**WIFI WEATHER STATION USING NodeMCU AND THINGSPEAK**

**INTRODUCTION:**

Weather forecast these days is unpredictable to be exact because of the climate changes drastically over weather. In cause of that, Weather Reporting System is mostly used to monitor the continuously changing climatic and weather conditions over controlled areas likes house, industry, agriculture and etc. in real time monitoring. So, in this project we have made an IoT based Weather Station to measure environmental conditions. In this project, we will measure Humidity, Temperature and Pressure parameters and display them on the web server, which makes it a IoT based Weather Stationwhere the weather conditions can be monitored from anywhere using the Internet.

**IoT BASED WEATHER STATION:**

In this project, we measure the temperature, humidity, pressure of the surrounding using which weather condition of that place is determined. These parameters are measured using various sensors and the output of those sensors are sent to a microcontroller connected to a webserver. The parameters of the surrounding can be visible in the webserver. DHT11 sensor is used to measure the Humidity and temperature, BMP280 sensor measures the absolute Pressure of the surrounding air. The measured data are sent to NodeMCU which is the microcontroller used here which processes those data and sends the information to Webserver. This project will focus on development of the ThingSpeak an IoT platform that to show the data of the sensor. In this project, all the weather data will be recorded and saved on the ThingSpeak IoT cloud which can be monitored from anywhere in the world.

**WORKING:**

This project uses sensors like DTH11 sensor, DHT11 module features a humidity and temperature complex with a calibrated digital signal output means DHT11 sensor module is a combined module for sensing humidity and temperature which gives a calibrated digital output signal. DHT11 gives us very precise value of humidity and temperature and ensures high reliability and long-term stability. The BPM180 sensor, which is a barometric pressure sensor and it works with an I2C interface. This sensor measures the absolute pressure of the air around it. The pressure value depends on both the weather and altitude. It depends on how you interpret the data, and can easily monitor changes in the weather, measure the altitude, or any other tasks that require an accurate pressure reading. The informations collected from these sensors are sent to an ESP8266 NodeMCU module where the data gets processed and sent to the ThinkSpeak IoT cloud, using which the weather condition can be monitored from anywhere in the world and can be used for future purposes.

**CODING USED:**

#include "ThingSpeak.h"

#include <ESP8266WiFi.h>

#include <SFE\_BMP180.h>

#include <Wire.h>

#define ALTITUDE 1655.0

#include <Adafruit\_Sensor.h>

//#include "DHT.h"

SFE\_BMP180 pressure;

//Adafruit\_BMP280  bmp280;

//SFE\_BMP180 pressure;

//#define DHTPIN 3          // GPIO Pin where the dht11 is connected

//DHT dht(DHTPIN, DHT11);

String stat = "";

const char\* ssid     = "123456789";//Replace with Network SSID

const char\* password = "0987654321";//Replace with Network Password

const char\* server = "api.thingspeak.com";

String apiWritekey = "HXQP85BCCXT1SLHC";//Replace with Thingspeak write API

WiFiClient  client;

void setup(void)

{

  Serial.begin(115200);

  WiFi.disconnect();

  delay(1000);

  WiFi.begin(ssid, password);

  Serial.println();

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Wire.begin(4, 0);

 // dht.begin();

  if (pressure.begin())

    Serial.println("BMP180 init success");

  else

  {

    Serial.println("BMP180 init fail\n\n");

    //while(1); // Pause forever.

  }

}

void loop()

{

  char status;

  double T,P,p0,a;

  // Loop here getting pressure readings every 10 seconds.

  // If you want sea-level-compensated pressure, as used in weather reports,

  // you will need to know the altitude at which your measurements are taken.

  // We're using a constant called ALTITUDE in this sketch:

  Serial.println();

  Serial.print("provided altitude: ");

  Serial.print(ALTITUDE,0);

  Serial.print(" meters, ");

  Serial.print(ALTITUDE\*3.28084,0);

  Serial.println(" feet");

  // If you want to measure altitude, and not pressure, you will instead need

  // to provide a known baseline pressure. This is shown at the end of the sketch.

