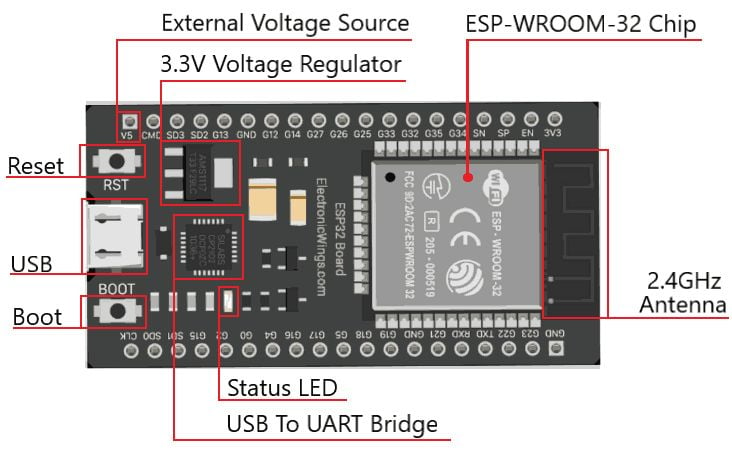
**TEMPERATURE MONITORING SETUP USING ESP32 IN WOKWI SIMULATOR AND ARDUINO CLOUD**

1. **ESP32**

The ESP32 is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth. It's a system-on-a-chip (SoC), which means it's an integrated circuit that contains an entire electronic system.

What it's used for Mobile devices, Wearable electronics, Internet of Things (IoT) applications, and Industrial environments.



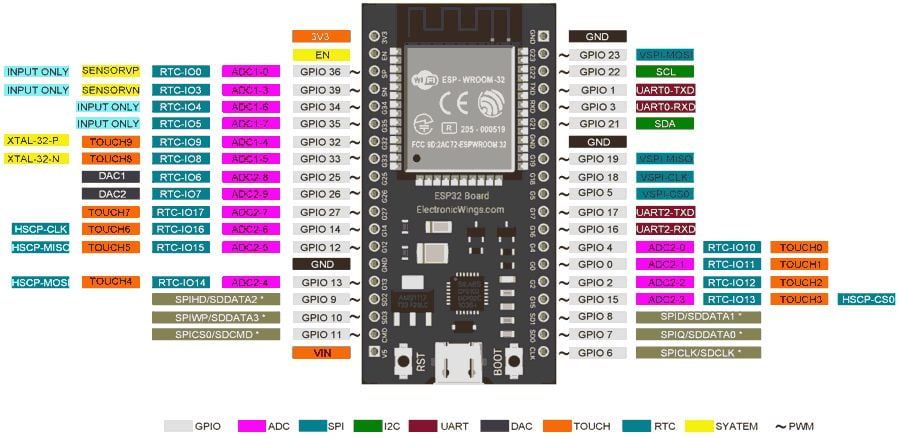
**Features:**

* **Wi-Fi**: Supports 802.11b/g/n standard at 2.4 GHz band, up to 150 Mbps
* **Bluetooth**: Supports Bluetooth v4.2 BR/EDR and Bluetooth LE specifications
* **Power consumption**: Ultra-low power consumption
* **Operating temperature**: Can function reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C
* **Digital pins**: 34 digital pins that can be used to connect sensors, buttons, buzzers, and more

Espressif Systems, a Chinese company based in Shanghai

Manufactured by TSMC using their 40 nm process

**PIN DIAGRAM**



1. **WOKWI**

Wokwi is an online Electronics simulator. Used it to simulate Arduino, ESP32, STM32, and many other popular boards, parts and sensors.

