WEATHER MONITORING SYSTEM

Report submitted to IIT Hyderabad
as part of Course Requirements of

CS5453: **INTERNET OF THINGS**

Submitted by

SANDEEP KUMAR NALLALA
(Reg. No.: CS21MTECH11002, M.Tech - Computer Science and Engineering)
RAGURU SAI SANDEEP
(Reg. No.: CS21MTECH11008, M.Tech - Computer Science and Engineering)
PREM KUMAR SARAF
(Reg. No.: CS21MTECH11014, M.Tech - Computer Science and Engineering)

Jan - May 2022

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Table of Contents

Abbreviations

IoT	- Internet of Things
IFTTT	- If This Then That
Node MCU	- Node MicroController Unit
DHT	- Digital Humidity and Temperature
WiFi	- Wireless Fidelity
SoC	- System on a Chip
TCP/IP	- Transmission Control Protocol / Internet Protocol
HTTP	- HyperText Transfer Protocol
GPIO Pin	- General Purpose Input/Output Pin
NTC	- Negative Temperature Coefficient
API	- Application Program Interface
RH	- Relative Humidity
T	- Temperature

List of Figures

List of Tables

ABSTRACT

Weather forecasting is challenging to anticipate these days due to the tremendous changes in climate. As a result, Climate Reporting Systems are commonly used to monitor the constantly changing climatic and weather conditions over regulated regions such as homes, industries, farms, etc. ThingSpeak is the Internet of Things (IoT) platform used, and it should be capable of showing weather parameters and information that will be viewable everywhere in the world. The condition of a specific location as given by a satellite weather report system does not provide the exact situation. However, the issue arises when an exact weather forecast for the current moment is required.

With the weather reporting system, weather parameters like temperature, humidity, heat index, etc will be collected by the sensors and will be handled by an ESP8266 microcontroller as the server, which will transfer all sensor data to the database via ThingSpeak. This data will then be compared to weather forecast data and statistics compiled by forecast stations. All data collected will be recorded in Google Sheets format via the IFTTT application to facilitate data analysis. This system will monitor changes in weather conditions occurring around the environment and then provide users with real-time data.

Keywords: Weather, ThingSpeak, Sensor, ESP8266, IFTTT

CHAPTER 1

INTRODUCTION

Climate has a significant impact on human life. The extraordinary expansion of industries and automobile traffic had a significant impact on the purity of the air and the environment. The satellite weather report system provides the current situation but does not provide the exact condition of a specific location. The building industry has a high potential for energy savings. However, precise meteorological data on the particular site where the building is being erected is required to improve the calibration of energy modeling systems. By developing a controlled local weather reporting system with the ESP8266, the error in the weather forecast system at the same site can be reduced. Precision agriculture and farming are the art and science of using technology to improve crop yields.

Even though water is a finite resource, 50 percent of water is wasted in agriculture due to incorrect irrigation scheduling. In this context, real-time monitoring of water usage in the fields can help to prevent its wastage. The use of technology in agriculture plays a significant part in improving productivity while reducing extra workforce efforts. Some studies have been conducted to benefit farmers and give systems that use technologies that are beneficial in enhancing agriculture yield. Difficulty in monitoring weather parameters happens via offline systems, such as agriculture zones, during particular hazardous circumstances and essential situations where individuals must personally verify the weather condition at the locations, which is time-consuming.

The concept of smart cities and IoT has made a new imprint on the world in the rising generation of wireless technology. One such remark directs attention to the online intelligent weather station system. The weather parameters should be able to be displayed, analyzed, and monitored via ThingSpeak, which connects users to the internet and is visible everywhere in the world. Thingspeak provides a platform to analyze, monitor a system and connects users to the internet which is viewable from anywhere. With the use of software, the internet, and embedded systems, the Internet of Things (IoT) is taking the lead in offering solutions to many problems.

CHAPTER 2

HARDWARE REQUIREMENTS

Hardware Used

The hardware components used in the project are

- NodeMCU (with ESP8266)
- DHT11 temperature and humidity sensor
- Jumper wires

2.1 NodeMCU (with ESP8266)

It is an open-source firmware and development board specifically designed and targeted for IoT applications. It is designed and manufactured by Espressif Systems. It includes firmware that runs on the ESP8266 Wi-Fi SoC, and hardware which is based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

The **ESP8266** is a low-cost Wi-Fi microchip, with built-in TCP/IP networking stack software and microcontroller capability. This simple module allows the microcontrollers to connect to WiFi and make connections and exchange the data.

It can be powered using the Micro-USB and also by supplying the regulated 3.3V the “3.3V” pin. It consists of 30 pins. It contains only one Analog pin, which is used to measure analog voltage in the range of 0-3.3V. It helps in reading data from the analog sensors.

NodeMCU has 16 general purpose input-output pins on its board (GPIO1 - GPIO16). It has four pins available, namely SD1, CMD, SD0, CLK for SPI communication. The Device has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.

Specifications of the device

Microcontroller	Tensilica 32-bit RISC CPU Xtensa LX106
Operating Voltage	3.3V
Flash Memory	4 MB
SRAM	64 KB
Clock Speed	80 MHz
Total pins	30
GPIO pins	16
Analog pin	1

Table 1. Specifications of the NodeMCU device

Pin Diagram

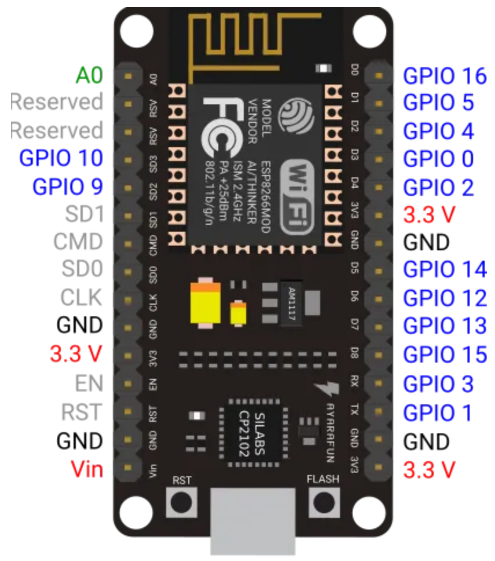


Fig 1. Pin Diagram of NodeMCU

Advantages of NodeMCU

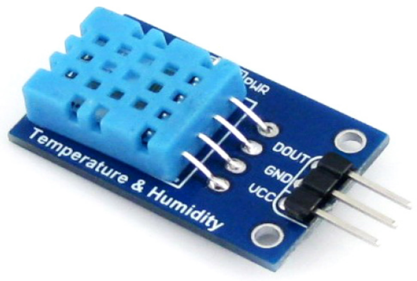
- Inexpensive
- Integrated support for WiFi networks
- lower energy consumption
- Smaller board size

2.2 DHT11 Sensor

The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

2.2.1 Pin Configuration

Vcc: Power supply 3.5V to 5.5V
Data: for outputting the data
Ground: Connected to the ground in circuit.



2.2.2 Specifications

The specifications of the DHT11 sensor are shown in the below table

Operating Voltage	3.5V to 5.5V
Operating current	0.3mA
Output	Serial data (temperature and humidity)
Temperature Range	0°C to 50°C
Humidity Range	20% to 90%
Accuracy	±1°C and ±1%

Table 2. Specifications of the DHT11 sensor.

2.2.3 Communication process

Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4ms. Data consists of decimal and integral parts. A complete data transmission is 40 bit, and the sensor sends higher data bits first.

Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8 bit checksum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

DHT11 switches from low-power consumption to operating mode when MCU delivers a start signal, waiting for MCU to complete the start signal. DHT11 provides a response signal to MCU with 40-bit data that includes relative humidity and temperature information after it's finished. DHT11 will not respond to MCU unless it receives a start signal from it. DHT11 will go into low-power mode once the data has been collected until it receives a start signal from MCU again.

2.3 Jumper Wires

A jump wire is an electrical wire, or a bundle of electrical wires in a cable, with a connector or pin at each end that is generally used to connect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.
(ref: https://en.wikipedia.org/wiki/Jump_wire)

CHAPTER 3

METHODOLOGY

The project involves getting the temperature and humidity data from the DHT11 temperature and humidity sensor. This data is then read from the program written in embedded C++ in the arduino IDE and uploaded to the ThingSpeak cloud, where it is displayed in the forms of graphs over a period of time and they are monitored and any adversary effects in the climatic conditions can be predicted earlier resulting in adopting any precautionary measures.