  // You must first get a temperature measurement to perform a pressure reading.

  // Start a temperature measurement:

  // If request is successful, the number of ms to wait is returned.

  // If request is unsuccessful, 0 is returned.

  status = pressure.startTemperature();

  if (status != 0)

  {

    // Wait for the measurement to complete:

    delay(status);

    // Retrieve the completed temperature measurement:

    // Note that the measurement is stored in the variable T.

    // Function returns 1 if successful, 0 if failure.

    status = pressure.getTemperature(T);

    if (status != 0)

    {

      // Print out the measurement:

      Serial.print("temperature: ");

      Serial.print(T,2);

      Serial.print(" deg C, ");

      Serial.print((9.0/5.0)\*T+32.0,2);

      Serial.println(" deg F");

      // Start a pressure measurement:

      // The parameter is the oversampling setting, from 0 to 3 (highest res, longest wait).

      // If request is successful, the number of ms to wait is returned.

      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);

      if (status != 0)

      {

        // Wait for the measurement to complete:

        delay(status);

        // Retrieve the completed pressure measurement:

        // Note that the measurement is stored in the variable P.

        // Note also that the function requires the previous temperature measurement (T).

        // (If temperature is stable, you can do one temperature measurement for a number of pressure measurements.)

        // Function returns 1 if successful, 0 if failure.

        status = pressure.getPressure(P,T);

        if (status != 0)

        {

          // Print out the measurement:

          Serial.print("absolute pressure: ");

          Serial.print(P,2);

          Serial.print(" mb, ");

          Serial.print(P\*0.0295333727,2);

          Serial.println(" inHg");

          // The pressure sensor returns abolute pressure, which varies with altitude.

          // To remove the effects of altitude, use the sealevel function and your current altitude.

          // This number is commonly used in weather reports.

          // Parameters: P = absolute pressure in mb, ALTITUDE = current altitude in m.

          // Result: p0 = sea-level compensated pressure in mb

          p0 = pressure.sealevel(P,ALTITUDE); // we're at 1655 meters (Boulder, CO)

          Serial.print("relative (sea-level) pressure: ");

          Serial.print(p0,2);

          Serial.print(" mb, ");

          Serial.print(p0\*0.0295333727,2);

          Serial.println(" inHg");

          // On the other hand, if you want to determine your altitude from the pressure reading,

          // use the altitude function along with a baseline pressure (sea-level or other).

          // Parameters: P = absolute pressure in mb, p0 = baseline pressure in mb.

          // Result: a = altitude in m.

          a = pressure.altitude(P,p0);

          Serial.print("computed altitude: ");

          Serial.print(a,0);

          Serial.print(" meters, ");

          Serial.print(a\*3.28084,0);

          Serial.println(" feet");

        }

        else Serial.println("error retrieving pressure measurement\n");

      }

      else Serial.println("error starting pressure measurement\n");

    }

    else Serial.println("error retrieving temperature measurement\n");

  }

  else Serial.println("error starting temperature measurement\n");

    int rain = analogRead(A0);

  if (client.connect(server,80)) {

    String tsData = apiWritekey;

    tsData +="&field1=";

    tsData += String(T);

    tsData +="&field2=";

    tsData += String(a);

    tsData +="&field3=";

    tsData += String(P);

    tsData +="&field4=";

    tsData += String(rain);

    tsData += "\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");

    client.print("Host: api.thingspeak.com\n");

    client.print("Connection: close\n");

    client.print("X-THINGSPEAKAPIKEY: "+apiWritekey+"\n");

    client.print("Content-Type: application/x-www-form-urlencoded\n");

    client.print("Content-Length: ");

    client.print(tsData.length());

    client.print("\n\n");

    client.print(tsData);

  }

  client.stop();

    Serial.print("Rain: ");

    Serial.print(rain);

    Serial.print("Pressure:");

