

UE19CS313
The Internet of Things (IoT)



**The Assignment of IoT
(UE19CS313)**

Implementation of a strict IoT application.

Team members	SRN
Achyut Jagini	PES2UG19CS013
Achyuta B Mudhol	PES2UG19CS014
Anurag.R.Simha	PES2UG19CS052

Section: A

Semester: 5

Title: Heater and Fan Manoeuvring System.

a) Identification of a use case for the physical computing project.

The device designed aids a user in switching modes between heater and fan control. The presence of a control switch on the circuit board performs this action. According to the weather, the user has the freedom to select either of the choices. The choice of the user and the temperature/speed is transferred to the cloud storage.

b) Listing the features planned.

Imperative features:

1. Switch to control the operation between the heater and fan.
2. A temperature sensor to alter the temperature of the heater.
3. A potentiometer to manoeuvre the speed of the fan.
4. A wi-fi module (ESP8266) to connect the circuit to the internet.
5. All manipulation is done over the Arduino board.

Discretionary features:

1. A servo motor as an actuator to examine the triumphant functioning of the components.

c) Listing the requirements of SW and HW components to realise the project.

Software components:

1. C Program.
2. Arduino board IDE,

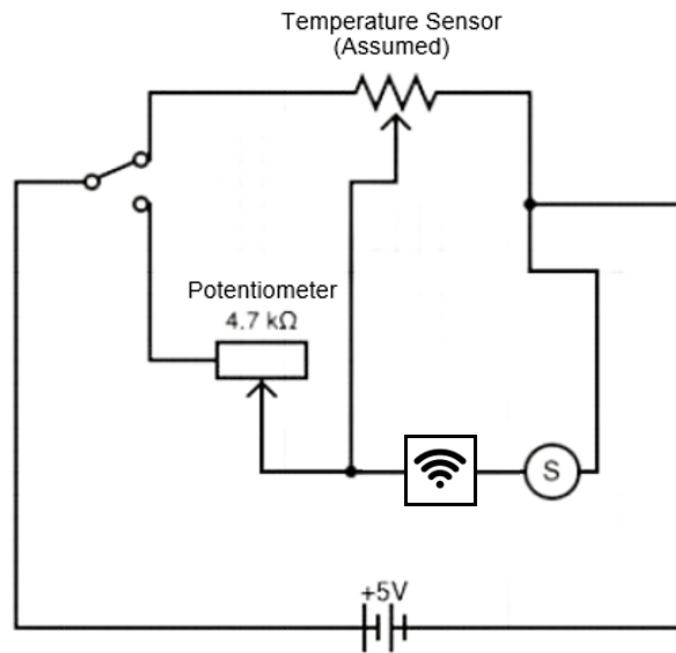
Hardware components:

1. Arduino UNO board
2. Breadboard
3. Servo motor
4. Temperature sensor
5. Switch
6. Wi-fi module

d) Coming up with the circuit design for the project.

P.T.O

The Internet of Things (IoT)



e) Coming up with the necessary logic to implement all the listed features in (b).

```
#include <Servo.h>

#include <math.h>

float reading=0;

int duty;

int angle;

int switch_pin = 12;

int LED_PIN = 10;

float voltage = 0;

String httpPacket;

String ssid      = "Simulator Wifi"; // SSID to connect to

String password = ""; //virtual wifi has no password

String host      = "api.thingspeak.com"; // Open Weather Map API
```

```

const int httpPort    = 80;

String url            = "/update?api_key=XPT0V1HQ91F976H4&field1=";

//Replace XXXXXXXXXXXXXXX by your ThingSpeak Channel API Key

Servo servo_11;

void setupESP8266(void) {

    // Start our ESP8266 Serial Communication

    Serial.begin(115200);    // Serial connection over USB to computer

    Serial.println("AT");    // Serial connection on Tx / Rx port to
ESP8266

    delay(10);                // Wait a little for the ESP to respond

    if (Serial.find("OK"))

        Serial.println("ESP8266 OK!!!");

    // Connect to Simulator Wifi

    Serial.println("AT+CWJAP=\"\" + ssid + "\",\"\" + password + "\"");

    delay(10);                // Wait a little for the ESP to respond

    if (Serial.find("OK"))

        Serial.println("Connected to WiFi!!!");

    // Open TCP connection to the host:

    //ESP8266 connects to the server as a TCP client.

    Serial.println("AT+CIPSTART=\"TCP\", \"\" + host + "\", \"\" + httpPort);

    delay(50);                // Wait a little for the ESP to respond

    if (Serial.find("OK"))

        Serial.println("ESP8266 Connected to server!!!");