**Features**[**​**](https://docs.wokwi.com/?utm_source=wokwi#unique-features)**:**

* [**WiFi simulation**](https://docs.wokwi.com/guides/esp32-wifi) - Connect your simulated project to the internet. You can use MQTT, HTTP, NTP, and many other network protocols.
* [**Virtual Logic Analyzer**](https://docs.wokwi.com/guides/logic-analyzer) - Capture digital signals in your simulation (e.g. UART, I2C, SPI) and analyze them on your computer.
* [**Advanced debugging with GDB**](https://docs.wokwi.com/gdb-debugging) - Powerful Arduino and Raspberry Pi Pico debugger for advanced users.
* [**SD card simulation**](https://docs.wokwi.com/parts/wokwi-microsd-card) - Store and retrieve files and directories from your code. [Paying users](https://wokwi.com/pricing?ref=docs_sdcard) can also upload binary files (such as images)
* [**Chips API**](https://docs.wokwi.com/chips-api/getting-started) - Create your own custom chips and parts, and share them with the community.
* [**Visual Studio Code integration**](https://docs.wokwi.com/vscode/getting-started) - Simulate your embedded projects directly from VS Code.

1. **ARDUINO CLOUD**

The [Arduino Cloud](https://app.arduino.cc/) is a platform for developing Arduino projects and connecting them to the world. It supports secure connections with boards via [Wi-Fi®](https://docs.arduino.cc/arduino-cloud/hardware/wifi), [LoRa®](https://docs.arduino.cc/arduino-cloud/hardware/lora), [Ethernet](https://docs.arduino.cc/arduino-cloud/hardware/ethernet) and [Cellular (GSM/NB-IoT)](https://docs.arduino.cc/arduino-cloud/hardware/cellular), and lets you create a system for sending any variable information you can think of from one board to another within minutes of unboxing them. The Arduino Cloud platform includes:

* an **Integrated Development Environment (IDE)** for programming your boards,
* a **cloud backend service** for synchronizing data from Arduino boards, but also from [Python](https://docs.arduino.cc/arduino-cloud/guides/python) & [JavaScript](https://docs.arduino.cc/arduino-cloud/guides/javascript) clients,
* a **graphical tool (dashboard)** for controlling and monitoring your board (as well as an [mobile app](https://docs.arduino.cc/arduino-cloud/iot-remote-app/getting-started)).
* [REST API](https://docs.arduino.cc/arduino-cloud/api/arduino-iot-api) and [command line tools](https://docs.arduino.cc/arduino-cloud/arduino-cloud-cli/getting-started) for larger scale automations.

Very simply explained, with the Arduino Cloud you can:

1. Create a program for an Arduino based on a brilliant idea you just hatched.
2. Upload the program to your board and synchronize any data you want to (most commonly through Wi-Fi®).
3. Create a dashboard with a set of widgets to control and monitor your data.

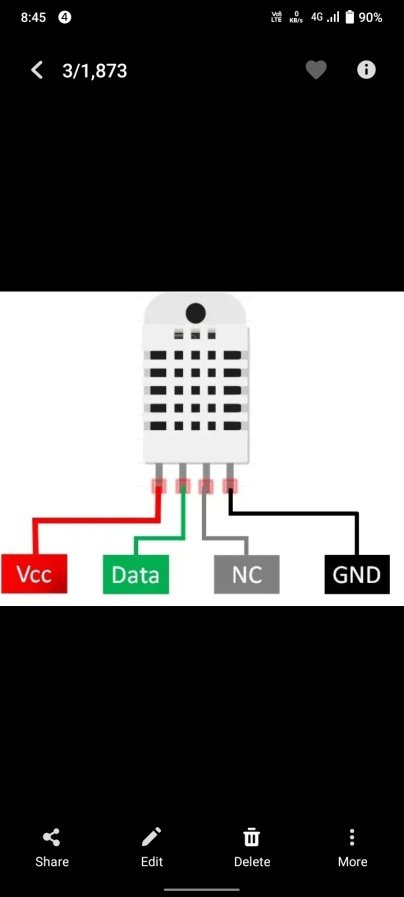
Developing a sketch made for the Arduino Cloud is almost just like developing any other sketch. You have your setup, and loop just like normal. But you are also able to select variables to be automatically synced to the Arduino Cloud. Once a variable is synced to the Arduino Cloud you can access it from any of your other devices connected to the Arduino Cloud. Once you have setup your project, this is how you could interact & monitor it. Anything in the Arduino Cloud can be accessed via the left action bar. In the list below, some with links to a documentation pages.

