**INTERNET OF THINGS**

**ACTIVITY - 6**

PREPARED BY

**Garv Baheti**

(2020BTechCSE031)

SUBMITTED TO

**Mr. Divanshu Jain**

Logo, company name

Description automatically generated

**Department of Computer Science Engineering**

**Institute of Engineering and Technology**

**JK Lakshmipat University Jaipur**

**November 07, 2022**

## OBJECTIVE

## Interface BME sensor with NodeMCU and upload the pressure, temperature and humidity data  using suitable dashboard.

**Hardware**

1. ESP8266 NodeMCU board
2. BME280 Sensor
3. Connecting Wires
4. Breadboard

**Wiring**

The connection of BME280 with the ESP8266 NodeMCU is very easy. We have to connect the VCC terminal with 3.3V, ground with the ground (common ground), SCL of the sensor with SCL of the module, and SDA of the sensor with the SDA pin of the ESP modules.

The I2C pin in ESP8266 NodeMCU for SDA is GPIO5 (D1) and for SCL is GPIO4 (D2).

**SETTING UP THINGSPEAK ACCOUNT FOR LM35 SENSOR:**

Before we proceed towards construction details we need to setup our thingspeak account correctly to receive the sensor data, this procedure need to be done for all three methods mentioned here.

You can sign up for *[thingspeak account here](https://thingspeak.com/users/sign_up)*, if you haven’t signed up yet.

* **You need to create a new channel** by clicking “New Channel” button and in the channel.

Ultra-Sonic Sensor Data to Thingspeak

1) Go to Channel settings tab and edit the name.

2) Enable Field 1 by checking the box and write the label as “Distance”.

3) Scroll down and click save.

* Now click on API keys tab:

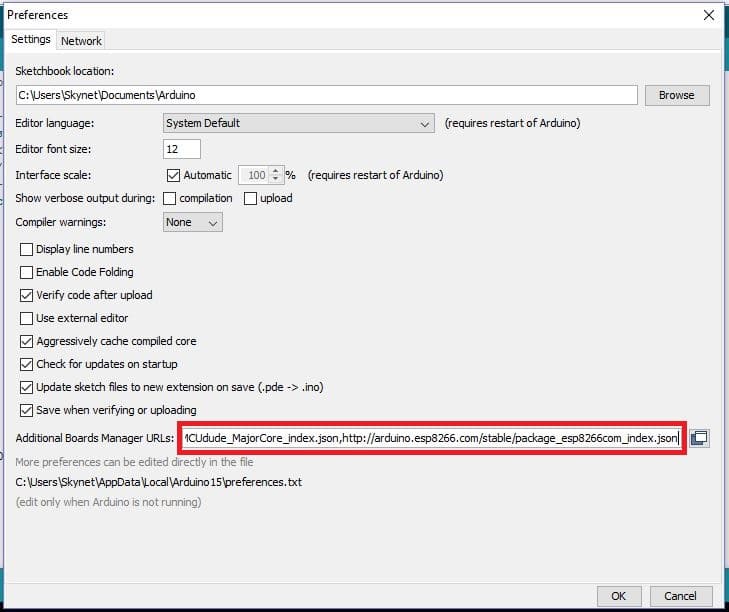
API keys are the access keys to your channel using which you can write and read values. In this project we are using only write API key which is already generated for your channel. You need to take note of it and this write API key need to be inserted to the given program codes.

* Now by clicking “Private View” tab you will see a blank channel. You will see some data once we send to it.