This project will make use of ThingSpeak, an IoT platform to show the sensor's data. The method is divided into two parts: hardware and software development. The hardware development involves the circuit construction and develops the prototype. Meanwhile, the software involves IoT coding, circuit schematic diagrams, circuit simulations, and data acquisition.

Using the DHT11 sensor to monitor the weather parameters, which are temperature, and humidity, we will be able to display the weather condition by analyzing the current weather with the sensor value data. A microcontroller based WiFi module ESP8266 will control all the data and helps in connecting to the cloud over the internet. Furthermore, this system will also be seen on the ThingSpeak channel created to simplify users checking online. The Internet of Things (IoT) will connect the system with the user wirelessly and online without checking manually at the sensor deployed location.

Project Block Diagram

The Block Diagram of the project is shown in the Figure 2.
The block diagram consists of the entire setup of the hardware and the software components involved. It also includes the connections that has to be made from NodeMCU board to the DHT11 sensor and the data monitoring from the ThingSpeak website.

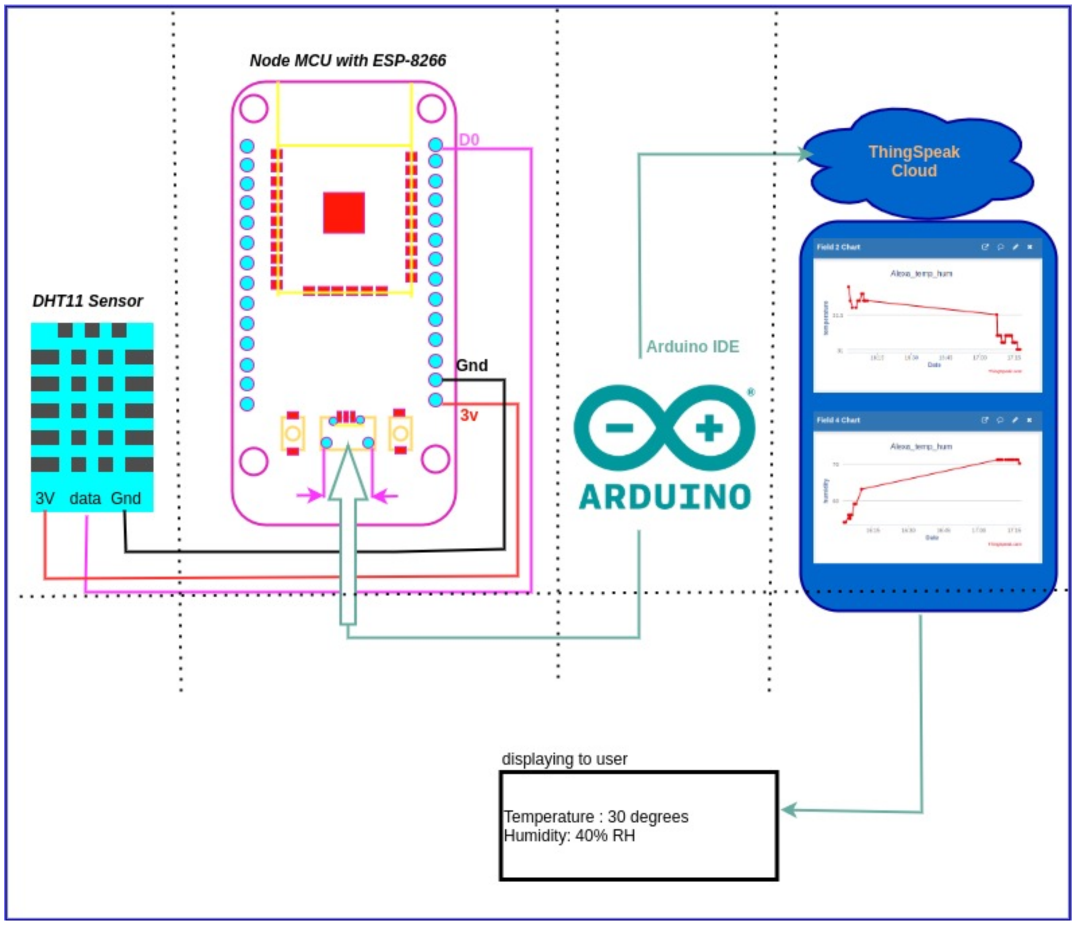
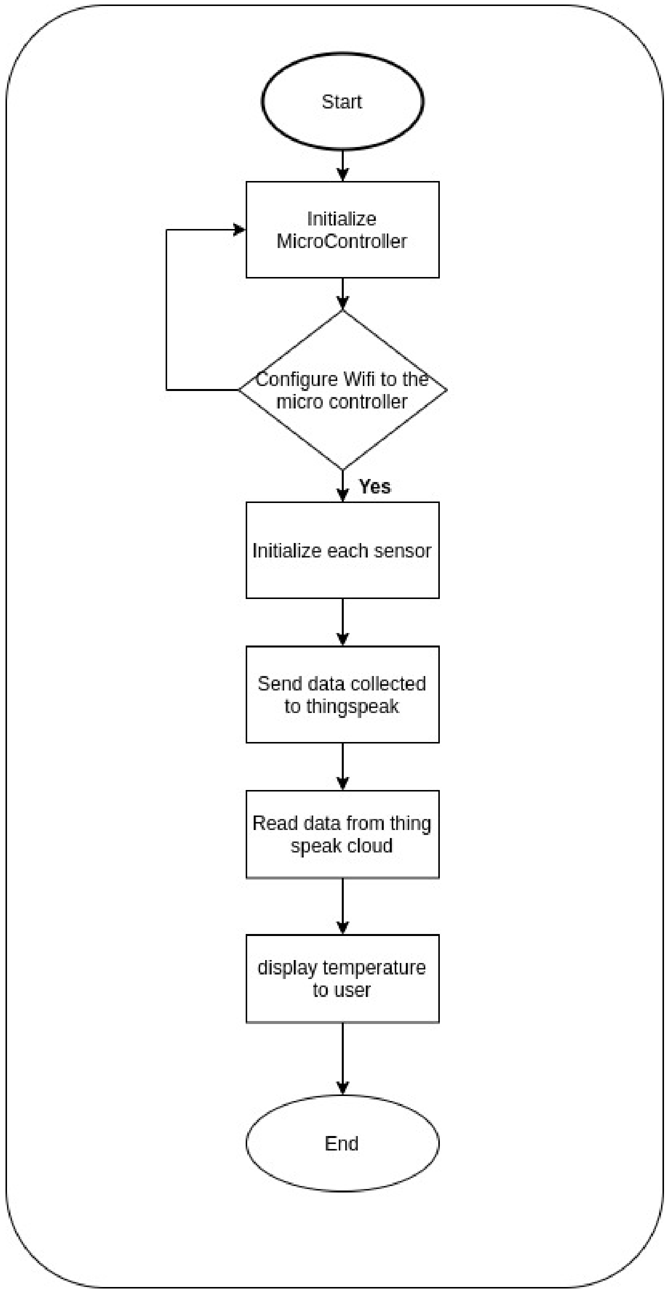


Fig 2. Architecture

Flow Chart

In general, IoT technology is proposed to be used as a communication channel in this project. The system's process begins once the ESP8266 microcontroller has configured all of the sensors followed by powering the NodeMCU board and has begun reading data from them. The data is then sent to the IoT platform, ThingSpeak, through ESP8266 Wi-Fi network wireless communication. The sensors that connect to the ESP8266 serve as the system's control unit, collecting all the data. This system will automatically display the temperature, humidity on a specific IoT web page in ThingSpeak. The following is a flowchart of the process.


Start
Initialize MicroController
Configure Wifi to the micro controller
Yes
Initialize each sensor
Send data collected to thingspeak
Read data from thing speak cloud
display temperature to user
End
Fig 3. Flow Chart

Hardware Development

Hardware selection is very important in any kind of IoT based project. A thorough study regarding all the candidate hardware devices has to be done and select the best one that fits our needs and behave optimally. Here we are using the NodeMCU development board with ESP8266 wifi module embedded in it and DHT11 temperature and humidity sensor.