* [**Sketches**](https://docs.arduino.cc/arduino-cloud/cloud-interface/sketches) - your sketches (programs) are stored here, divided into either a "normal" or "cloud" sketch.
* [**Devices**](https://docs.arduino.cc/arduino-cloud/hardware/devices) - here you can configure your Arduino boards, ESP32 devices, manual devices (Python, JavaScript and more).
* [**Things**](https://docs.arduino.cc/arduino-cloud/cloud-interface/things) - a Thing is a project configuration, where you select device, create variables to synchronize, and enter credentials.
* [**Dashboards**](https://docs.arduino.cc/arduino-cloud/cloud-interface/dashboard-widgets) - dashboards are used to monitor & control your board through widgets. There's also a [mobile version](https://docs.arduino.cc/arduino-cloud/iot-remote-app/getting-started) available to view your dashboards from a smartphone.
* [**Triggers**](https://docs.arduino.cc/arduino-cloud/cloud-interface/triggers) - triggers can be used to send emails and push notifications based on a value change of a variable.
* **Resources** - helpful links and resources. You might have found this article here.
* **Courses** - tailored content for specific products and kits connected to the Arduino Cloud.
* **Templates** - templates are ready-made project that will automatically configure your device, Thing,
* **Integrations** - third party services that are integrated with the Arduino Cloud.

**Features:**

* [**Data Monitoring**](https://docs.arduino.cc/arduino-cloud/application-notes/cloud-environmental-data) - learn how to easily monitor environmental sensor values through a dashboard.
* [**Variable Synchronisation**](https://docs.arduino.cc/arduino-cloud/features/thing-to-thing) - variable synchronisation allows you to sync variables across devices, enabling communication between devices with minimal coding.
* [**Scheduler**](https://docs.arduino.cc/arduino-cloud/features/cloud-scheduler) - schedule jobs to go on/off for a specific amount of time (seconds, minutes, hours).
* [**Over-The-Air (OTA) Uploads**](https://docs.arduino.cc/arduino-cloud/features/ota-getting-started) - upload code to devices not connected to your computer.
* [**Webhooks**](https://docs.arduino.cc/arduino-cloud/features/webhooks) - integrate your project with another service, such as IFTTT.
* [**Amazon Alexa Support**](https://docs.arduino.cc/arduino-cloud/guides/alexa) - make your project voice controlled with the Amazon Alexa integration.
* [**Dashboard Sharing**](https://docs.arduino.cc/arduino-cloud/features/sharing-dashboards) - share your data with other people around the world.