```

The Internet of Things (IoT)

```
}

void setup() {

    Serial.begin(9600);

    setupESP8266();

    pinMode(switch_pin, INPUT);

    pinMode(A0, INPUT);

    pinMode(LED_PIN, OUTPUT);

    servo_11.attach(11);

    pinMode(A1, INPUT);

}

void loop() {

    if (digitalRead(switch_pin) == 0)

    {

        reading = analogRead(A0);

        duty= map(reading,0,1023,0,225);

        analogWrite(LED_PIN, duty);

        angle= map(reading,0,1023,0,180);

        servo_11.write(angle);

        httpPacket = "GET " + url + "0" + "&field2=" +
String(digitalRead(switch_pin))+"&field3="+String(reading)+ "
HTTP/1.1\r\nHost: " + host + "\r\n\r\n";

    }

    else

    {
```

```
reading = analogRead(A1);

duty= map(reading,20,359,0,255);

analogWrite(LED_PIN, duty);

angle= map(reading,20,359,0,180);

servo_11.write(angle);

voltage=reading*0.0048828125;

reading = (voltage - 0.5) * 100.0;

reading = round(reading);

httpPacket = "GET " + url + String(reading) + "&field2=" +
String(digitalRead(switch_pin))+"&field3=0"+ " HTTP/1.1\r\nHost: " +
host + "\r\n\r\n";

}

Serial.print(reading);

Serial.print('\n');

Serial.print(digitalRead(switch_pin));

Serial.print('\n');

Serial.print('\n\n');

int length = httpPacket.length();

//int length1 = httpPacket1.length();

// Send our message length

Serial.print(String(digitalRead(switch_pin)) +String(reading));

Serial.print("AT+CIPSEND=");

Serial.println(length);

//Serial.print("AT+CIPSEND=");
```

The Internet of Things (IoT)

```
//Serial.println(length1);

delay(10); // Wait a little for the ESP to respond if
(!Serial.find(">")) return -1;

// Send our http request

Serial.print(httpPacket);

delay(10); // Wait a little for the ESP to respond

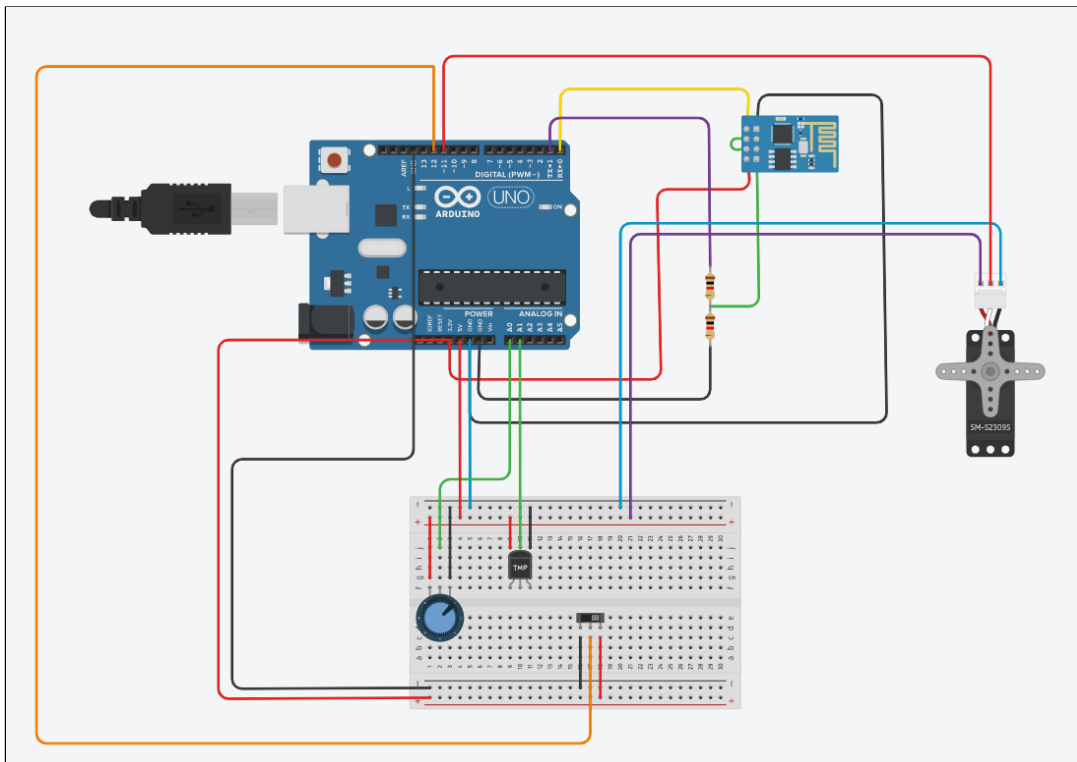
if (Serial.find("SEND OK\r\n"))

    Serial.println("ESP8266 sends data to the server");

delay(100);
}
```