**Installing ESP8266 package to Arduino IDE:**

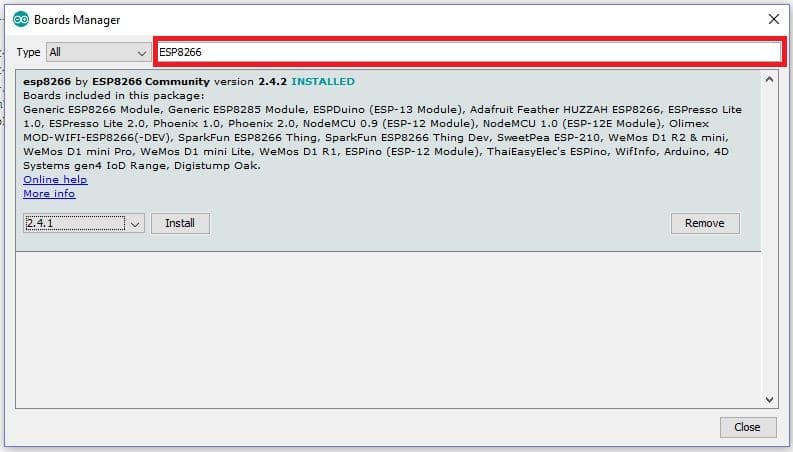
This process needs to be done if you are using NodeMCU or generic ESP8266 to connect to thingspeak. If you are going to use GSM modem to send LM35 data, this step is irrelevant for you, but **for NodeMCU and ESP8266 boards this step is mandatory.**

* In this step you need internet; we are going to install core files to Arduino IDE for IoT based boards.
* Copy this link: <http://arduino.esp8266.com/stable/package_esp8266com_index.json>
* Now open Arduino IDE and click on “**File**” >> “**Preferences**”.
* A window will open like this:



Preferences

* Paste the URL on the box and click “OK”.
* Now go to **tools > Board > Boards Manager**.
* Now a window will popup:



boards manager

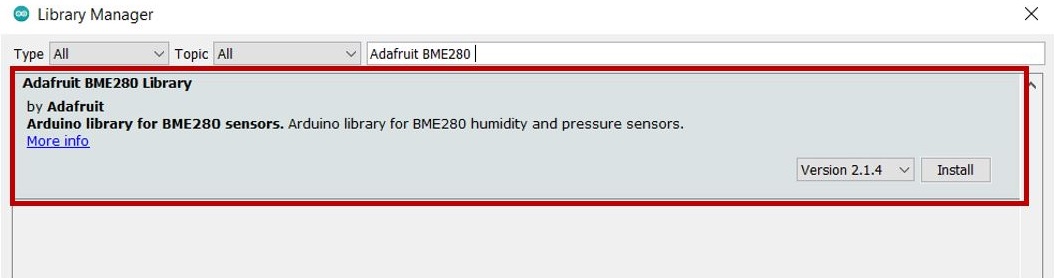
* Type ESP8266 on the box as shown and you will get an installation option, select the latest version and click install.

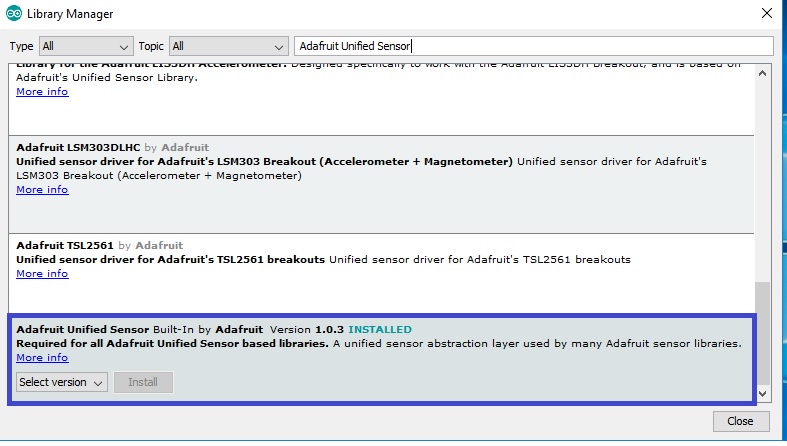
**Installing BME280 Libraries**

As we are connecting the BME280 sensor with ESP8266 NodeMCU so we will have to install BME280 libraries to our module. We will require two libraries for this project:

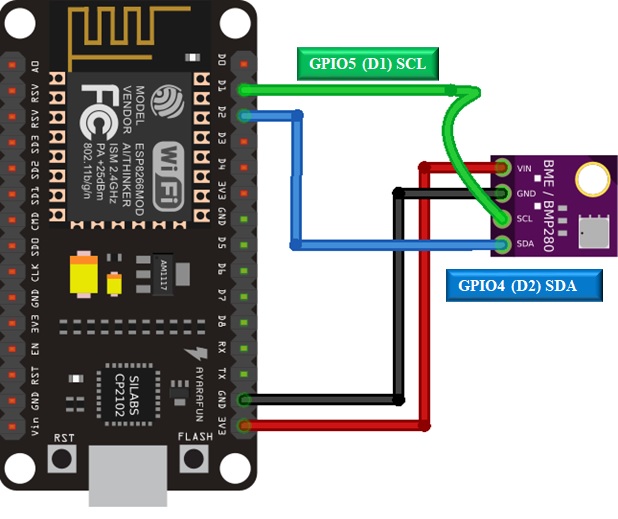
1. Adafruit\_BME280 library
2. Adafruit\_Sensor library

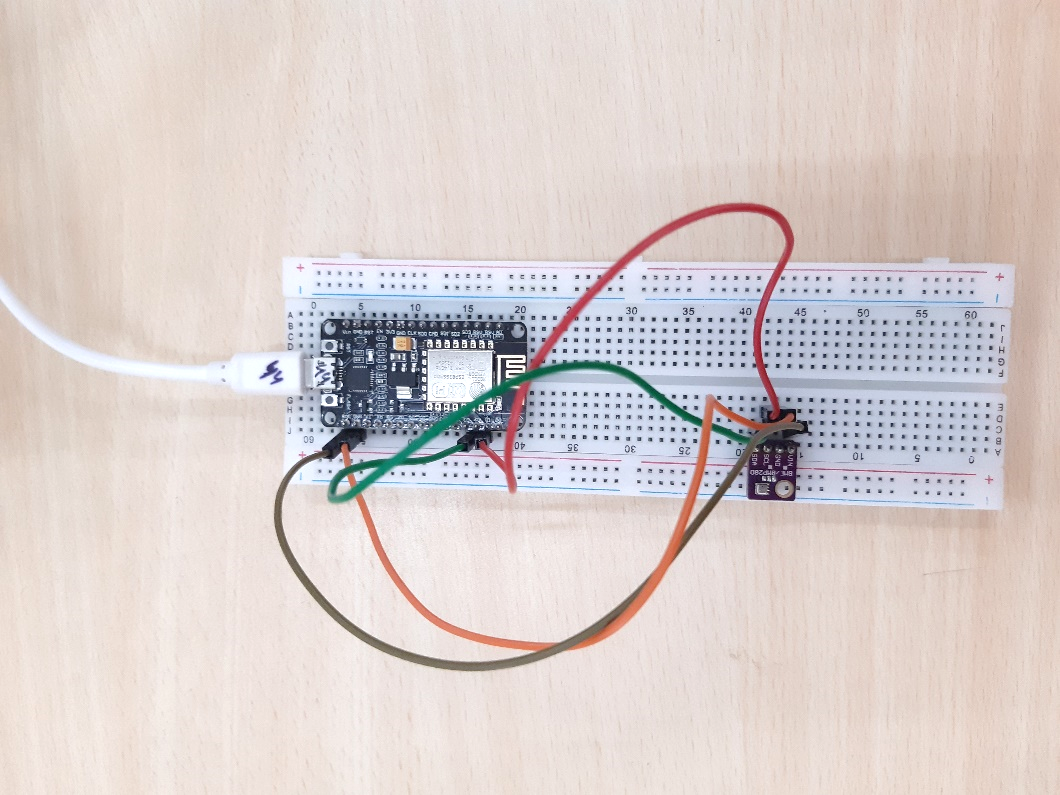
We will use the Library Manager in our Arduino IDE to install the latest versions of the libraries. Open your Arduino IDE and go to Sketch > Include Libraries > Manage Libraries. Type each library name in the search bar and install them both.

