WIRELESS WEATHER STATION POWERED BY ARDUINO

BY:

➢ SANOFAR JAHAN MOHAMED NIAZ ALI

➢ VARUN SUBRAMANIAN

➢ VAIDESHWARY K

➢ REVANTH KUMAR K

ABSTRACT

This IoT based wireless weather station project is used to get a relay of data based on weather conditions of a particular place. It will monitor temperature, humidity and moisture levels of the particular place the station is situated in.

Global warming has led to unpredictable climates wherein researchers around the world are using weather stations to observe, record and analyse weather patterns to study climate change and provide weather forecasts. These Weather stations normally comprise of sensors used to measure environmental parameters and a monitoring or logging system to analyse these parameters. Also, since our weather station is IoT enabled, we are able to send these data parameters to an IoT cloud (ThingSpeak Channel) where we can store, analyse, and access the data remotely

INTRODUCTION

Climate change and environmental weather monitoring have received a great deal of attention recently. People want to stay updated about the latest weather conditions of any place, for example, a student in a college campus wants to know whether it will rain later in the evening or not. In this paper we present a weather station that can be situated in just about any place. This weather station is based on IoT (internet of things). This prototype is equipped with environmental sensors used to measure data at any location and report them in real time on our IoT cloud. To accomplish this we used an Arduino Uno board and different environmental sensors like the DHT11 and BMP180 sensors. The sensors constantly sense the weather parameters and keeps on transmitting it to the cloud over a Wi-Fi connection. The weather parameters are then uploaded on the cloud and this provides a live relay of weather information. The system has been developed particularly in the view of building smart cities by giving the weather update of any particular place like a particular office or room.

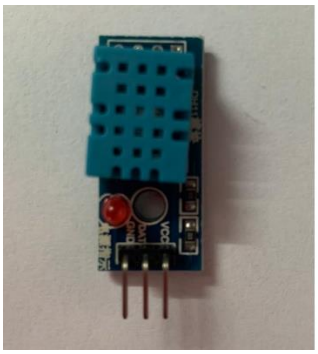
METHODOLOGY

This project focuses on the transfer of data to the ThingSpeak channel and to extract the data of the sensors. The project is twofold, where the first part deals with the hardware and the second part deals with the software development. The hardware development involves the circuit construction of the prototype. Meanwhile, the software part involves the circuit simulation, data acquisition and data transfer. We have used the Arduino Uno board along with the DHT11 sensor, BMP180 sensor and an ESP8266 Wi-Fi module. The DHT11 sensor senses the temperature and humidity, while the BMP180 sensor calculates the pressure and ESP8266 is used for internet connectivity. Sending these data to ThingSpeak enables live monitoring from anywhere in the world and we can also view the logged data which will be stored on the website and even graph it over time to analyse it.

HARDWARE

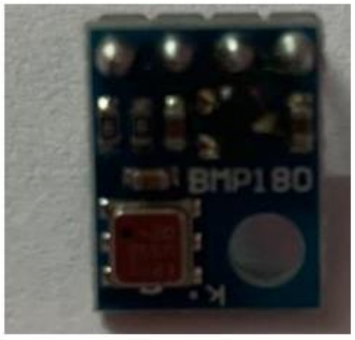
1. Arduino Uno board


2. DHT11 Sensor




3. ESP8266 Wi-Fi Shield


4. BMP180 Sensor



5.Power cable

6.. Male to female jumper wires

7.Male to male jumper wires

Arduino Uno Board:

Arduino Uno is a microcontroller board developed by Arduino. It is an open-source electronics platform mainly based on AVR microcontroller Atmega328. The current version of Arduino Uno comes with USB interface, 6 analog input pins, 14 I/O digital ports that are used to connect with external electronic circuits. Out of 14 I/O ports, 6 pins can be used for PWM output. It allows the designers to control and sense the external electronic devices in the real world. This board comes with all the features required to run the controller and can be directly connected to the computer through USB cable that is used to transfer the code to the controller using IDE software, mainly developed to program Arduino. Programming languages like C and C++ are used in IDE. Apart from USB, battery or AC to DC adopter can also be used to power the board. Arduino Uno are the most official versions that come with Atmega328 8-bit AVR Atmel microcontroller where RAM memory is 32KB

DHT11 Sensor:

This DHT11 Temperature and Humidity Sensor features digital signal output .It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability. It has excellent quality, fast response, anti-interference ability and high performance. Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

BMP180 Sensor :

BMP180 is one of sensor of BMP XXX series. They are all designed to measure Barometric Pressure or Atmospheric pressure. BMP180 is a high precision sensor designed for consumer applications. Barometric Pressure is nothing but weight of air applied on everything. The air has weight and wherever there is air its pressure is felt. BMP180 sensor senses that pressure and provides that information in digital output. Also the temperature affects the pressure and so we need temperature compensated pressure reading. To compensate, the BM180 also has good temperature sensor.