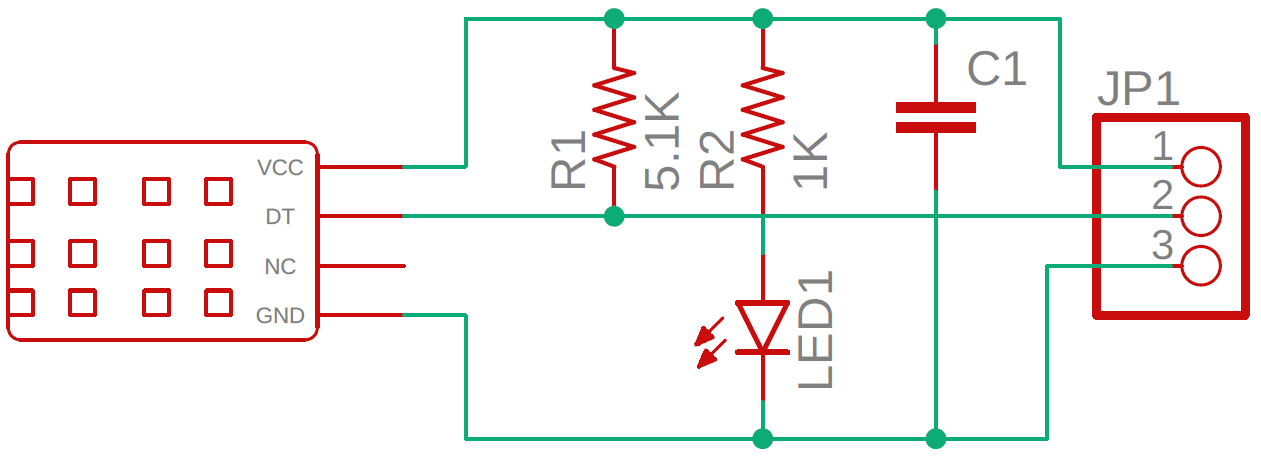
1. **DHT 22 TEMPERATURE SENSOR**

****

****

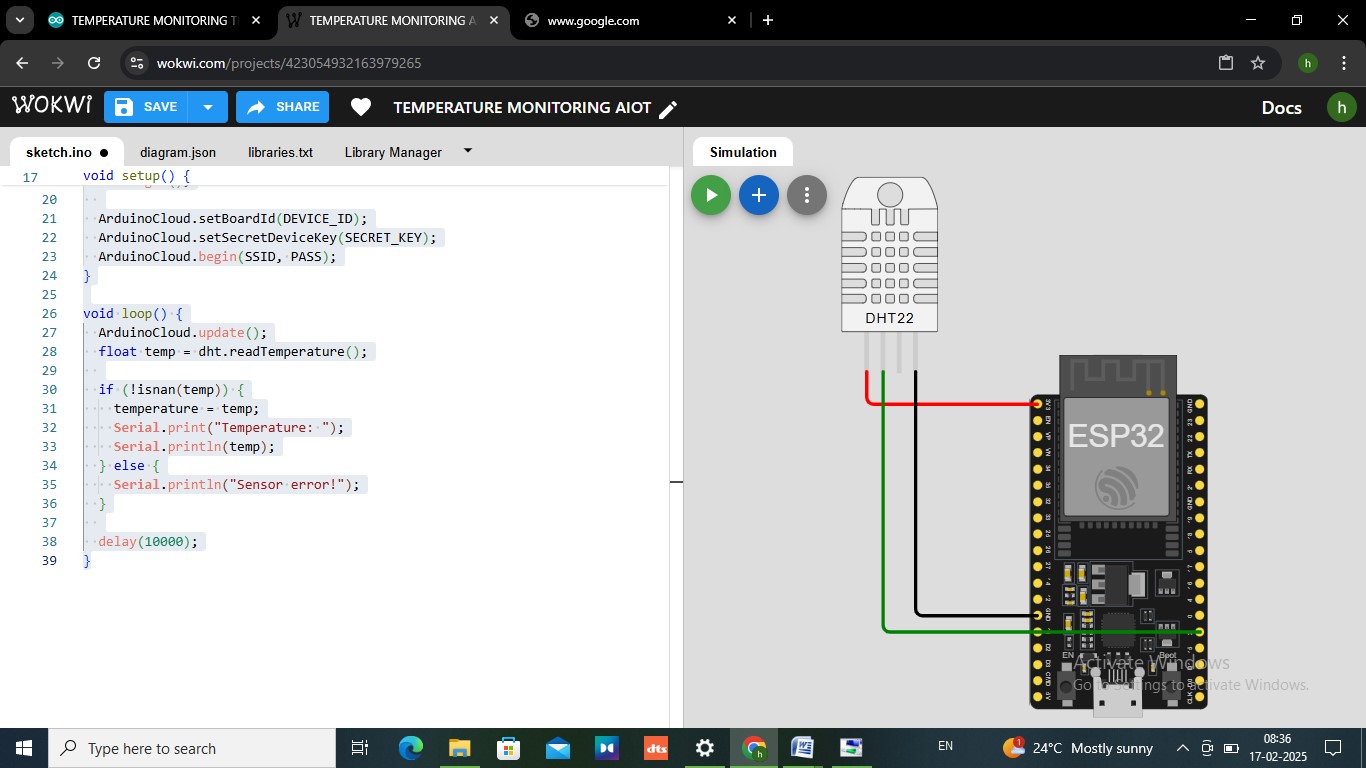
The DHT22 is a low-cost digital temperature and humidity sensor with a single wire digital interface. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).The sensor is calibrated and doesn’t require extra components so you can get the right to measuring relative humidity and temperature.