Circuit Construction:

The connections of the circuit involving NodeMCU and the DHT11 temperature and humidity sensor are shown in Figure 4.

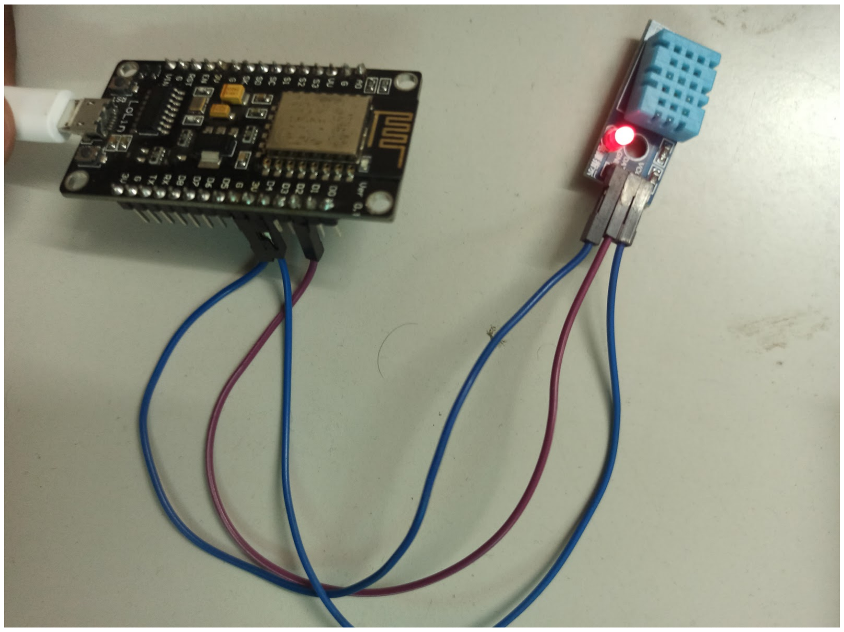


Fig 4. Circuit Connections

The NodeMCU development board is powered using the Micro USB cable connected from our laptop machine here.

“DATA” pin of the DHT11 sensor is connected to the Data pin “D2” of the NodeMCU.
“GND” pin of the DHT11 sensor is connected to the “GROUND” pin of the NodeMCU.
“VCC” pin of the DHT11 sensor is connected to the “3V” pin of the NodeMCU.

Software Development

In general, most projects require the use of various types of software in order to simulate and analyze hardware configurations. At the very least, this method can assist the project participant in troubleshooting and analyzing the project setup and outcome. As a result, just a few types of software will be used to develop this project. The Arduino IDE, which includes tools for compiling and uploading code, is the most helpful tool in this project. The libraries we used for various purposes are as follows.

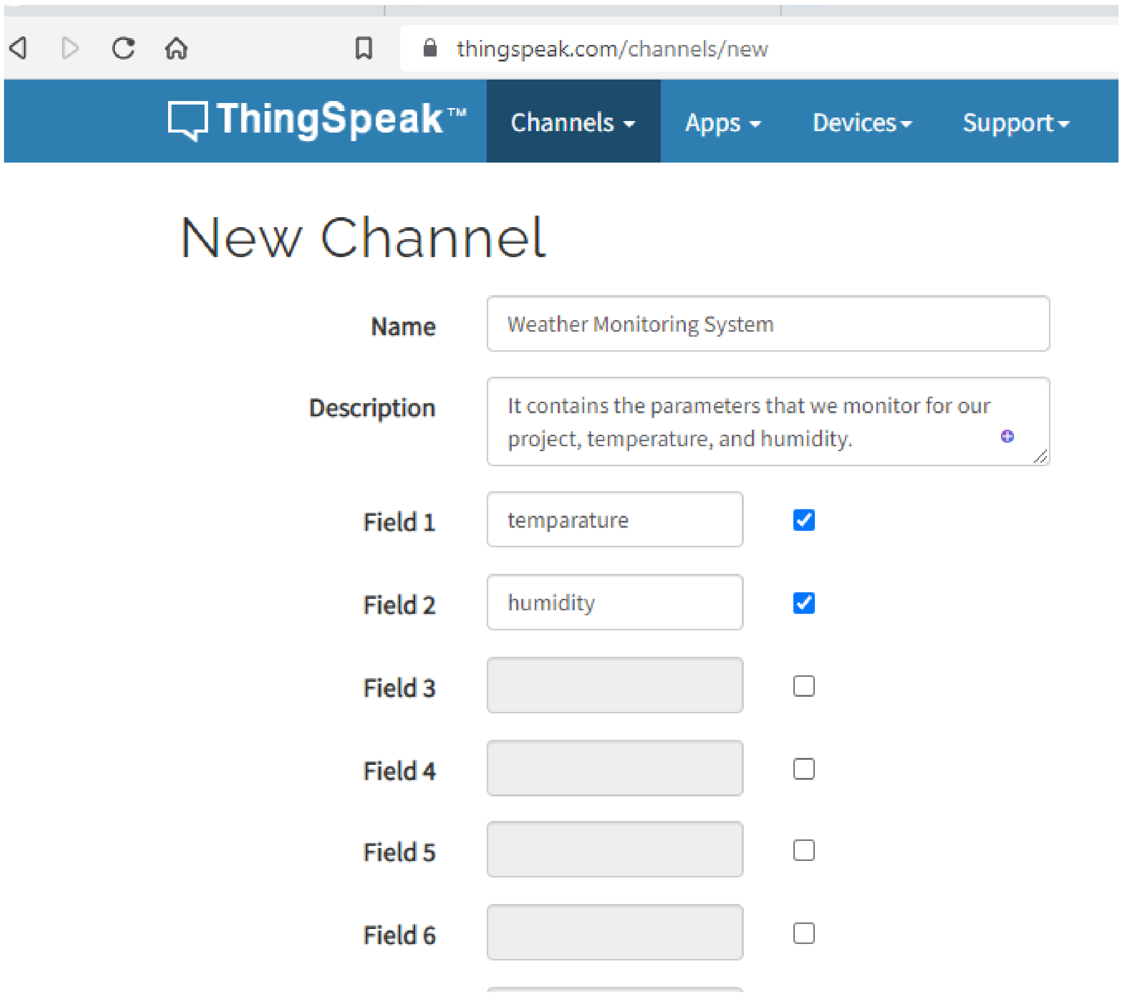
- ESP8266 API
- ThingSpeak API

Protocol Used:

HTTP Protocol is used for Communication. HTTP has the advantage of merely keeping the connection open for a limited time when a device sends and receives data, whereas MQTT must maintain connections open at all times. Because specific systems can only handle a limited number of connections at once, HTTP (when utilized appropriately) can accommodate more traffic.