ESP8266 Wi-Fi Shield:

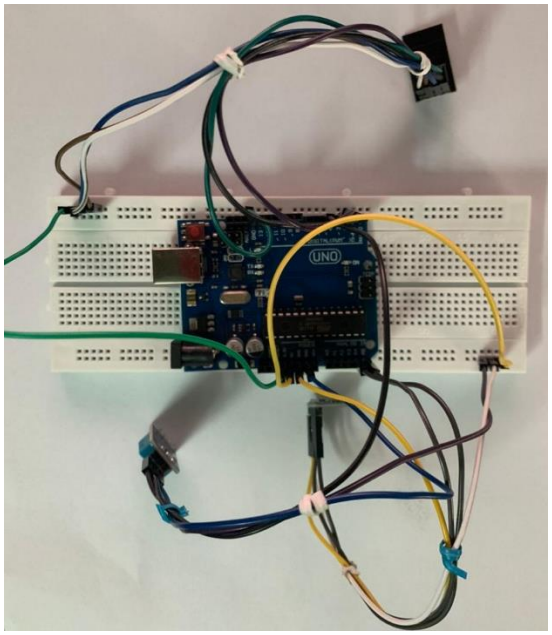
The Arduino Uno WiFi is an Arduino Uno with an integrated WiFi module. The ESP8266 WiFi Module is a self contained SoC with integrated TCP/IP protocol stack that can give access to your WiFi network (or the device can act as an access point). One useful feature of Uno WiFi is support for OTA (over-the-air) programming, either for transfer of Arduino sketches or WiFi firmware.

OTHERS:

The Arduino Uno Board provides a limited number of 3.3V and 5V connections, hence we use a breadboard.

The male to female jumper wires connect all the sensors to the Arduino board whereas the male to male jumper wires connect the Arduino board to the bread board.

CIRCUIT CONNECTIONS:



The DHT11 sensor is powered by the 5V pin of the Arduino and its data pin is connected to pin 5 for one-wire communication. The BMP180 sensor is powered by the 3.3V pin of Arduino and its data pins SCL (Serial Clock) and SDA (Serial Data) are connected to the A4 and A5 pin of Arduino for I2C communication.

The ESP8266 module is also powered by the 3.3V pin of the Arduino and its Tx and Rx pins are connected to Digital pins 2 and 3 of Arduino for serial communication.

S.NO.	Pin Name	Arduino Pin
1	ESP8266 VCC	3.3V
2	ESP8266 RST	3.3V
3	ESP8266 CH-PD	3.3V
4	ESP8266 RX	TX
5	ESP8266 TX	RX
6	ESP8266 GND	GND
7	BMP180 VCC	5V
8	BMP180 GND	GND
9	BMP180 SDA	A4
10	BMP180 SCL	A5
11	DHT-11 VCC	5V
12	DHT-11 Data	5
13	DHT-11 GND	GND

SOFTWARE

ThingSpeak:

ThingSpeak is an open data platform that allows you to aggregate, visualize, and analyse live data in the cloud. You can control your devices using ThingSpeak, you can send data to ThingSpeak from your devices, and even you can create instant visualizations of live data, and send alerts using web services like Twitter and Twilio. ThingSpeak has integrated support from the numerical computing software MATLAB. MATLAB allows ThingSpeak users to write and execute MATLAB code to perform pre-processing, visualizations, and analyses. ThingSpeak takes a minimum of 15 seconds to update your readings

SETTING UP YOUR THINGSPEAK CHANNEL

- ### Step 1: ThingSpeak Account Setup

To create a channel on ThingSpeak, first, you need to Sign up on ThingSpeak. In case if you already have an account on ThingSpeak, sign in using your id and password. For creating your account go to www.thinspeak.com.

- ### Step 2: Create a Channel for Your Data

Once you Sign in after your account verification, Create a new channel by clicking the "*New Channel*" button.

After clicking on "*New Channel,"* enter the *Name* and *Description* of the data you want to upload on this channel.

Enter the name of your data '*Humidity*' in *Field1*, '*Temp'* in *Field2,* and '*Pressure*' in *Field3.* If you want to use more Fields, you can check the box next to the Field option and enter the name and description of your data.

After this, click on the save channel button to save your details.

- ### Step 3: API Key

To send data to ThingSpeak, we need a unique API key, which we will use later in our code to upload our sensor data to Thingspeak Website.

Click on the *"API Keys"* button to get your unique API key for uploading your sensor data.

Now copy your "Write API Key." We will use this API key in our code.