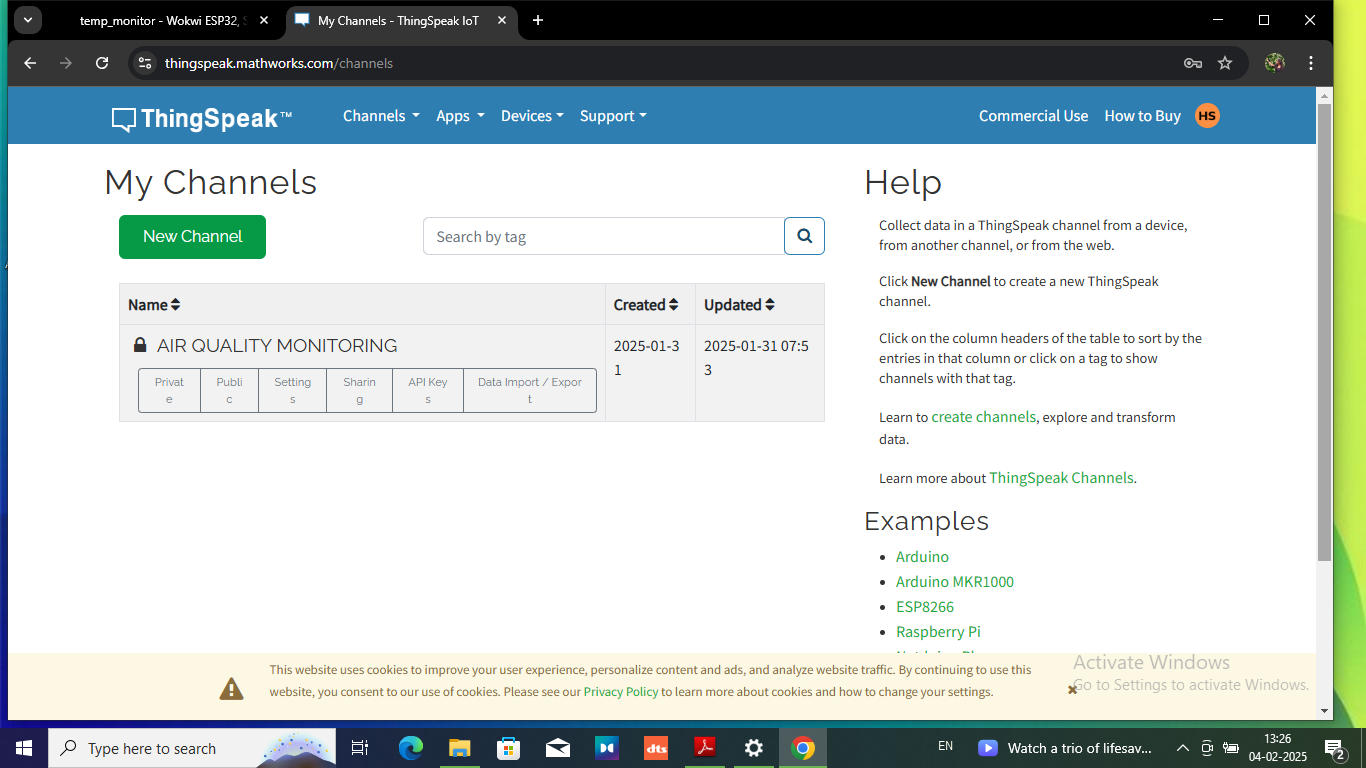
It is quite simple to use but requires careful timing to grab data. You can only get new data from it once every 2 seconds.