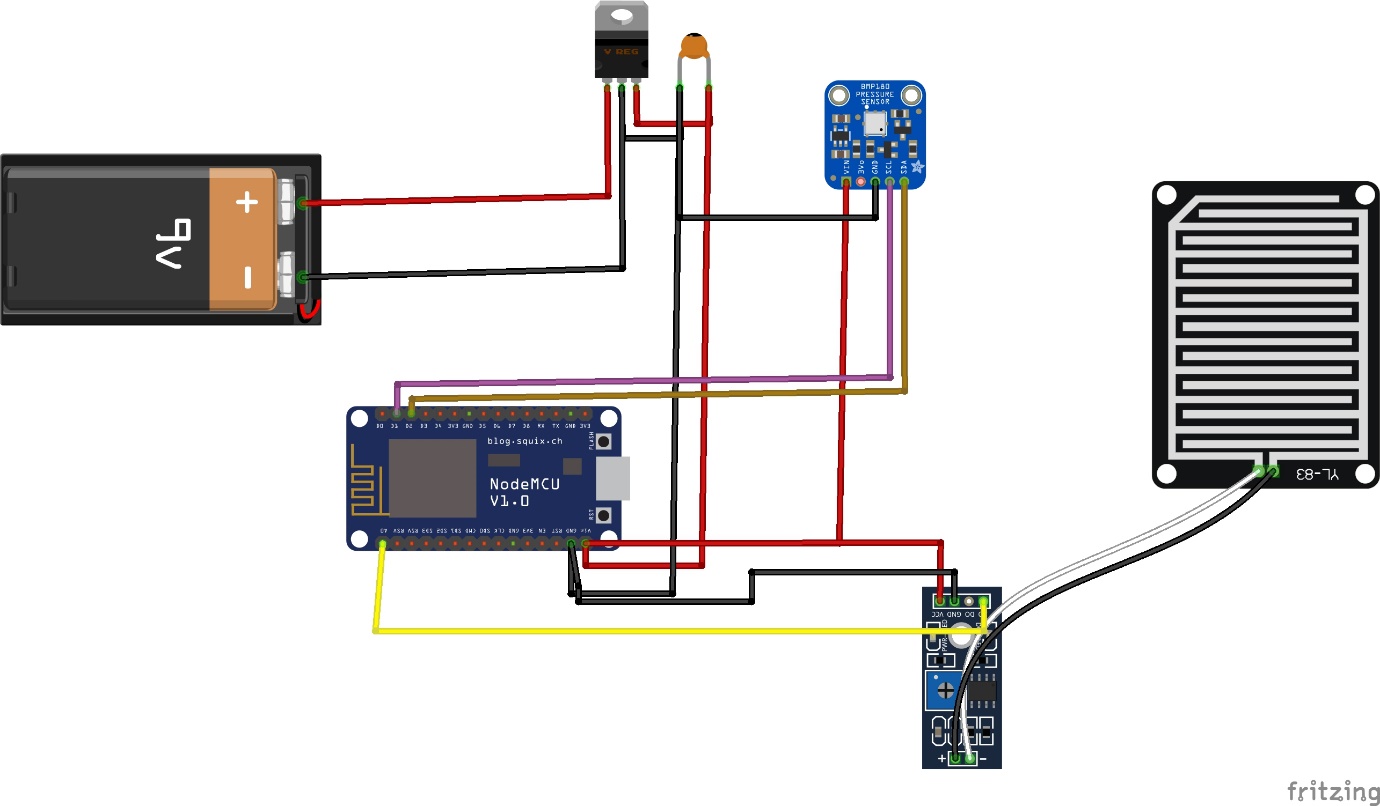
    Serial.println(P);

    Serial.println("uploaded to Thingspeak server....");