CODE:

```
String myAPIkey = "YDLJ7OMQ4JJHT8GI";
#include <SoftwareSerial.h>
#include <DHT.h>
#include <SFE_BMP180.h>
#include <Wire.h>
#define DHTTYPE DHT11
#define DHTPIN  5
#define ALTITUDE 1655.0

SFE_BMP180 pressure;
SoftwareSerial ESP8266(2, 3); // Rx,  Tx
DHT dht(DHTPIN, DHTTYPE, 11);



float humidity, temp_f;
long writingTimer = 17;
long startTime = 0;
long waitTime = 0;

boolean relay1_st = false;
boolean relay2_st = false;
unsigned char check_connection = 0;
unsigned char times_check = 0;
boolean error;

char status;
double T, P, p0, a;

void setup()
{
  Serial.begin(115200);
  ESP8266.begin(115200);
```

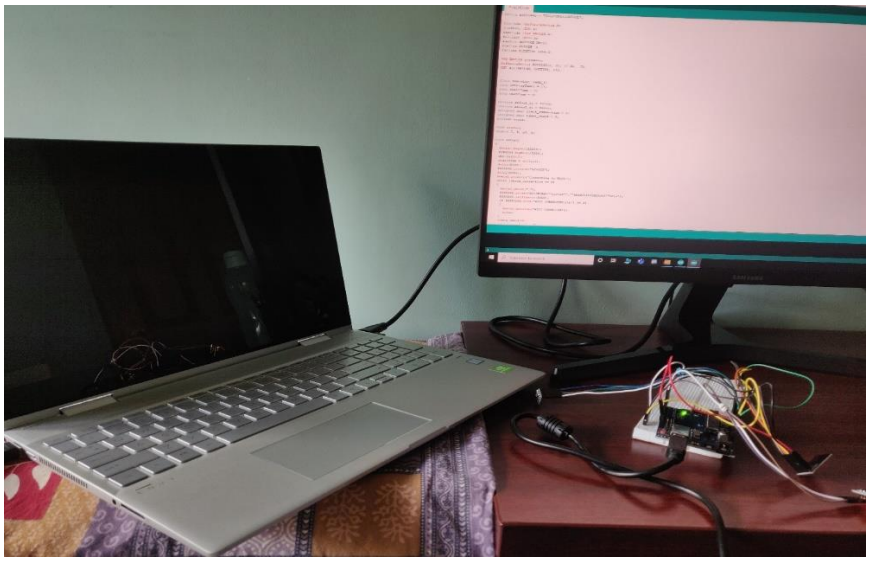
```
  dht.begin();
  startTime = millis();
  delay(5000);
  ESP8266.println("AT+RST");
  delay(2000);
  Serial.println("Connecting to Wifi");
  while (check_connection == 0)
  {
    Serial.print(".");
    ESP8266.print("AT+CWJAP=""tarvar"",""abc9003013622def""\r\n");
    ESP8266.setTimeout(5000);
    if (ESP8266.find("WIFI CONNECTED\r\n") == 1)
    {
      Serial.println("WIFI CONNECTED");
      break;
    }
    times_check++;
    if (times_check > 3)
    {
      times_check = 0;
      Serial.println("Trying to Reconnect..");
    }
  }
  if (pressure.begin())
    Serial.println("BMP180 init success");
}

void loop()
{
  readSensors();
  writeThingSpeak();
  delay(1000);
}
```

```
void readSensors(void)
{
  status = pressure.startTemperature();
  if (status != 0)
  {
    delay(status);
    status = pressure.getTemperature(T);
    if (status != 0)
    {
      status = pressure.startPressure(3);
      if (status != 0)
      {
        delay(status);
        status = pressure.getPressure(P, T);
        if (status != 0)
        {
          Serial.print(P, 2);
          Serial.println(" mb, ");
        }
        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");
  temp_f = dht.readTemperature();
  humidity = dht.readHumidity();
}


void writeThingSpeak(void)
{
```