****

****

**CIRCUIT**





**CODE**

#include <ESP8266WiFi.h>

#include "ThingSpeak.h"

#include <Adafruit\_BME280.h>

#include <Adafruit\_Sensor.h>

const char\* ssid = "me.baheti.07";

const char\* password = "123456789";

WiFiClient  client;

unsigned long Channel\_ID = 1923409;

const char \* API\_Key = "4JHYJDEPWG9IGO3W";

unsigned long last\_time = 0;

unsigned long Delay = 30000;

*// Variables to store sensor readings*

float temperature;

float humidity;

float pressure;

*// Create a sensor object*

Adafruit\_BME280 bme;

void initBME(){

  if (!bme.begin(0x76)) {

    Serial.println("BME280 not properly connected!");

    while (1);

  }

}

void setup() {

  Serial.begin(115200);

  initBME();

  WiFi.mode(WIFI\_STA);

  ThingSpeak.begin(client);

}

void loop() {

  if ((millis() - last\_time) > Delay) {

*// Connect or reconnect to WiFi*

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

      Serial.print("Connecting...");

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

        WiFi.begin(ssid, password);

        delay(5000);

      }

      Serial.println("\nConnected.");

    }

*// Obtaining a new sensor reading for all fields*

    temperature = bme.readTemperature();

    Serial.print("Temperature (ºC): ");

    Serial.println(temperature);

    humidity = bme.readHumidity();

    Serial.print("Humidity (%): ");

    Serial.println(humidity);

    pressure = bme.readPressure() / 100.0F;

    Serial.print("Pressure (hPa): ");

    Serial.println(pressure);

    ThingSpeak.setField(1, temperature);

    ThingSpeak.setField(2, pressure);

    ThingSpeak.setField(3, humidity);

    int Data = ThingSpeak.writeFields(Channel\_ID, API\_Key);

    if(Data == 200){

      Serial.println("Channel updated successfully!");

    }

    else{

      Serial.println("Problem updating channel. HTTP error code " + String(Data));