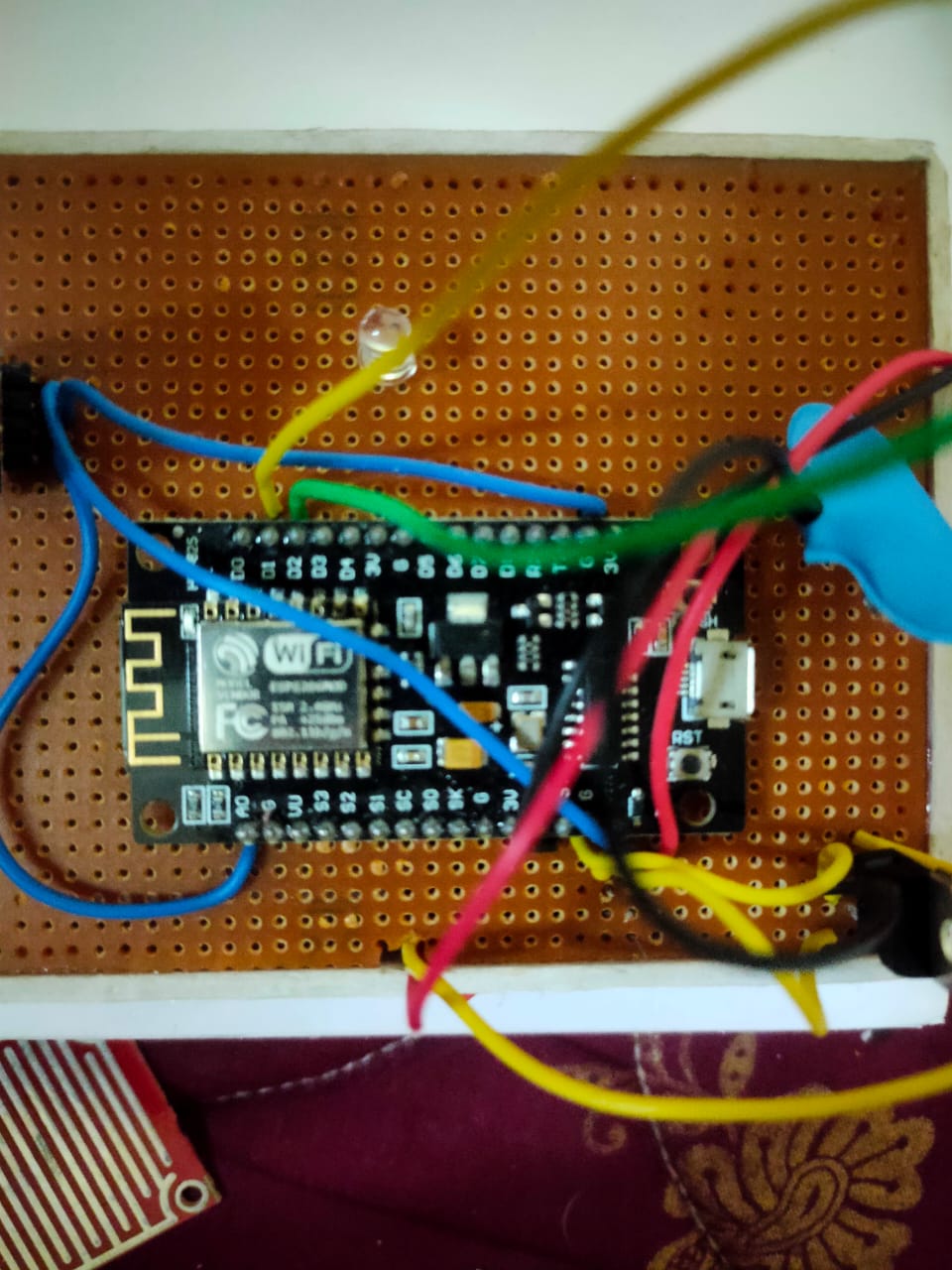
  delay(10000);