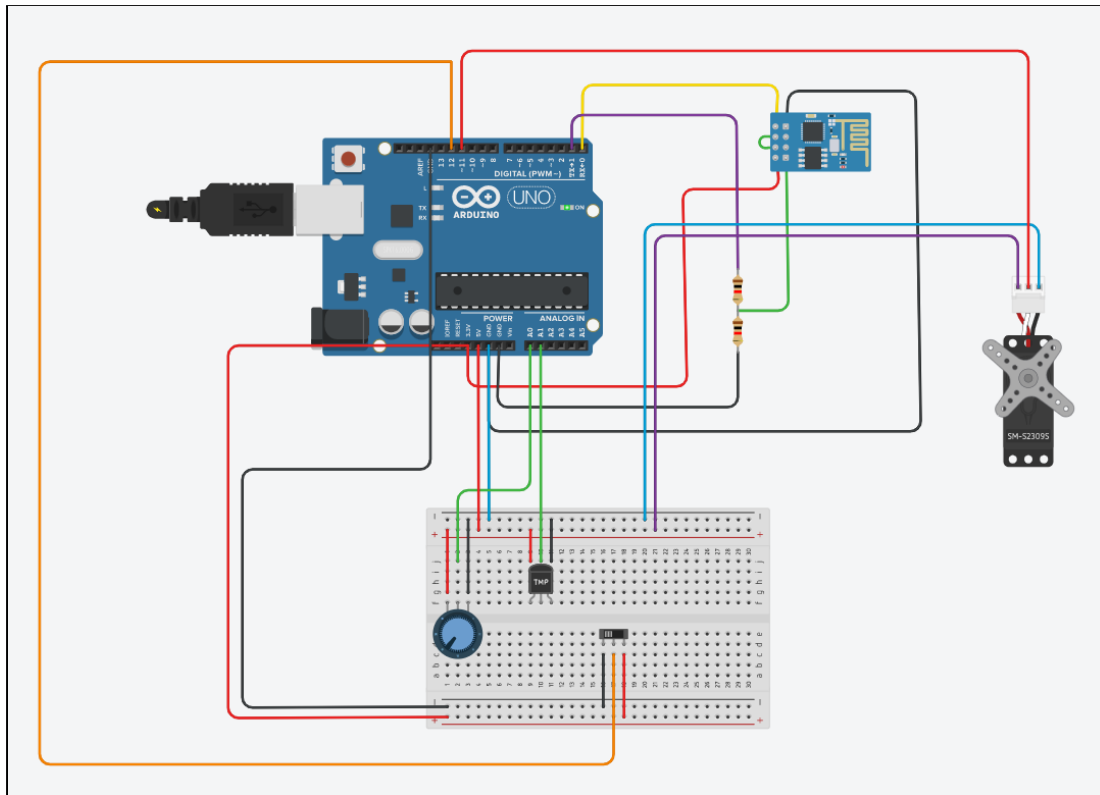
f) Testing and Packaging the circuit.

The finalised circuit:

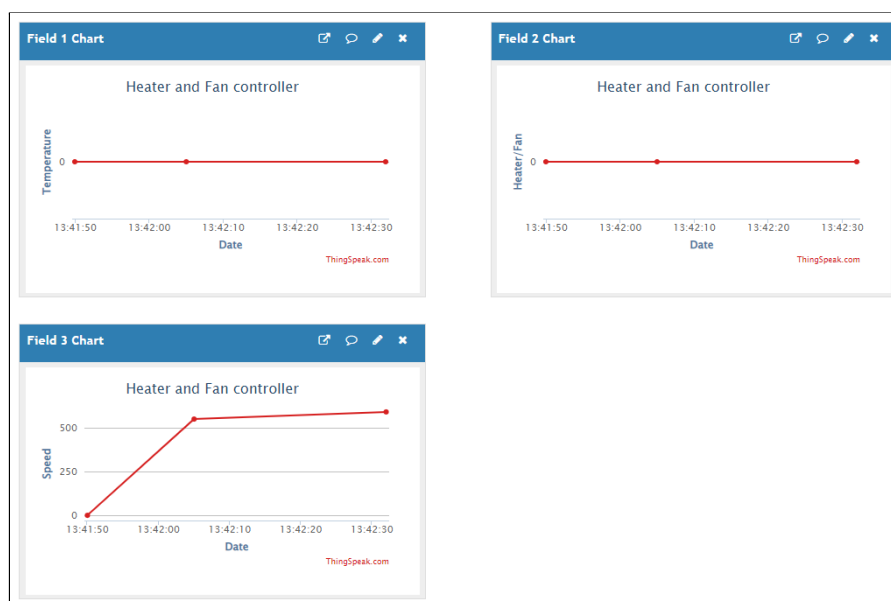


Testing/Simulating the devised circuit and analysing the results:

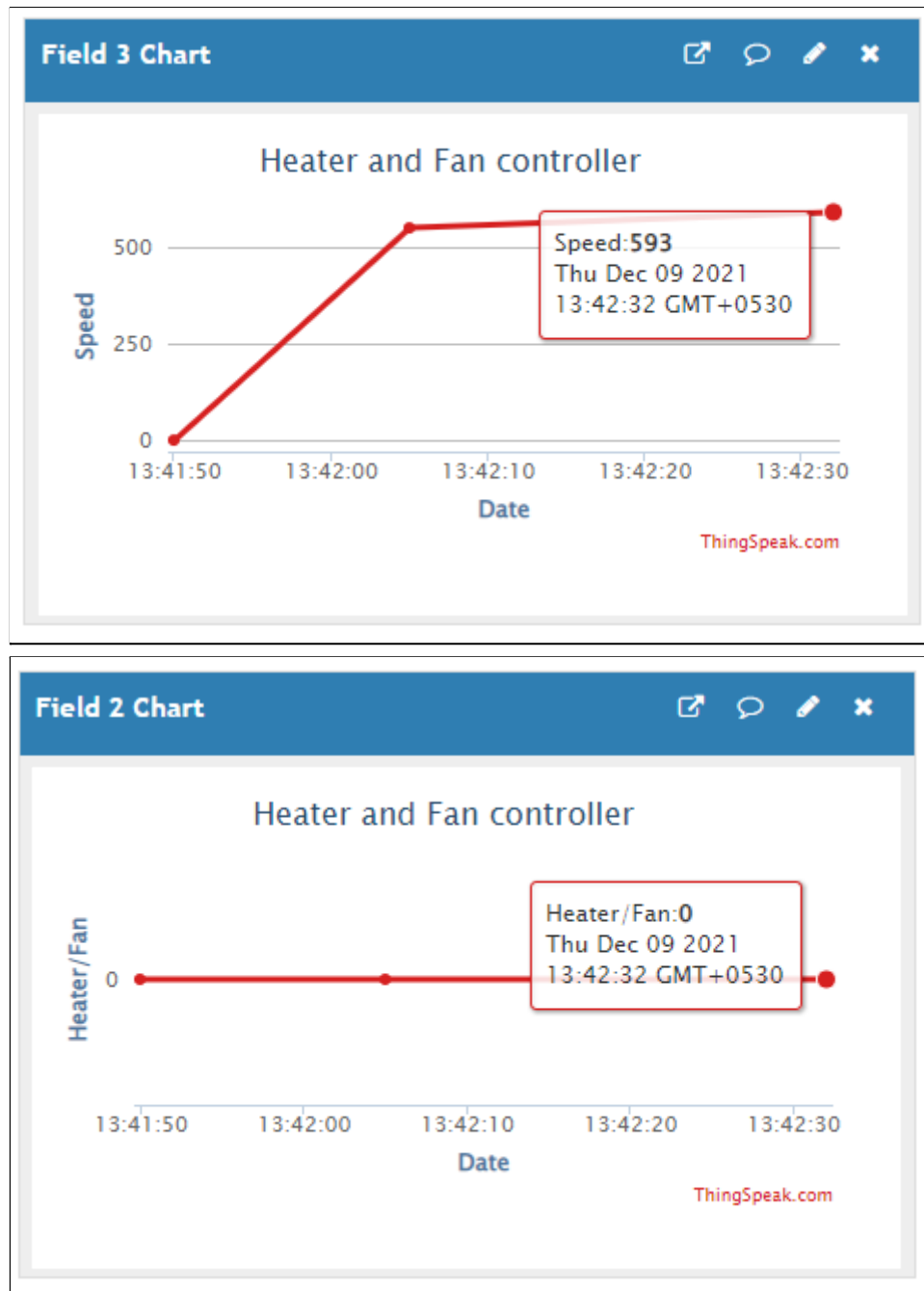
1. Manipulating the speed (fan):



The potentiometer that's present in the image above functions as a regulator, aiding the sine qua non of fan control. From 