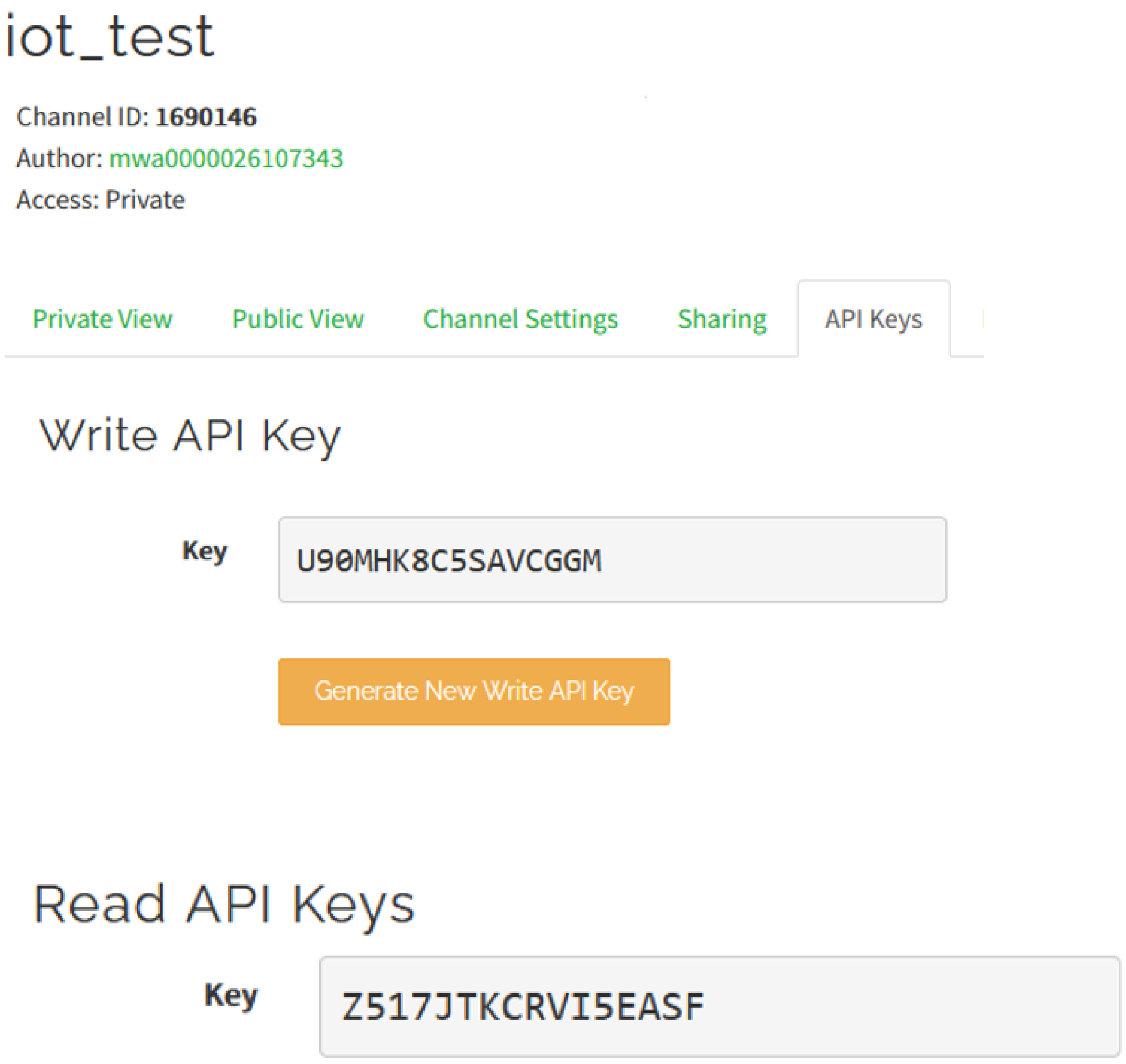
ThingSpeak Setup

ThingSpeak is the IoT platform utilized in this project; new users must create a ThingSpeak account at https://thingspeak.com. Create a new channel called Weather Monitoring System on the website and choose the following fields as weather parameter output results



Name	Weather Monitoring System	
Description	It contains the parameters that we monitor for our project, temperature, and humidity.	
Field 1	temparature	✓
Field 2	humidity	✓
Field 3		☐
Field 4		☐
Field 5		☐
Field 6		☐

Fig 5. Declaring Fields in ThingSpeak



iot_test

Channel ID: **1690146**
Author: mwa0000026107343
Access: Private

Write API Key

Key: U90MHK8C5SAVCGGM

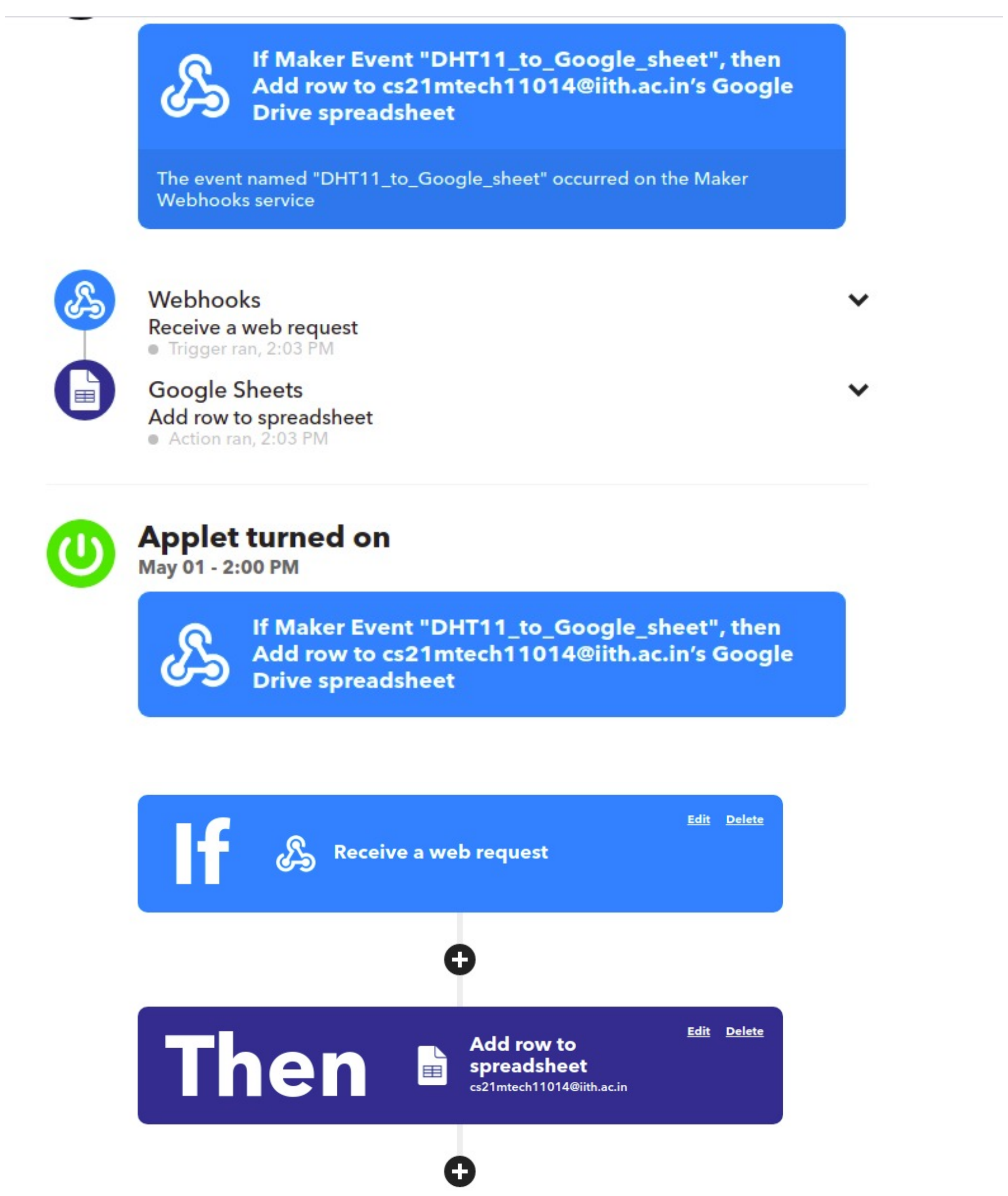
Read API Keys

Key: Z517JTKCRVI5EASF

Fig 6. Creating Write and Read API Keys in ThingSpeak

Google Sheet with IFTTT platform:

Using IFTTT, an IoT platform, we can save data from any IoT device and ensure that it is saved to a Google Sheet. The screenshot below depicts the IFTTT platform built for this project and how the data was successfully recorded in a Google Sheet. With the help of this platform, data can be quickly recorded, and tables can be created.