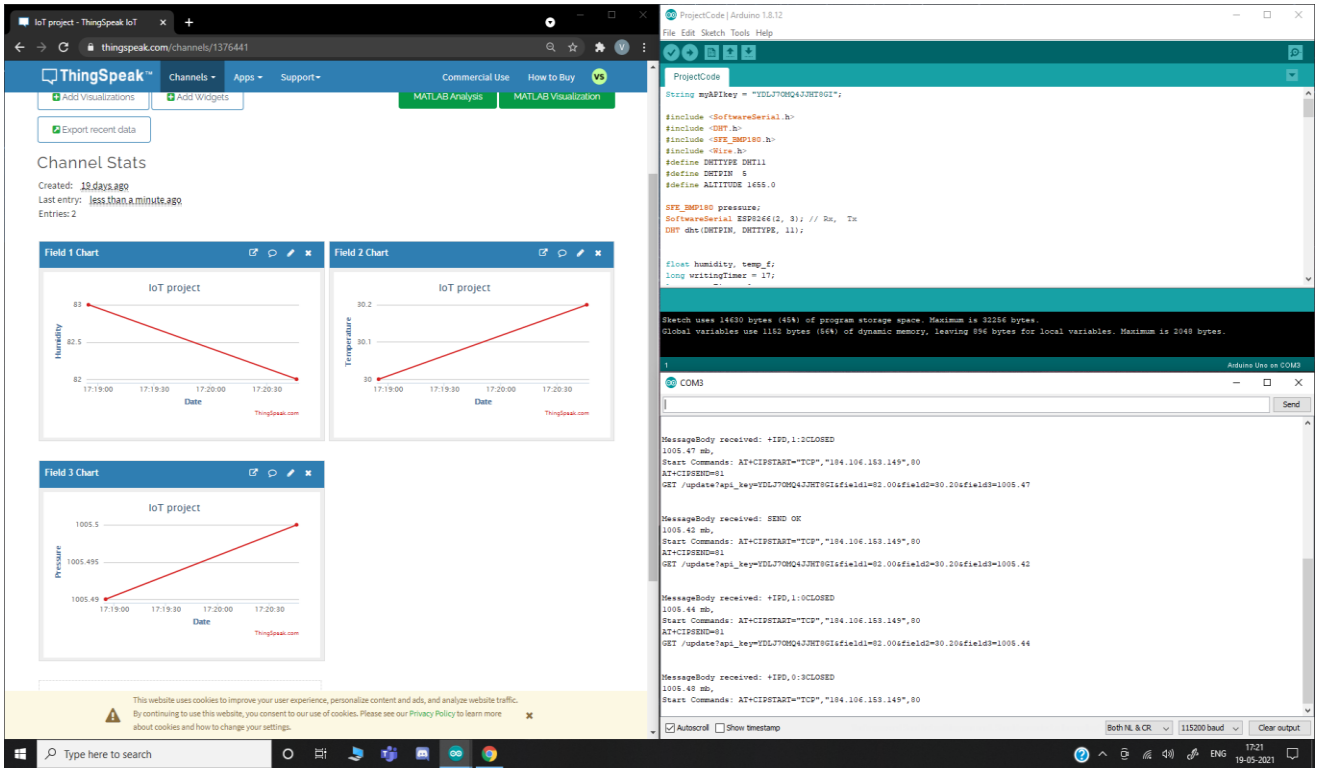
```
  startThingSpeakCmd();
  String getStr = "GET /update?api_key=";
  getStr += myAPIkey;
  getStr += "&field1=";
  getStr += String(humidity);
  getStr += "&field2=";
  getStr += String(temp_f);
  getStr += "&field3=";
  getStr += String(P);
  getStr += "\r\n\r\n";
  GetThingspeakcmd(getStr);
}

void startThingSpeakCmd(void)
{
  ESP8266.flush();
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com IP address
  cmd += "\",80";
  ESP8266.println(cmd);
  Serial.print("Start Commands: ");
  Serial.println(cmd);

  if (ESP8266.find("Error"))
  {
    Serial.println("AT+CIPSTART error");
    return;
  }
}

String GetThingspeakcmd(String getStr)
{
  String cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
```

```
  ESP8266.println(cmd);
  Serial.println(cmd);

  if (ESP8266.find(">"))
  {
    ESP8266.print(getStr);
    Serial.println(getStr);
    delay(500);
    String messageBody = "";
    while (ESP8266.available())
    {
      String line = ESP8266.readStringUntil('\n');
      if (line.length() == 1)
      {
        messageBody = ESP8266.readStringUntil('\n');
      }
    }
    Serial.print("MessageBody received: ");
    Serial.println(messageBody);
    return messageBody;
  }
  else
  {
    ESP8266.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");
  }
}
```

FINAL PROJECT



RESULT:



ADVANTAGES:

- IOT weather monitoring system using Arduino Uno is fully automated.
- It does not require any human attention.
- We can get prior alert of weather conditions
- Cost efficient
- High accuracy
- Smart way to monitor Environment

FUTURE SCOPE:

- One can implement a few more sensors and connect it to the satellite as a global feature of this system.
- Adding more sensor can monitor other environmental parameters such as CO2,Pressure and Oxygen Sensor
- In aircraft, navigation and military there is a great scope of this real-time system.
- It can also be implemented in hospitals or medical institutes for the research & study in "Effect of Weather on Health and Diseases", hence to provide better precaution alerts

CONCLUSION

This project has cleared the objective of building a system that can monitor weather parameters by means of a wireless station linked by IoT to a cloud. The Sensor station and Weather station will be connected through Wi-Fi. To implement this weather station, we've used the DHT11 and BMP180 sensors for collecting the data. Then the collected data and analysis of results is made available to the user through Wi-Fi. Through this project the environment is monitored in a smart way using an efficient and low-cost system. It also sends the sensor parameters to the cloud. This data will be helpful for future analysis and it can be easily shared.