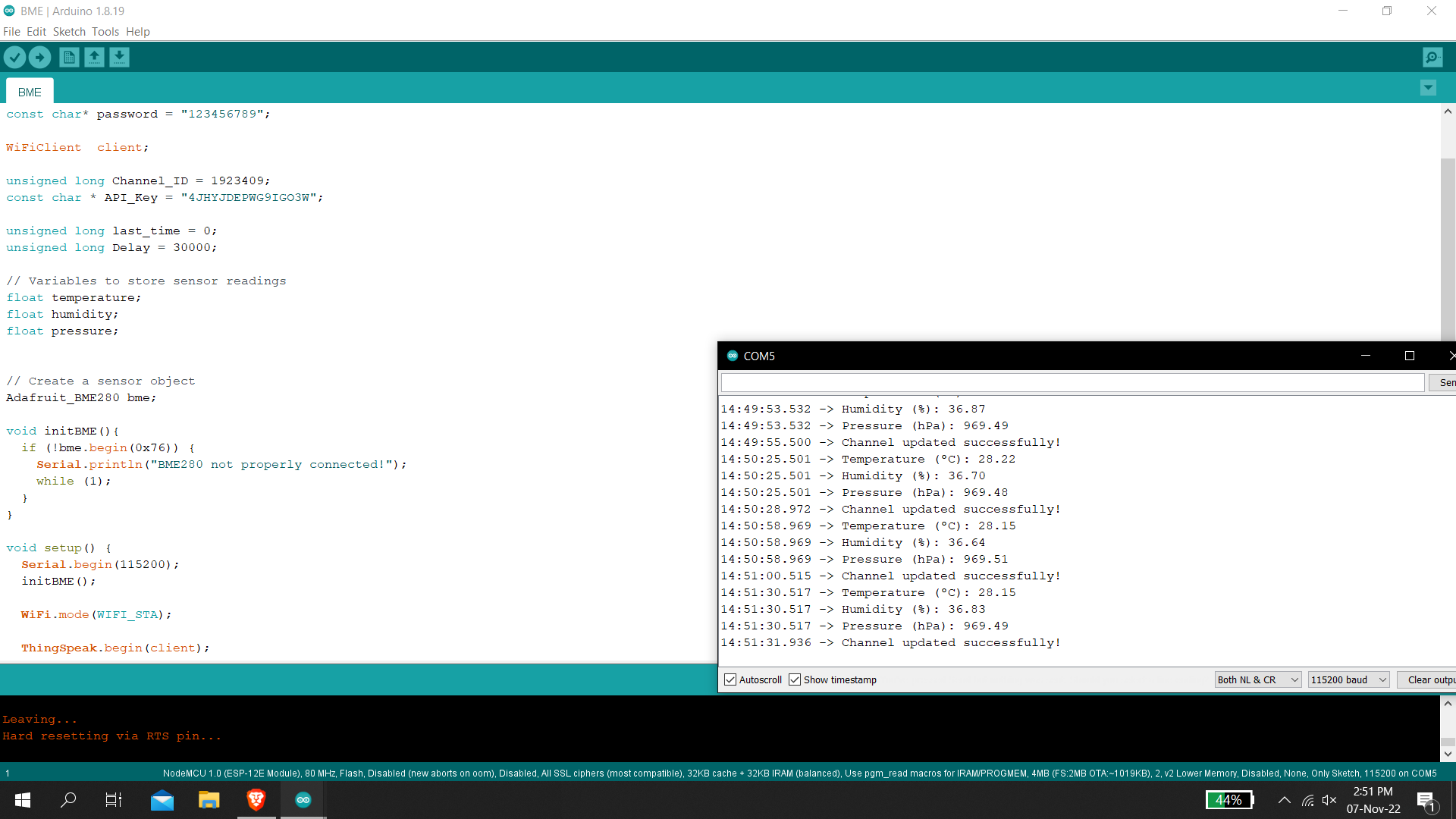
    }

    last\_time = millis();