0, it ascends to 593. This data and the mode selected got stored on the cloud in **ThingSpeak**. The image below is the result.



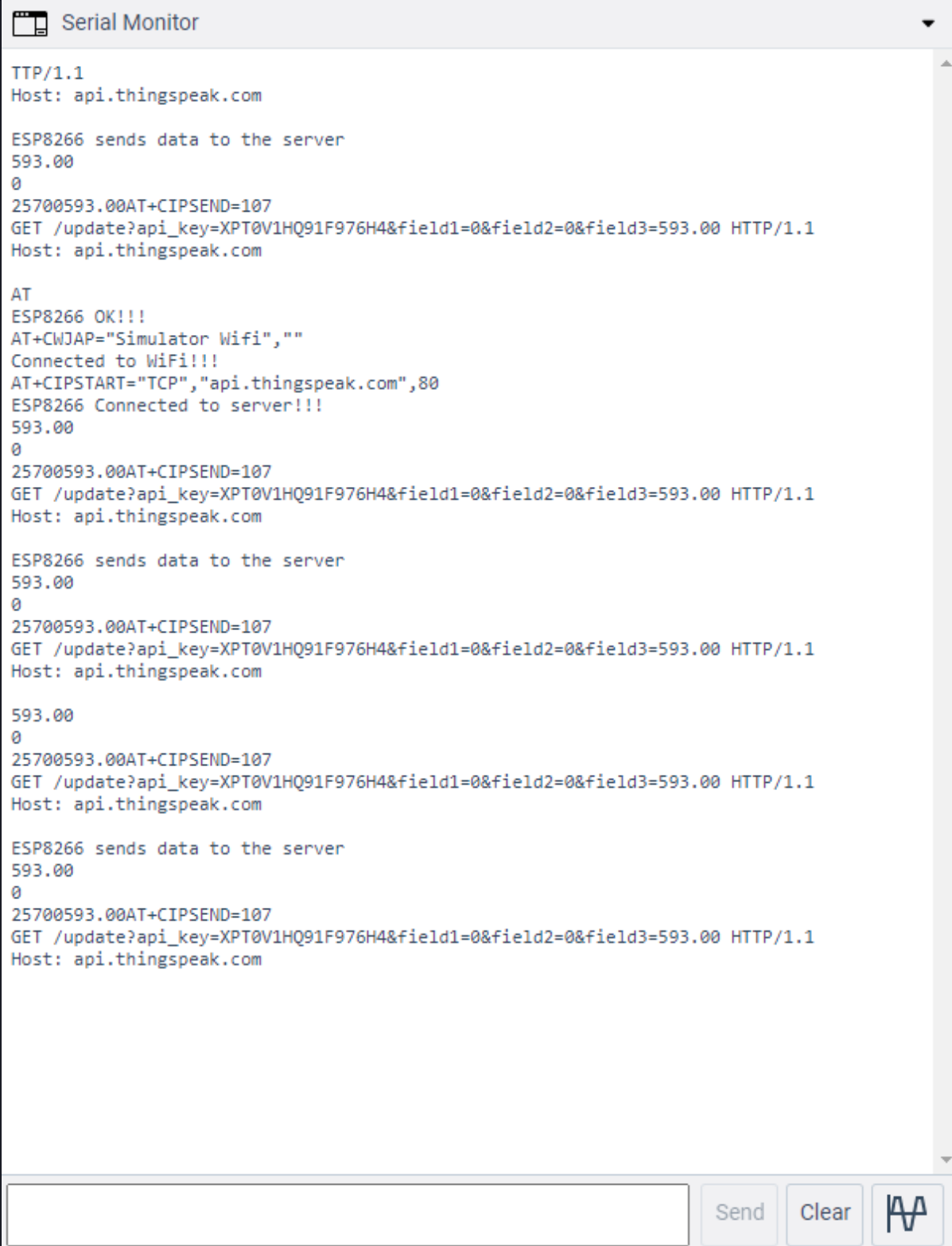
The Internet of Things (IoT)