If Maker Event "DHT11_to_Google_sheet", then Add row to cs21mtech11014@iith.ac.in's Google Drive spreadsheet

The event named "DHT11_to_Google_sheet" occurred on the Maker Webhooks service

Webhooks
Receive a web request
● Trigger ran, 2:03 PM

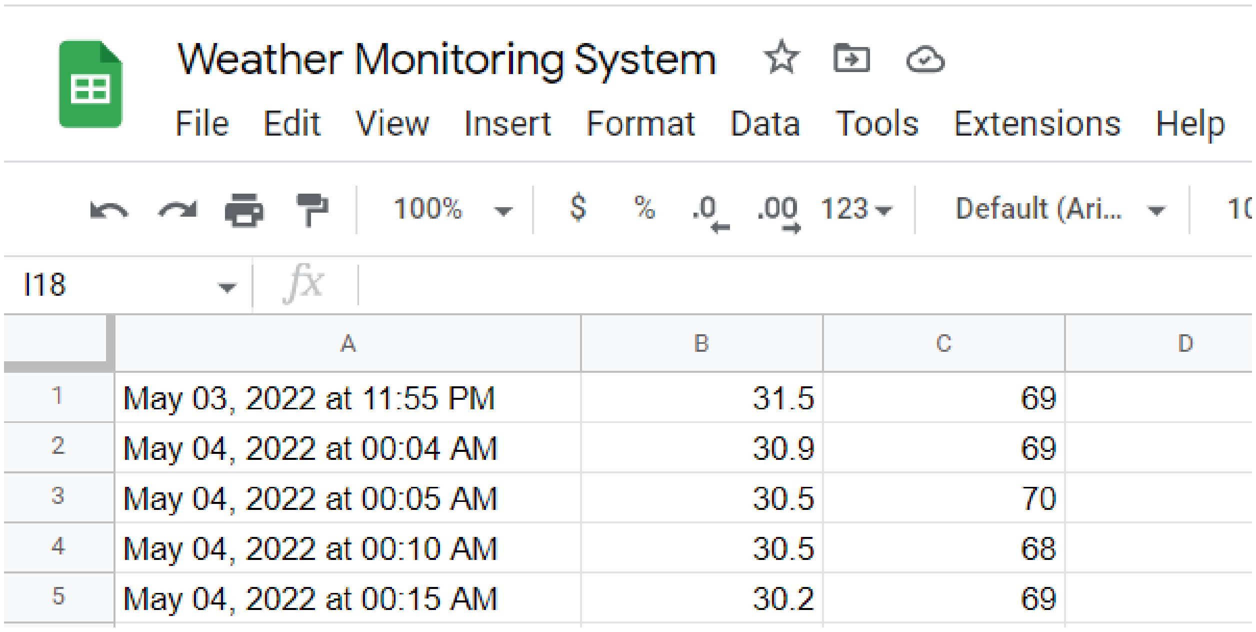
Google Sheets
Add row to spreadsheet
● Action ran, 2:03 PM

Applet turned on
May 01 - 2:00 PM

If Maker Event "DHT11_to_Google_sheet", then Add row to cs21mtech11014@iith.ac.in's Google Drive spreadsheet

Fig 7. IFTTT Applet

With the use of this platform, data can be quickly recorded, and Table 1 may be completed without verifying the sensor station physically. Go to the IFTTT website, sign up for a free account, build a new project that connects the sensor station to the server, and store it in Google Drive using Google Sheets.

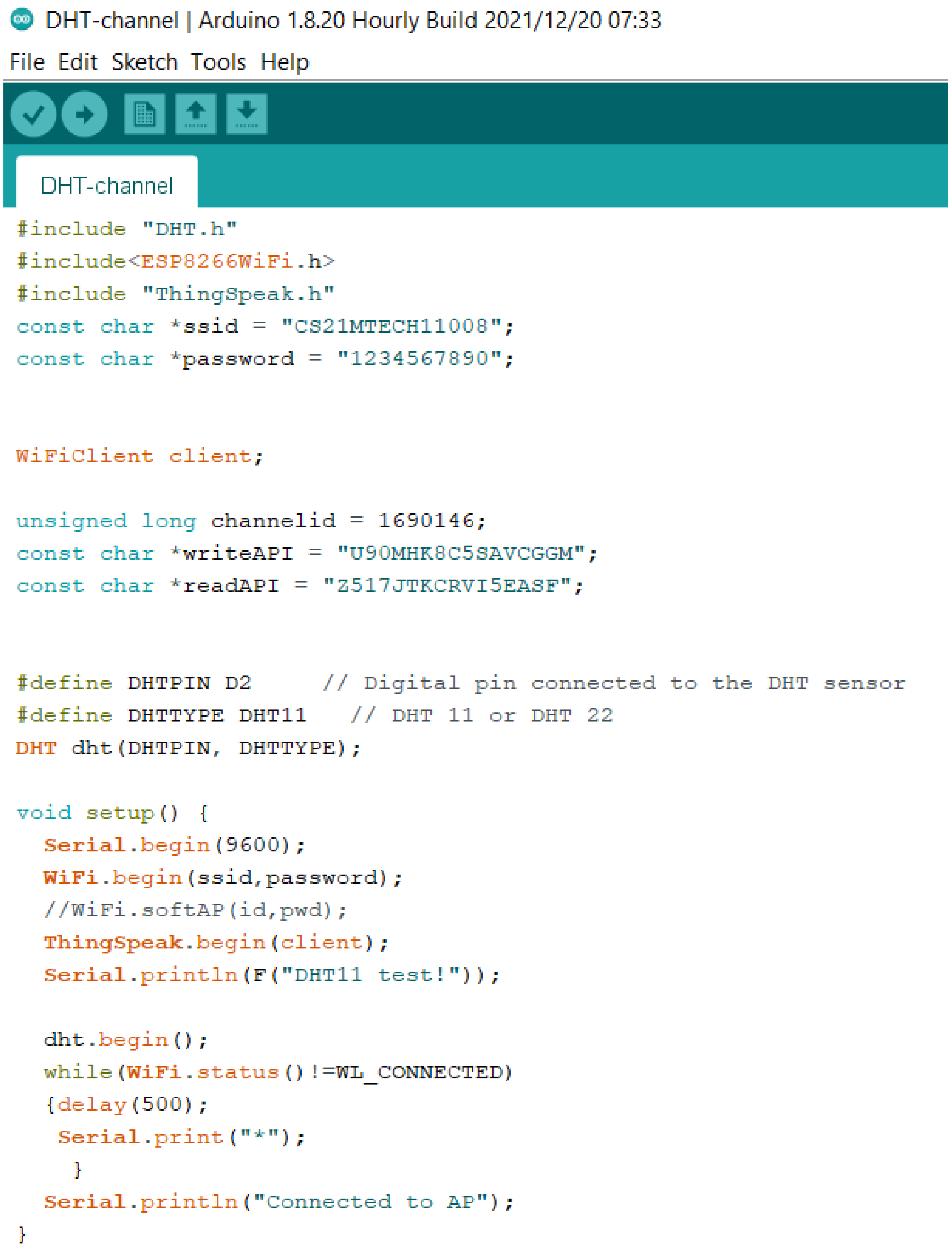

Weather Monitoring System
File Edit View Insert Format Data Tools Extensions Help
100%
$ % .0 .00 123
Default (Ari...
I18

	A	B	C	D
1	May 03, 2022 at 11:55 PM	31.5	69	
2	May 04, 2022 at 00:04 AM	30.9	69	
3	May 04, 2022 at 00:05 AM	30.5	70	
4	May 04, 2022 at 00:10 AM	30.5	68	
5	May 04, 2022 at 00:15 AM	30.2	69	

