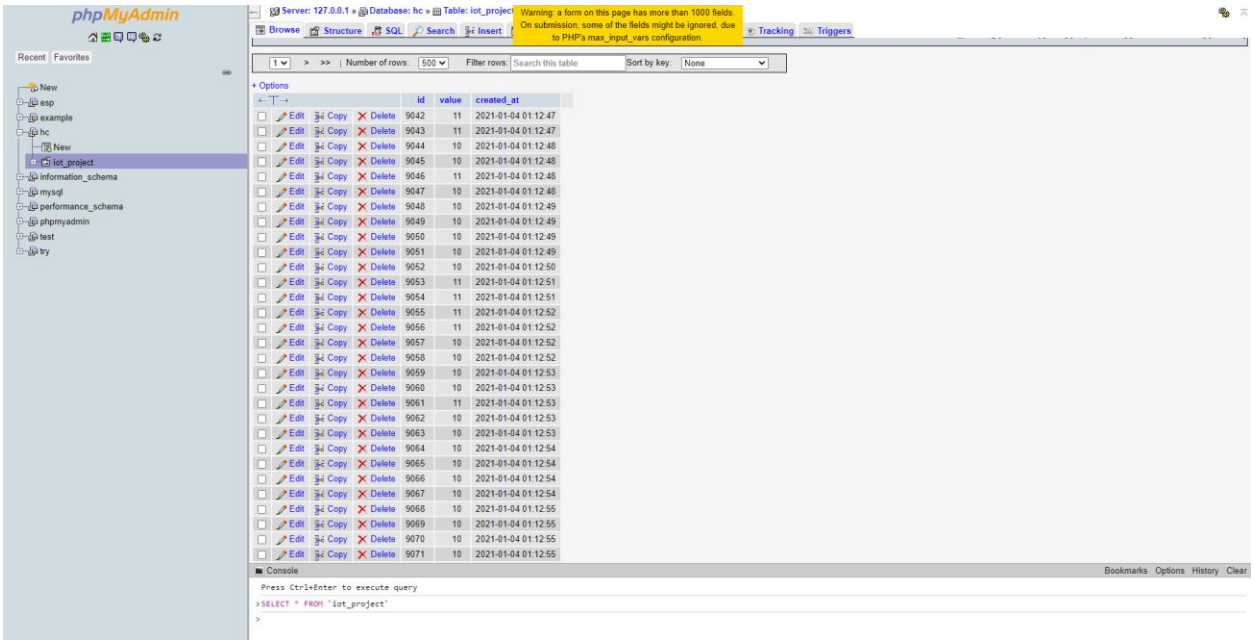


After the above setup for programming NodeMCU. You can upload the complete code to ESP32 NodeMCU. The step-by-step explanation of the full code is provided below.

PHP code

```php
<html>
<body>

<?php

$dbname = 'HC';
$dbuser = 'apo';
$dbpass = 'test1234';
$dbhost = 'localhost';

$connect = @mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!$connect){
    echo "Error: " . mysqli_connect_error();
    exit();
}

echo "Connection Success!<br><br>";

$value = $_GET["value"];


$query = "INSERT INTO iot_project (value) VALUES ('$value')";
$result = mysqli_query($connect,$query);

echo "Insertion Success!<br>";

?>
</body>
</html>

```

PHPMYADMIN



Program Code explanation

Begin the code by including all the necessary library files in the code for ESP32 boards. The ThingSpeak.h library is used for the ThingSpeak platform. It can be added to the Arduino IDE using the following steps:

- ***In the Arduino IDE, choose Sketch/Include Library/Manage Libraries.***
- ***Click the ThingSpeak Library from the list, and click the Install button.***

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

Next, define the pins which are used for the Ultrasonic sensor, and LEDs.

```
// defines pins numbers
#define redled 2
#define grnled 4
const int trigPin = 23;
const int echoPin = 22;
```

```
const int buzzer = 15; // Buzzer
```

Now, Enter the network credentials- i.e. SSID and password of your WiFi Network to connect the NodeMCU with the internet. We add host for database. 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, you have edited these credentials in place of these variables.

```
const char* ssid     = "your_SSID";
const char* password = "your_Password";
const char* host = "192.168.1.52";
// thingspeak

unsigned long ch_no = 1276640;//Replace with Thingspeak Channel number
const char * write_api = "O2GIAC35AN5EFB3M";//Replace with Thingspeak write API
char auth[] = "mwa0000020515336";
```

The variables are defined for timing purposes.