From the graphs, it can be manifested that there was a rise in speed and that was done under mode zero. The first graph depicts (field 1 chart) the temperature. The second graph (field 2 chart) depicts if the mode selected is a heater or a fan. Then, the third (field 3 chart) plots the speed graph. Here, it's a fan. When the program is allowed to run with no manoeuvring (of the speed), the graph is a straight line, depicting a constant value. It can be deemed with no qualm that it was fan control. For, at 13 hours, 42 minutes and 32 seconds, the value had ascended.

- P.T.O -

These were the results on the serial monitor:



The screenshot shows a Serial Monitor window with a title bar and a dropdown arrow. The main area displays a series of text messages, including HTTP requests and AT commands. At the bottom, there is a text input field, a 'Send' button, a 'Clear' button, and a waveform icon.

```

TTP/1.1
Host: api.thingspeak.com

ESP8266 sends data to the server
593.00
0
25700593.00AT+CIPSEND=107
GET /update?api_key=XPT0V1HQ91F976H4&field1=0&field2=0&field3=593.00 HTTP/1.1
Host: api.thingspeak.com

AT
ESP8266 OK!!!
AT+CWJAP="Simulator Wifi",""
Connected to WiFi!!!
AT+CIPSTART="TCP","api.thingspeak.com",80
ESP8266 Connected to server!!!
593.00
0
25700593.00AT+CIPSEND=107
GET /update?api_key=XPT0V1HQ91F976H4&field1=0&field2=0&field3=593.00 HTTP/1.1
Host: api.thingspeak.com

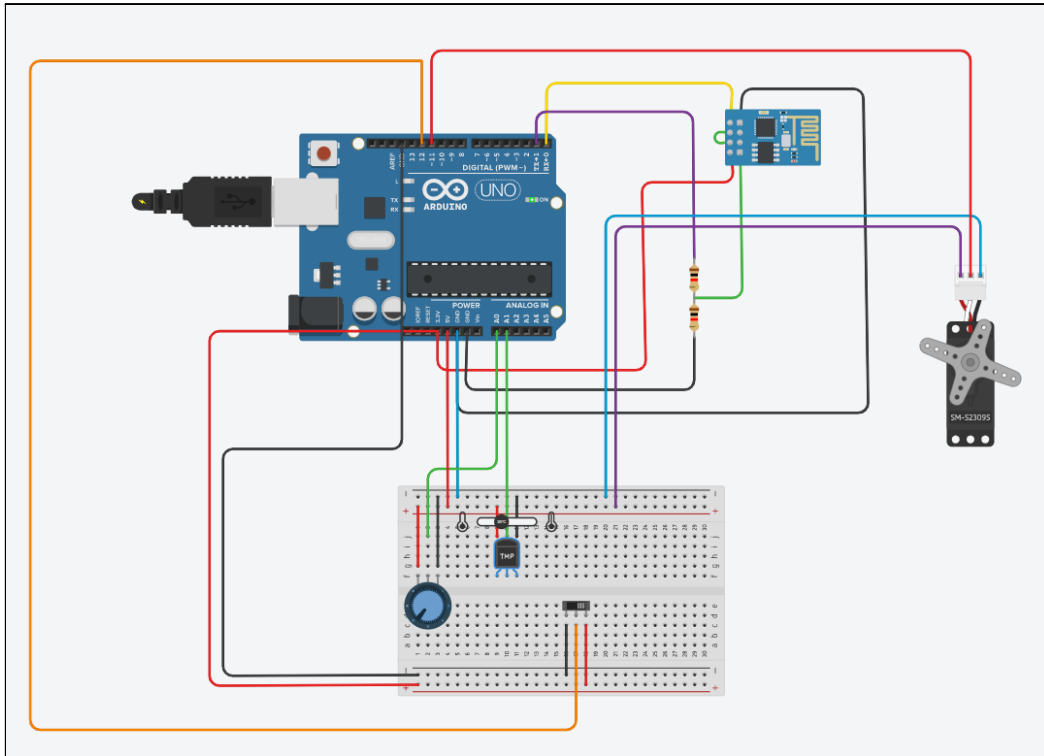
ESP8266 sends data to the server
593.00
0
25700593.00AT+CIPSEND=107
GET /update?api_key=XPT0V1HQ91F976H4&field1=0&field2=0&field3=593.00 HTTP/1.1
Host: api.thingspeak.com

593.00
0
25700593.00AT+CIPSEND=107
GET /update?api_key=XPT0V1HQ91F976H4&field1=0&field2=0&field3=593.00 HTTP/1.1
Host: api.thingspeak.com

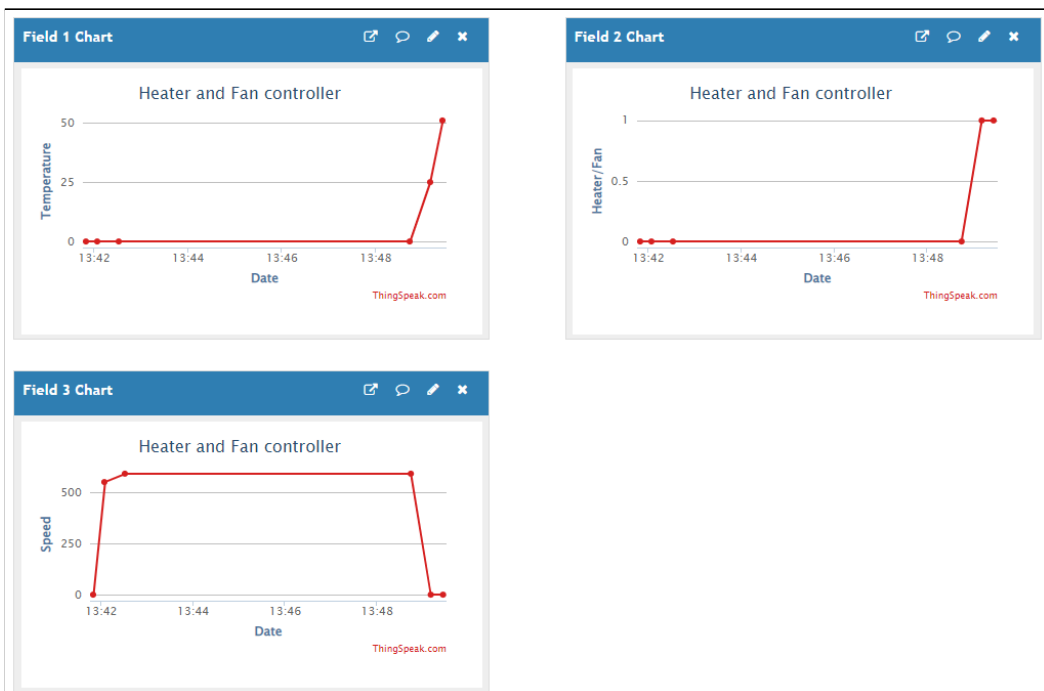
ESP8266 sends data to the server
593.00
0
25700593.00AT+CIPSEND=107
GET /update?api_key=XPT0V1HQ91F976H4&field1=0&field2=0&field3=593.00 HTTP/1.1
Host: api.thingspeak.com
  