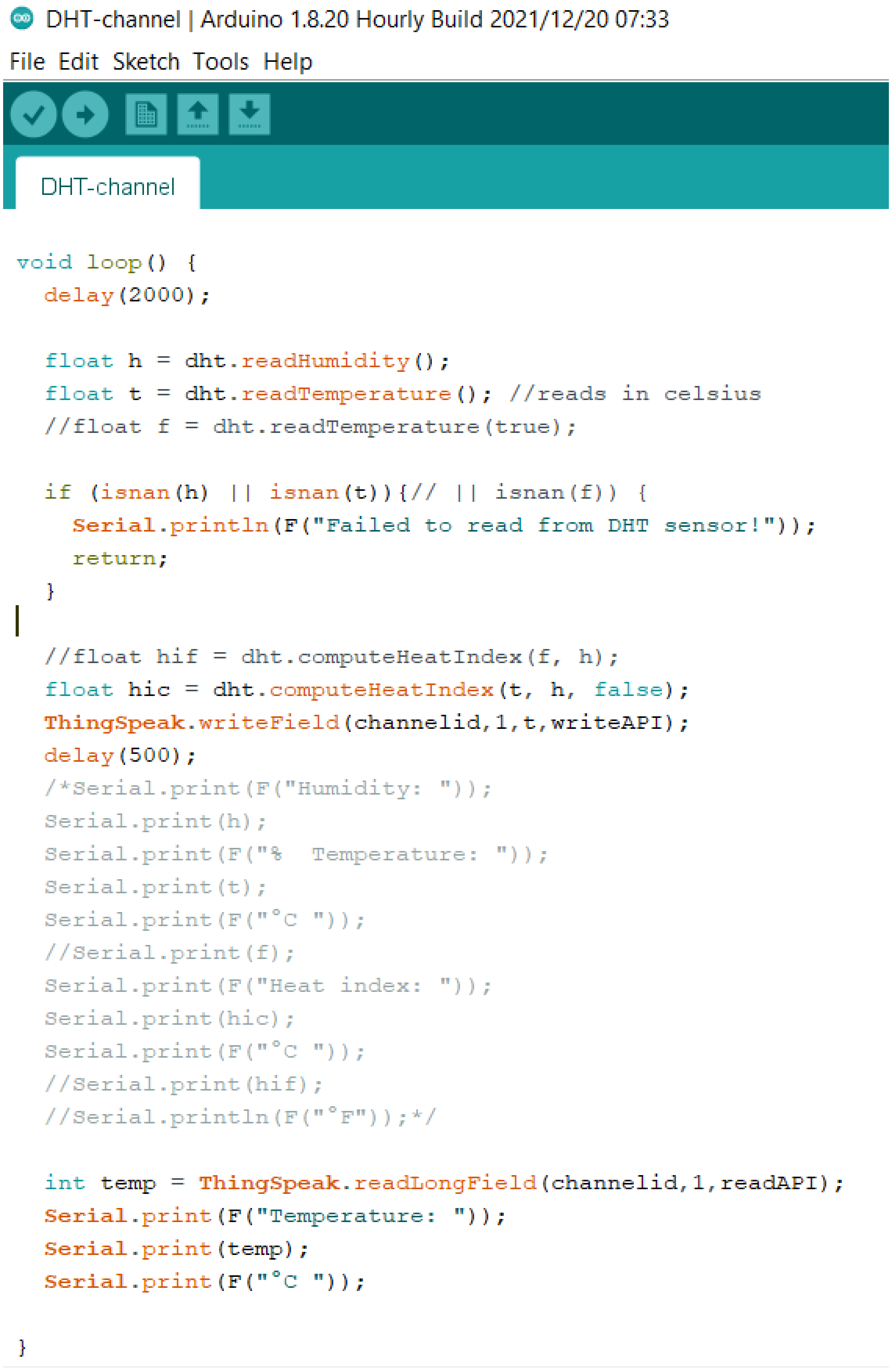
Fig 8. Data from Google Sheet

CHAPTER 4

SOURCE CODE

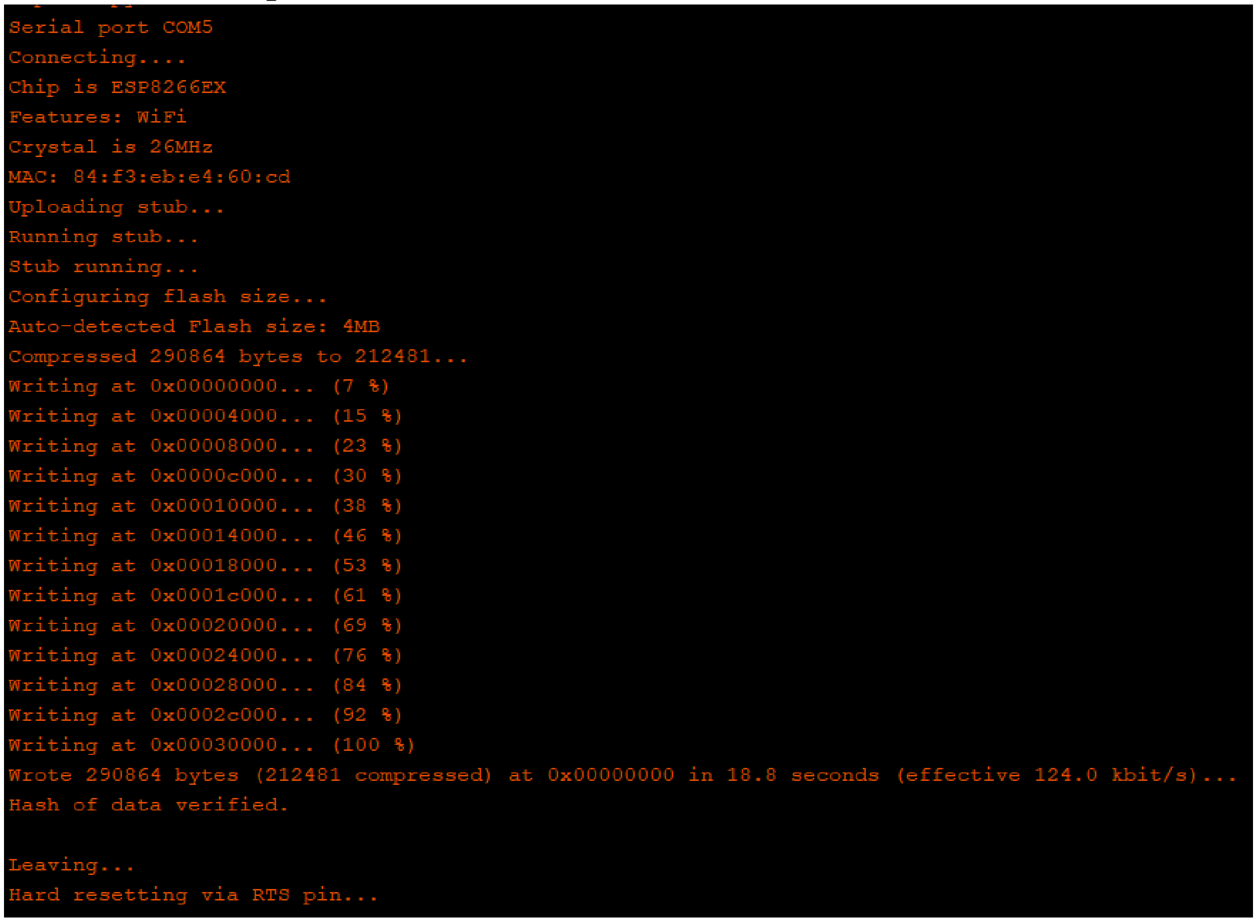

DHT-channel | Arduino 1.8.20 Hourly Build 2021/12/20 07:33
File Edit Sketch Tools Help
DHT-channel

```
#include "DHT.h"
#include<ESP8266WiFi.h>
#include "ThingSpeak.h"
const char *ssid = "CS21MTECH11008";
const char *password = "1234567890";


WiFiClient client;

unsigned long channelid = 1690146;
const char *writeAPI = "U90MHK8C5SAVCGGM";
const char *readAPI = "Z517JTKCRVI5EASF";


#define DHTPIN D2     // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11   // DHT 11 or DHT 22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid,password);
  //WiFi.softAP(id,pwd);
  ThingSpeak.begin(client);
  Serial.println(F("DHT11 test!"));

  dht.begin();
  while(WiFi.status()!=WL_CONNECTED)
  {delay(500);
   Serial.print("*");
    }
  Serial.println("Connected to AP");
}
```


DHT-channel | Arduino 1.8.20 Hourly Build 2021/12/20 07:33
File Edit Sketch Tools Help
DHT-channel

```
void loop() {
  delay(2000);

  float h = dht.readHumidity();
  float t = dht.readTemperature(); //reads in celsius
  //float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t)){// || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  //float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  ThingSpeak.writeField(channelid,1,t,writeAPI);
  delay(500);
  /*Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  //Serial.print(f);
  Serial.print(F("Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  //Serial.print(hif);
  //Serial.println(F("°F"));*/

  int temp = ThingSpeak.readLongField(channelid,1,readAPI);
  Serial.print(F("Temperature: "));
  Serial.print(temp);
  Serial.print(F("°C "));

}
```

Command Prompt:

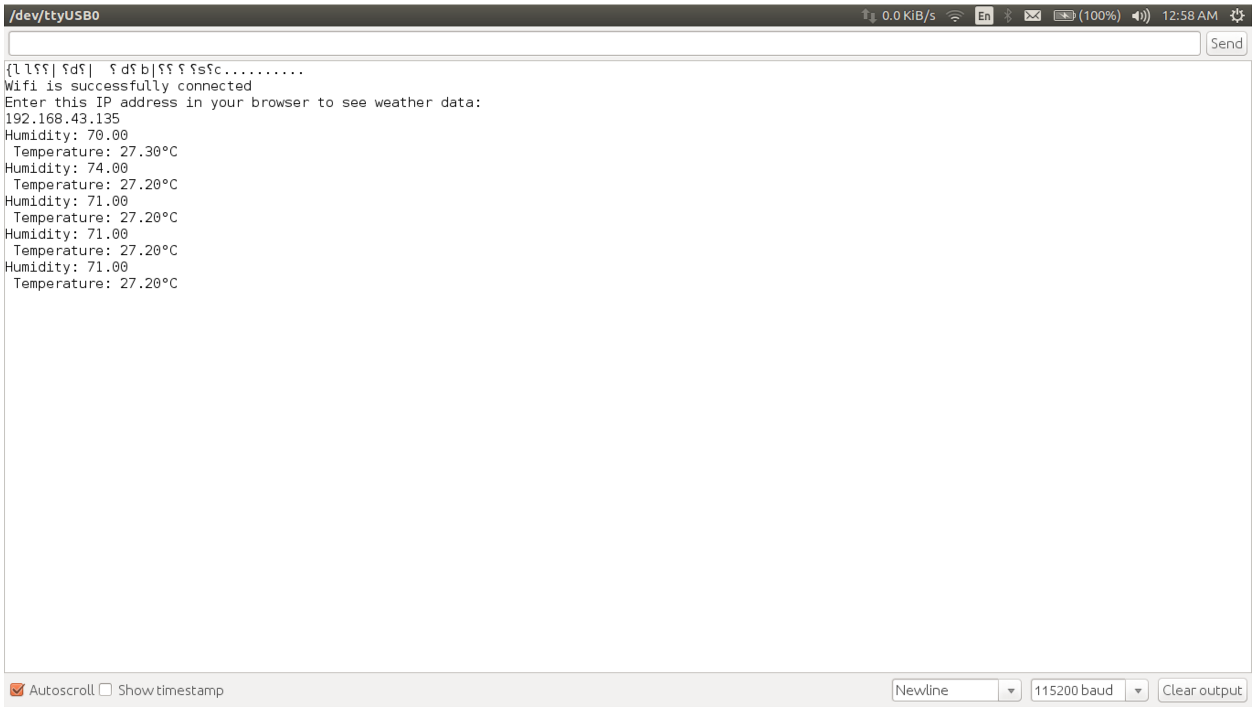


```
Serial port COM5
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 84:f3:eb:e4:60:cd
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 290864 bytes to 212481...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (15 %)
Writing at 0x00008000... (23 %)
Writing at 0x0000c000... (30 %)
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 290864 bytes (212481 compressed) at 0x00000000 in 18.8 seconds (effective 124.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

CHAPTER 5

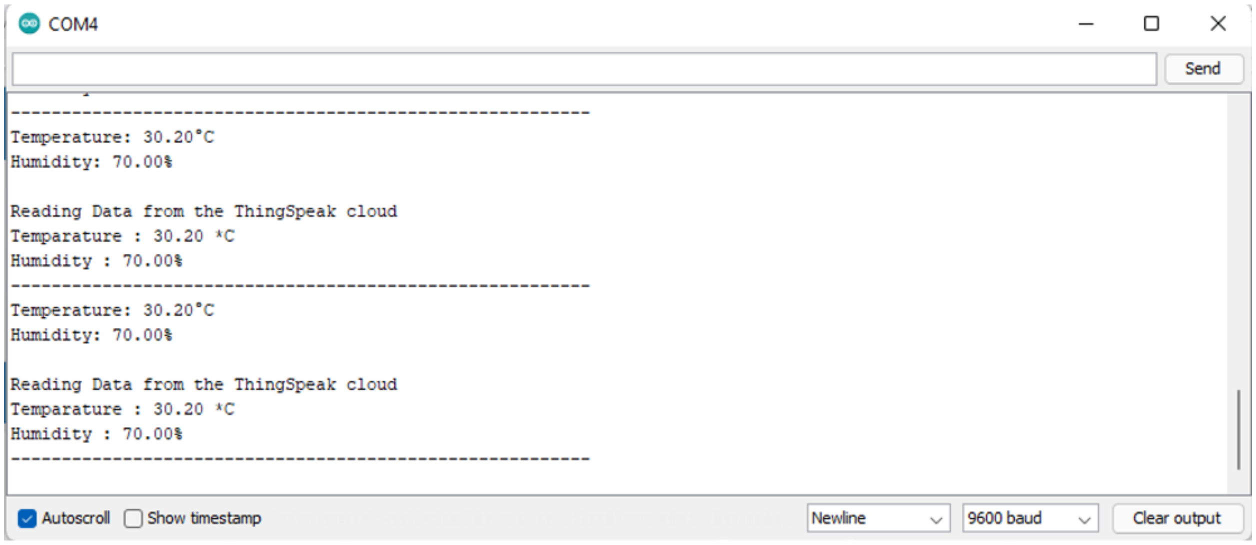
RESULTS

After collecting data from several sensor devices placed in a particular area of interest, the detected data will be automatically delivered to the webserver when a proper connection is established with the server. The web server page from which we will monitor and control the system. The web page provides information about the temperature and humidity in the area where the embedded monitoring device is located. The collected data will be saved in the cloud (Google Spread Sheets). The data stored in the cloud can be used for parameter analysis and continuous monitoring. All of the above data will be saved in the cloud so that we may provide trending temperature and humidity levels in a specific area at any time. To access the temperature and humidity data we need to access the IP address through which the NodeMCU is connected. This IP address will be displayed on the Arduino IDE. Using that IP address user can see the data in web app. Here the NodeMCU and the user should connect to the same network.



```
{l l⸮⸮| ⸮d⸮|  ⸮ d⸮ b|⸮⸮ ⸮ ⸮s⸮c..........
Wifi is successfully connected
Enter this IP address in your browser to see weather data:
192.168.43.135
Humidity: 70.00
 Temperature: 27.30°C
Humidity: 74.00
 Temperature: 27.20°C
Humidity: 71.00
 Temperature: 27.20°C
Humidity: 71.00
 Temperature: 27.20°C
Humidity: 71.00
 Temperature: 27.20°C
```

Fig 9: showing IP address on the IDE



```
------------------------------------------------------------
Temperature: 30.20°C
Humidity: 70.00%

Reading Data from the ThingSpeak cloud
Temparature : 30.20 *C
Humidity : 70.00%
------------------------------------------------------------
Temperature: 30.20°C
Humidity: 70.00%