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 10000;
WiFiClient  client;
// defines pins numbers
const int trigPin = 23;
const int echoPin = 22;
// defines variables
long duration;
int distance;
```

setup function for declaration variables and pins, and start connecting

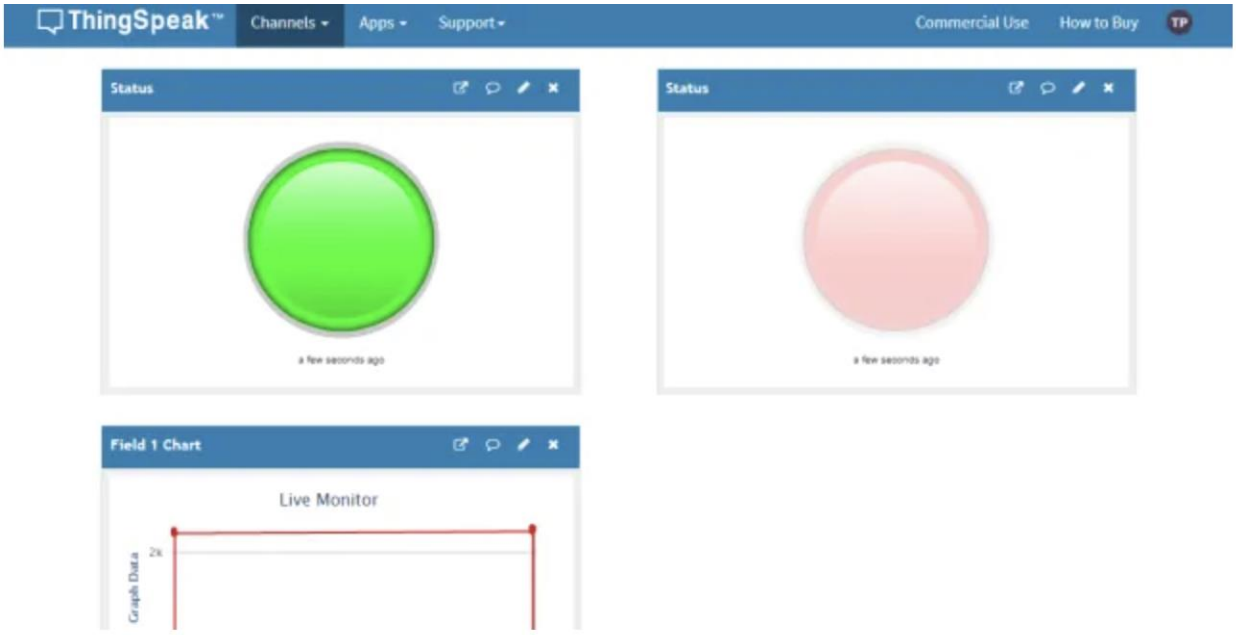
```
pinMode(grnled, OUTPUT);
   pinMode(redled, OUTPUT);
 pinMode(buzzer, OUTPUT);
 Serial.begin(115200);

   // We start by connecting to a WiFi network

   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);

   WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }


   Serial.println("");
   Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());