```

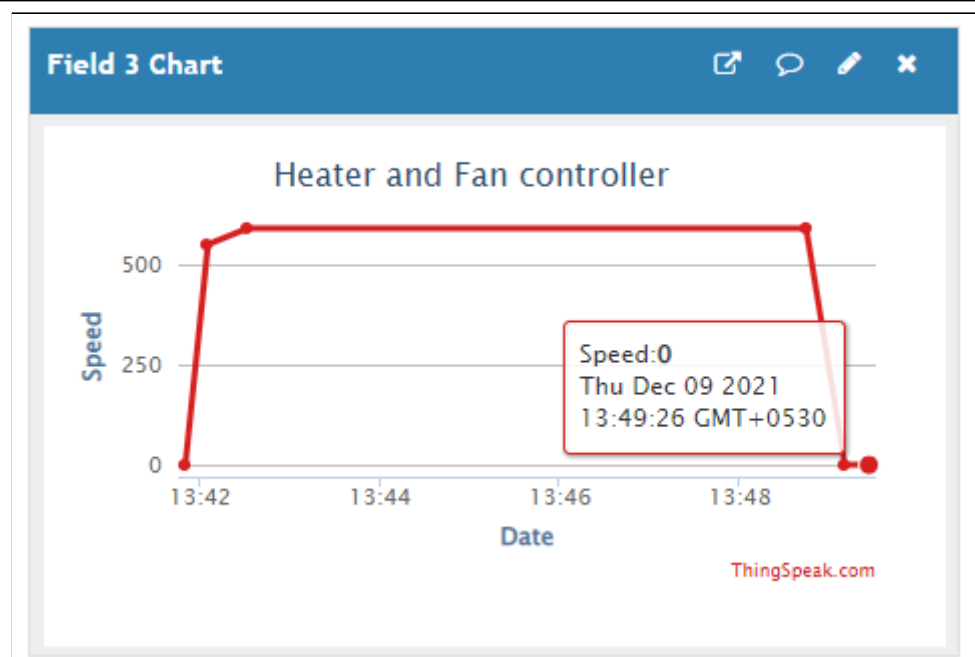
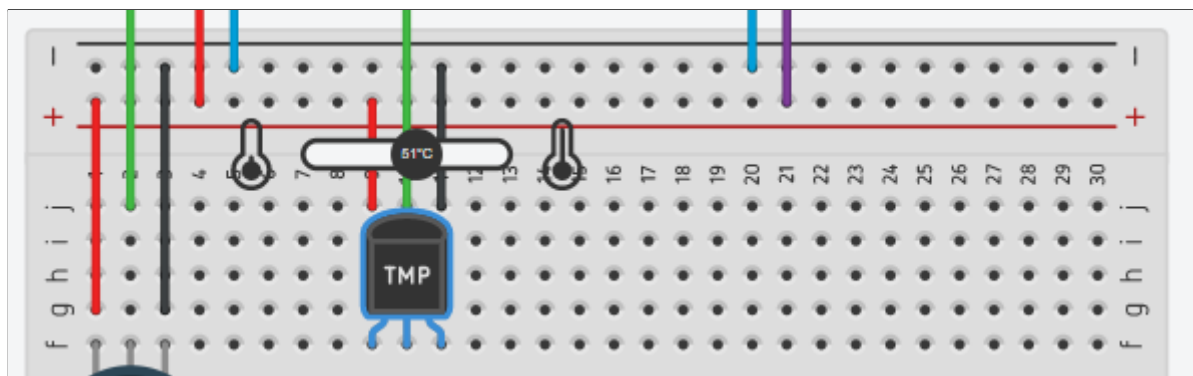
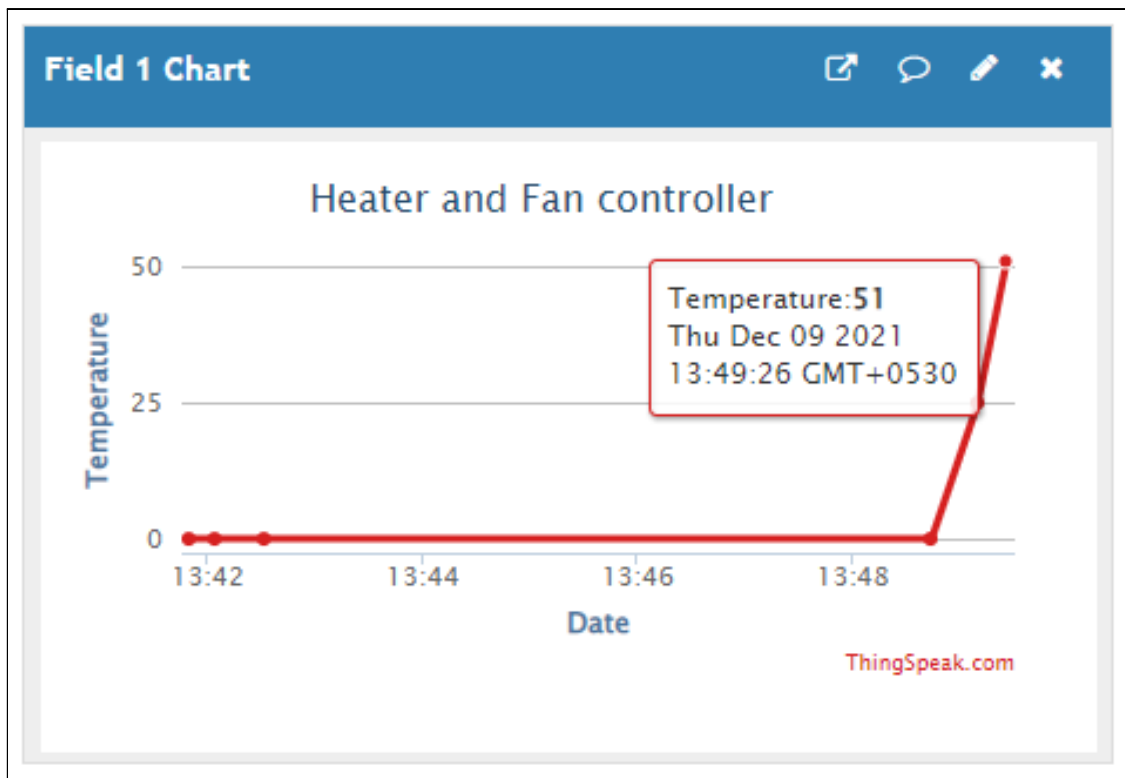
The Internet of Things (IoT)