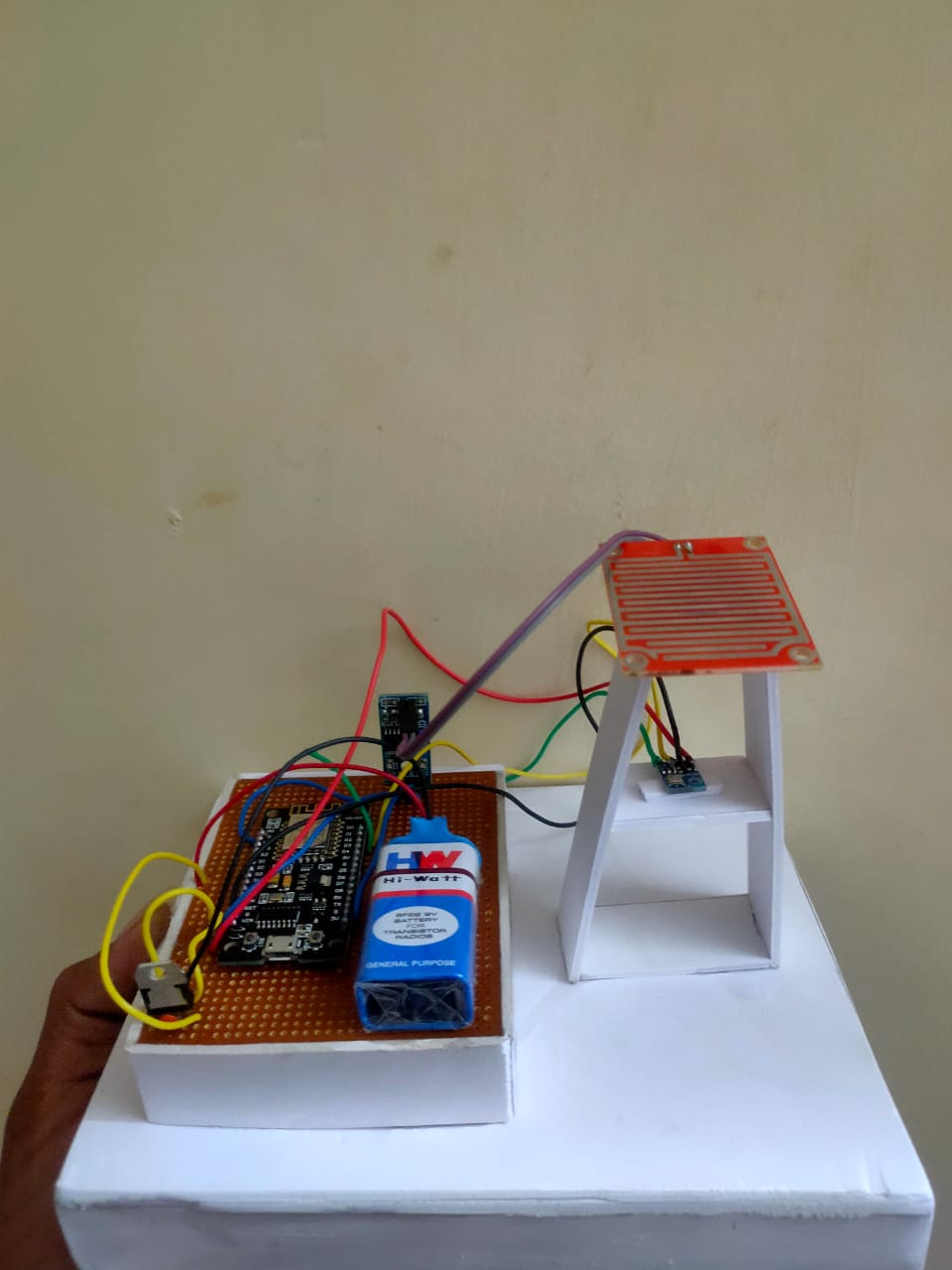
}

**CIRCUIT DIAGRAM:**

****

**PROJECT IMAGES:**

****

****

**RESULT:**

In applications like flood lines, reservoirs, hydroelectricity, water treatment, and urban drainage a weather station to measure rainfall, soil moisture, and other conditions is crucial. Using this project, monitoring the temperature of the surrounding is made ease and you can rest easy knowing that the information you are receiving is current, accurate and relevant.