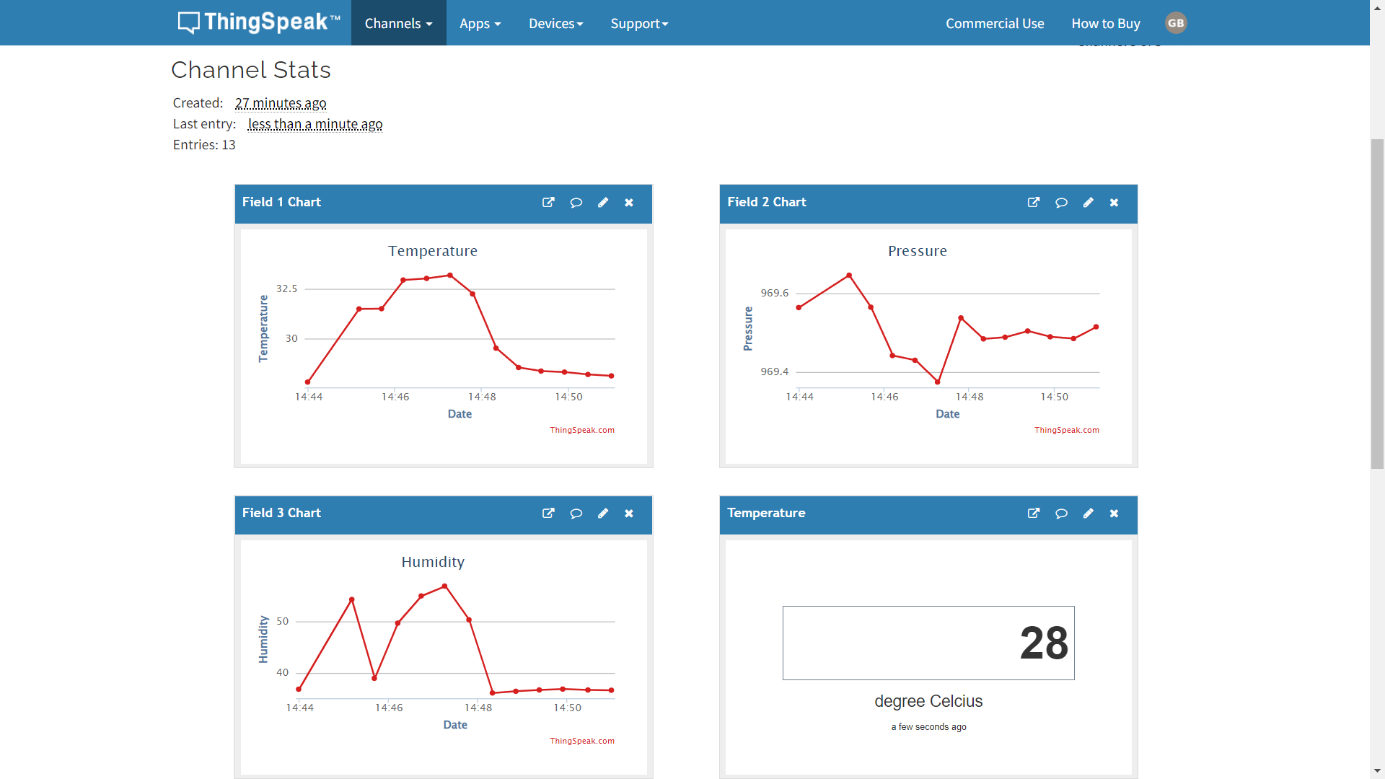
  }

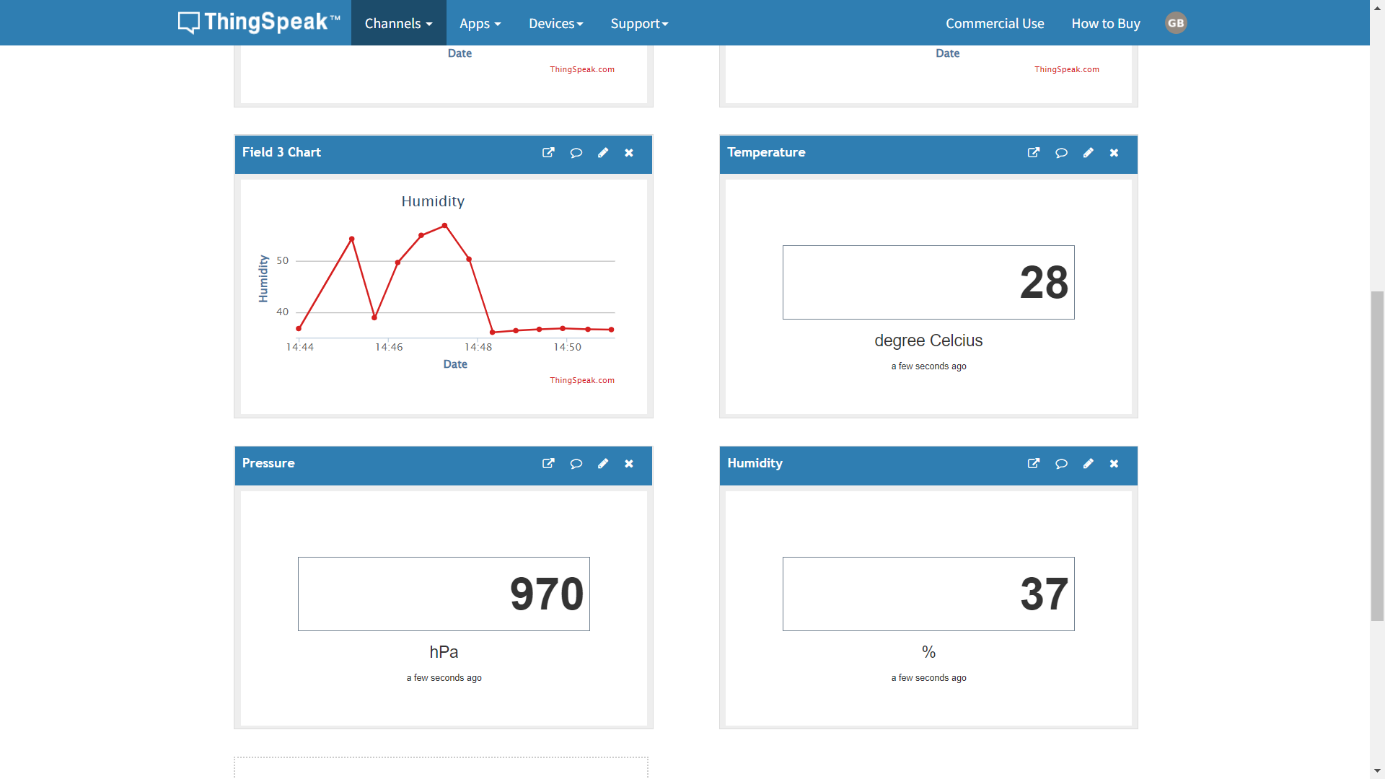
}

**OUTPUT**

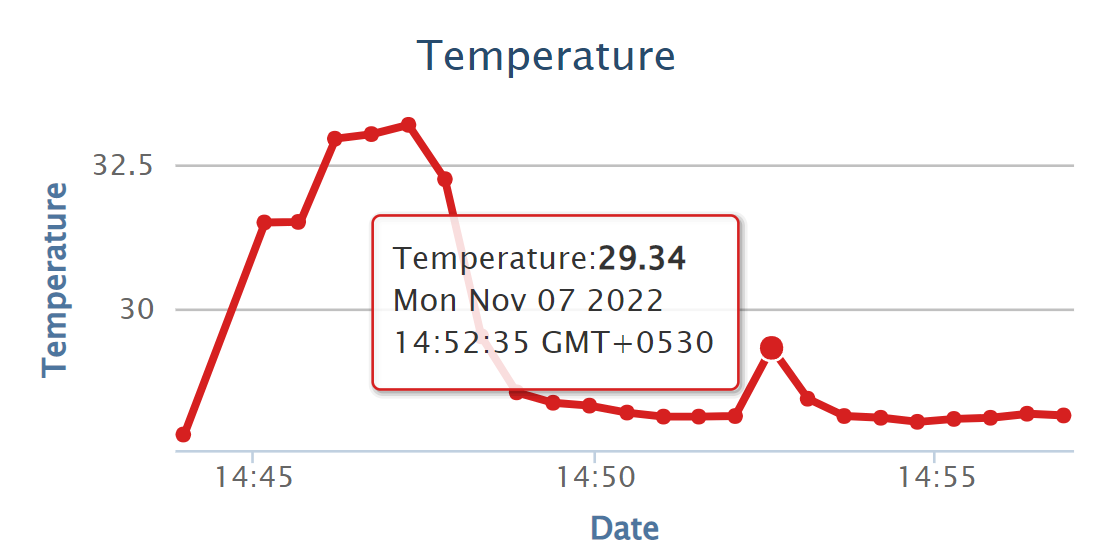


**THINGSPEAK DASHBOARD**





**GRAPHS**



Chart, line chart

Description automatically generated

Chart

Description automatically generated with medium confidence