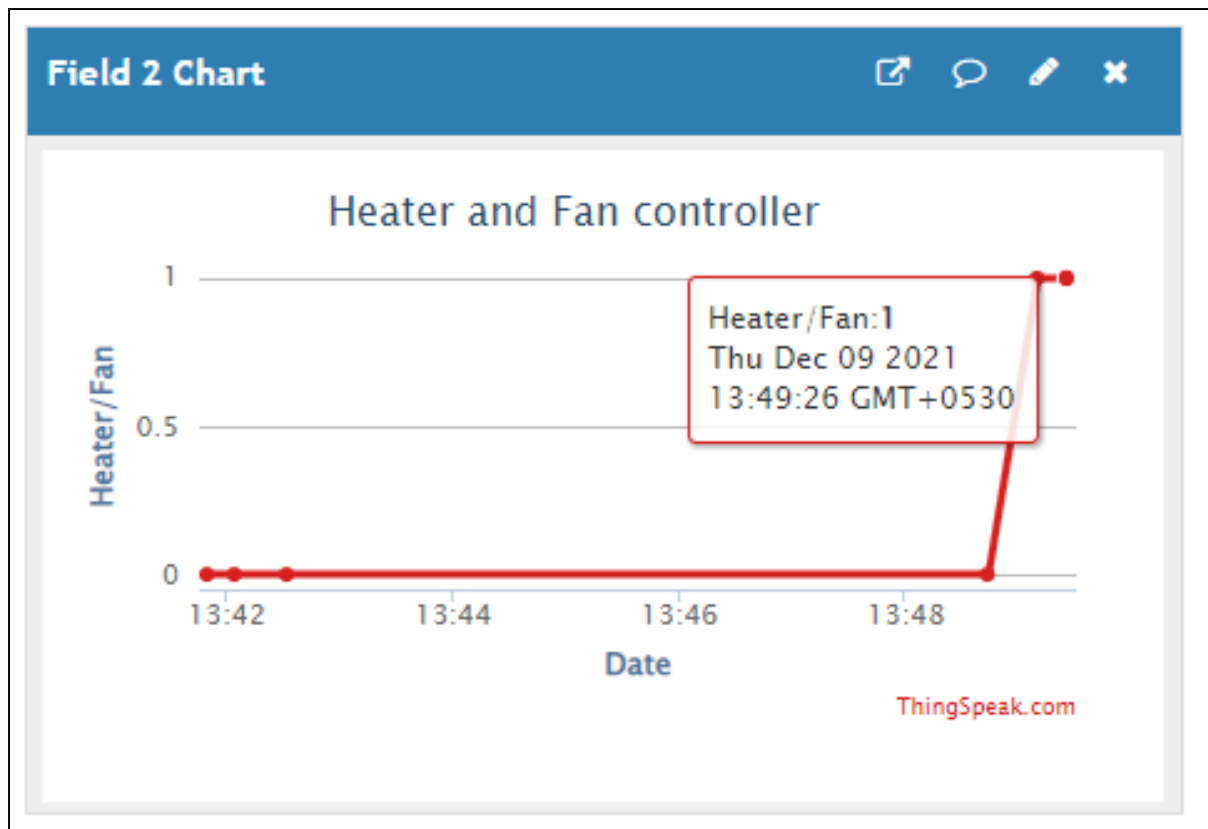
2. Manipulating the temperature (heater):



The temperature sensor that's present in the image above functions as a heater controller. The value is made to ascend. This data and the mode selected got stored on the cloud in **ThingSpeak**. The image below is the result.







From the first graph, it can be manifested that there's an in the value under mode one and a concomitant plummet in the speed graph. The first graph depicts (field 1 chart) the temperature. The second graph (field 2 chart) depicts if the mode selected is a heater or a fan. Then, the third (field 3 chart) plots the speed graph. Here, the temperature graph plots temperature variations on a heater. As the temperature was quite lower when compared to the speed, there's a plummet. And, once left with zero manipulation, the line graph moves linearly, manifesting a constant temperature. It can be deemed with no qualm that it was heater control. For, at 13 hours, 49 minutes and 26 seconds at GMT+0530, the value had ascended in the temperature and heater/fan graph accompanied by a plunge on the speed control graph.

These were the results on the serial monitor:

- P.T.O -



The image shows a 'Serial Monitor' window with a scrollable text area containing several lines of text. The text is organized into repeating blocks, each representing a sequence of events. Each block starts with an HTTP request line, followed by the host, then a timestamp, a line number, an AT command, another timestamp, another line number, another AT command, another timestamp, and finally another line number. The text ends with an incomplete line.

```
76H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

ESP8266 sends data to the server
51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

ESP8266 sends data to the server
51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.thingspeak.com

ESP8266 sends data to the server
51.00
1
2570151.00AT+CIPSEND=106
GET /update?api_key=XPT0V1HQ91F976H4&field1=51.00&field2=1&field3=0 HTTP/1.1
Host: api.th
```

At the bottom of the window, there is a text input field, a 'Send' button, a 'Clear' button, and a button with a waveform icon.