Reading Data from the ThingSpeak cloud
Temparature : 30.20 *C
Humidity : 70.00%
------------------------------------------------------------
```

Newline · 9600 baud

Fig 10. Serial Monitor Screenshot

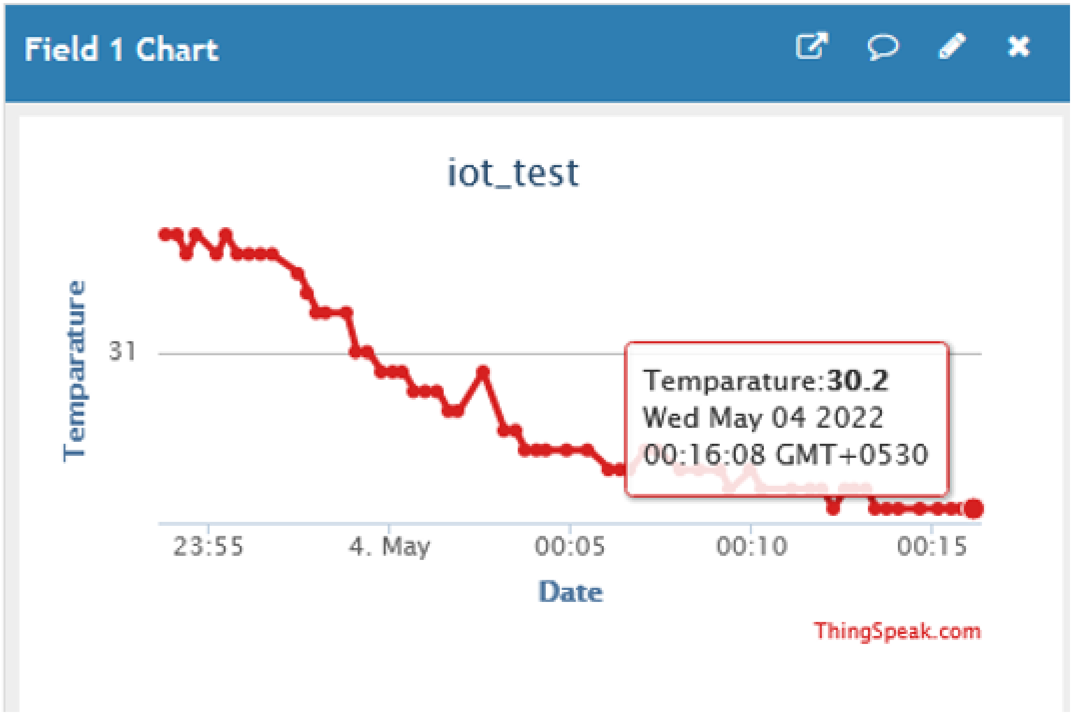


Fig 11. Simulation of Temperature v/s Time

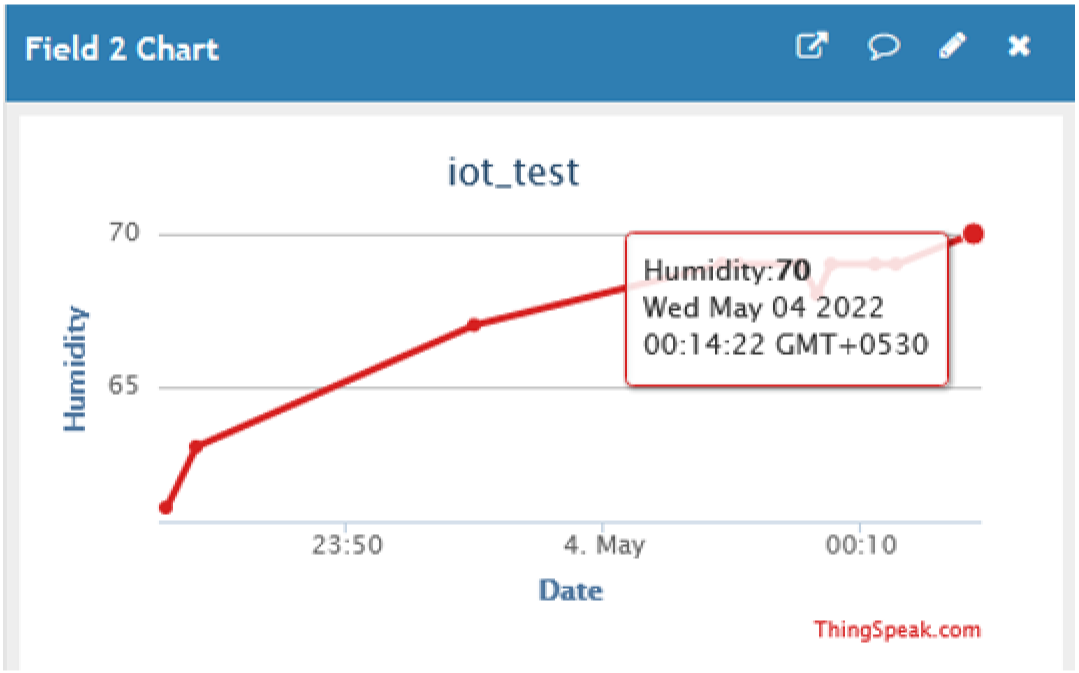
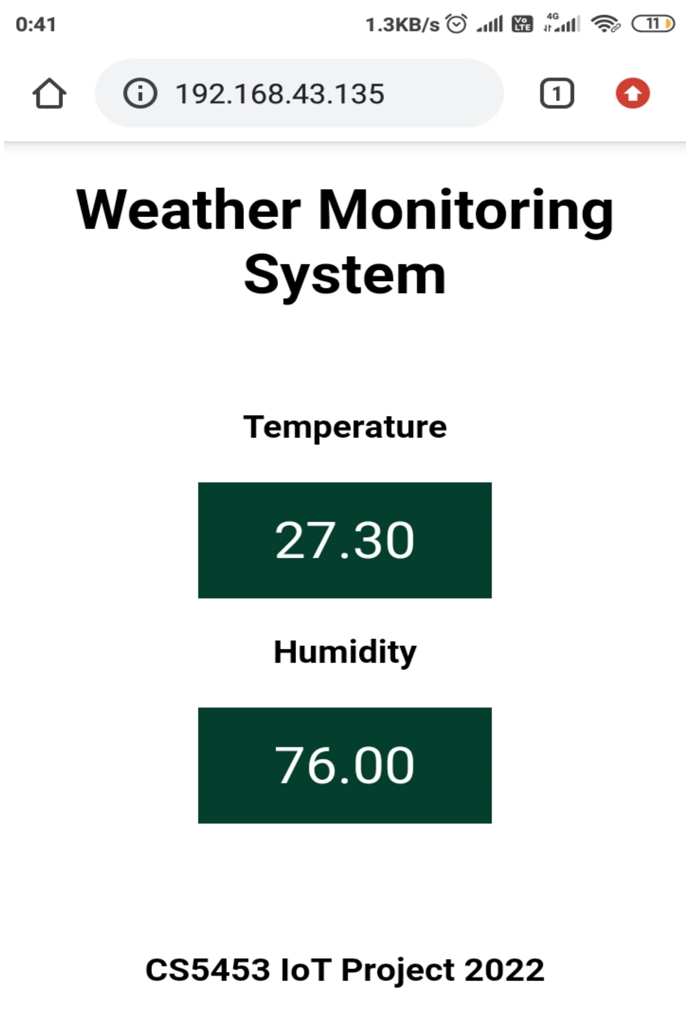


Fig 12. Simulation of Humidity v/s Time


0:41 1.3KB/s

192.168.43.135

Weather Monitoring System

Temperature

27.30

Humidity

76.00

CS5453 IoT Project 2022

Fig13: Accessing the data from a web application

CHAPTER 6

CONCLUSION

Keeping embedded devices in the environment for monitoring allows the environment to protect itself. To execute this, sensor devices must be deployed in the environment for data collection and processing. We can bring the environment to life by placing sensor devices in it, allowing it to interact with other objects over the network. The collected data and analysis results will be made available to the end-user via Wi-Fi. This study presents various concepts for an intelligent way to monitor the environment and an efficient, low-cost embedded system. The functions of several modules were discussed in the suggested architecture. Temperature and humidity may be tracked using the Internet of Things (IoT) concept, experimentally verified for two parameters. It also transmits sensor data to the cloud (Google Spread Sheets). This information will be helpful for future analyses and may be shared with other end users. This model can be expanded further to monitor developing cities and industrial zones for weather monitoring.
This methodology provides an efficient and low-cost solution for continuous environmental monitoring to protect public health from pollution.

REFERENCES

- Shifeng Fang; Li Da Xu; Yunqiang Zhu; JiaerhengAhati; Huan Pei; Jianwu Yan; Zhihui Liu., “An integrated system for regional environmental monitoring and management based on internet of things”, IEEE Transactions on Industrial Informatics,vol.10, no. 2,pp.1596-1605, May-Jun. 2014.
- BulipeSrinivasRao , Prof. Dr. K. SrinivasaRao , Mr. N. Ome, “Internet of Things (IOT) Based Weather Monitoring system”, IJARCCE Journal,vol. 5, no. 9, sept. 2016
- Kulkarni, V. A, Satpute G. M (2017). “Weather Reporting System Using FPGA : A Review,” vol. 4, no. 11, pp. 319–320.
- Joe F, and Joseph J (2019). “IoT Based Weather Monitoring System for Effective Analytics,” International Journal of Engineering and Advanced Technology (IJEAT), no. 4, pp. 311– 315.