 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input


 ThingSpeak.begin(client);
```

```
 startMillis = millis();  //initial start time
```

loop function to connect countainsly to the wifi and server and store date. Connect the nodemcu to thingspeak

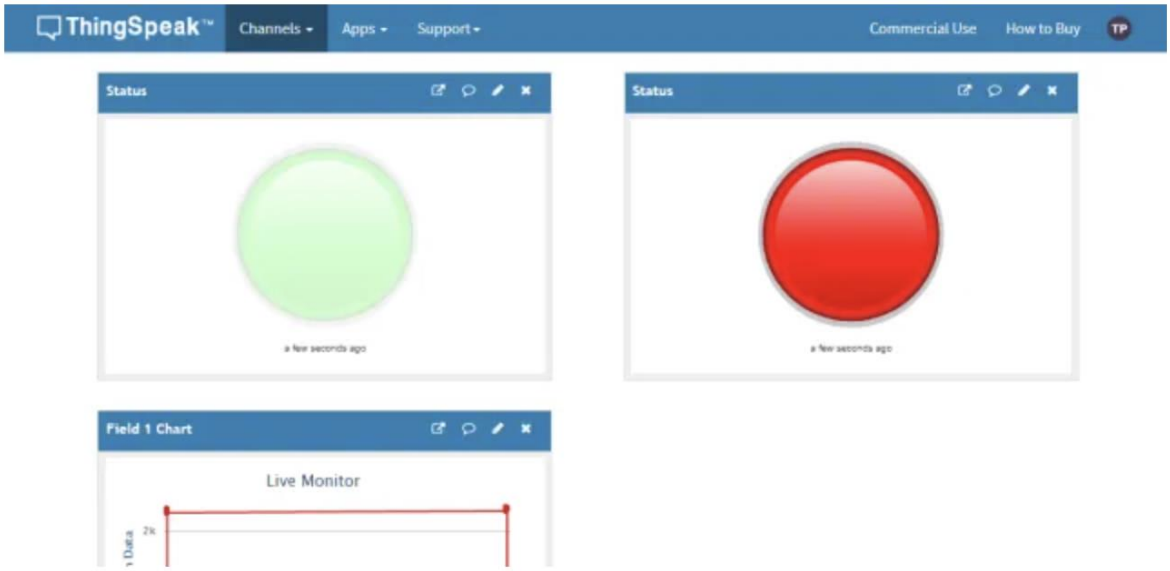
```
// Clears the trigPin
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 // Sets the trigPin on HIGH state for 10 micro seconds
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 // Reads the echoPin, returns the sound wave travel time in microseconds
 duration = pulseIn(echoPin, HIGH);
 // Calculating the distance
 distance= duration*0.034/2;
 // Prints the distance on the Serial Monitor
 Serial.print("Distance: ");
 Serial.println(distance);


 if(isnan(distance)){
  Serial.println("Failed to read HC");
 }else{
  Serial.print("value: ");
  Serial.print(distance);
  Serial.println("");
  delay(200);
 }

  if (distance <= 9)
 {
  digitalWrite(redled, HIGH);
  digitalWrite(buzzer, HIGH);
  digitalWrite(grnled, LOW);
  delay(200);
 }
 else
 {
  digitalWrite(grnled, HIGH);
  digitalWrite(buzzer, LOW);
  digitalWrite(redled, LOW);
 }
 currentMillis = millis();
 if (currentMillis - startMillis >= period)
 {
```

```
  ThingSpeak.setField(1, distance);
  ThingSpeak.writeFields(ch_no, write_api);
  startMillis = currentMillis;
 }

  Serial.print("connecting to ");
  Serial.println(host);


  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
     Serial.println(client.connect(host, httpPort));
     Serial.println("connection failed");
     return;
  }
 Serial.println("well done");


  // This will send the request to the server
client.print(String("GET http://192.168.1.52/iot/connect.php?") +
                    ("&value=") + distance +
                    " HTTP/1.1\r\n" +
            "Host: " + host + "\r\n" +
            "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
     if (millis() - timeout > 1000) {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return;
     }
  }



  Serial.println();
  Serial.println("closing connection");
```



Green Indicator in Normal Condition

Basically, ThingSpeak will show a big green indicator while there is no flood. and in case of Flood a Red Indicator as shown below:



Wrapping up

This is just a basic model for an IoT Based Flood Monitoring System Using NodeMCU & ThingSpeak. We can improve this project for better performance. Hence, there is a lot of scope for improvising it. I hope you enjoyed the project and learned something new.