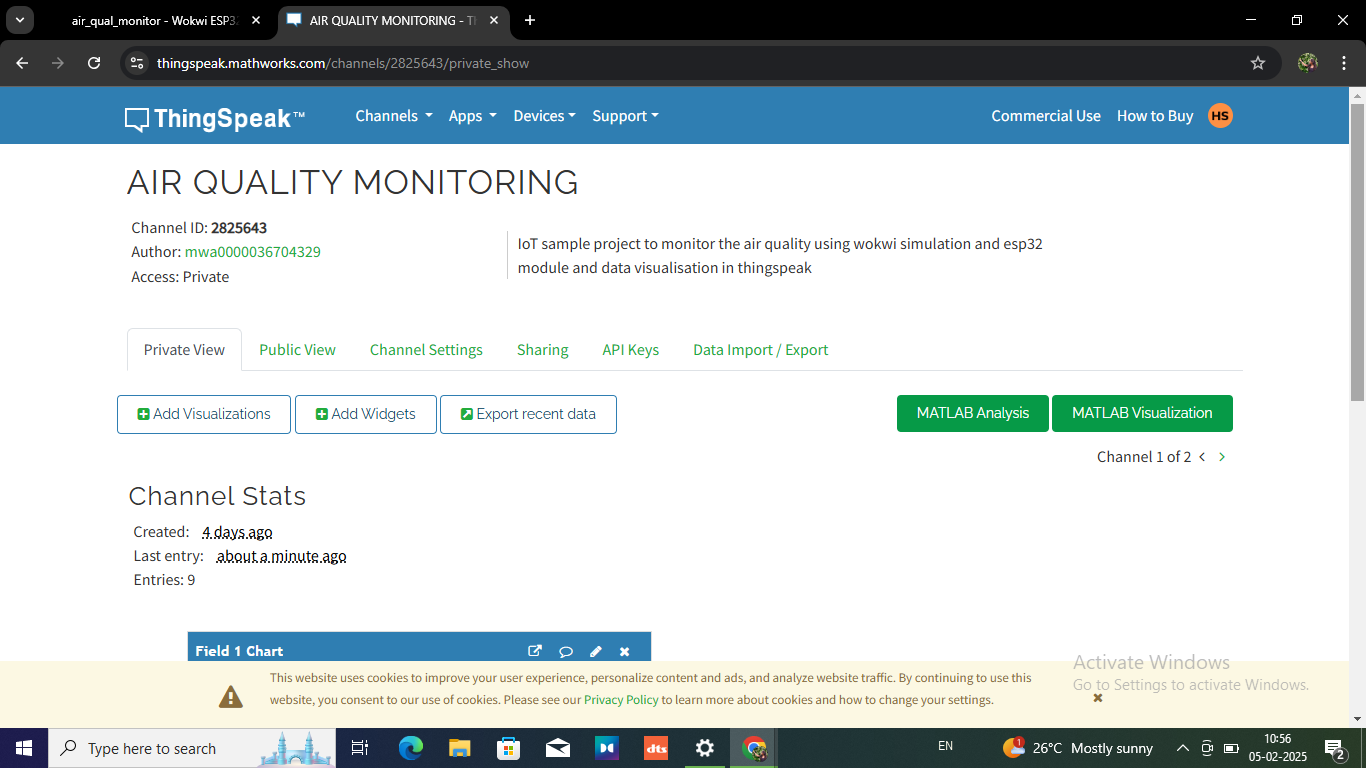


**Features:**

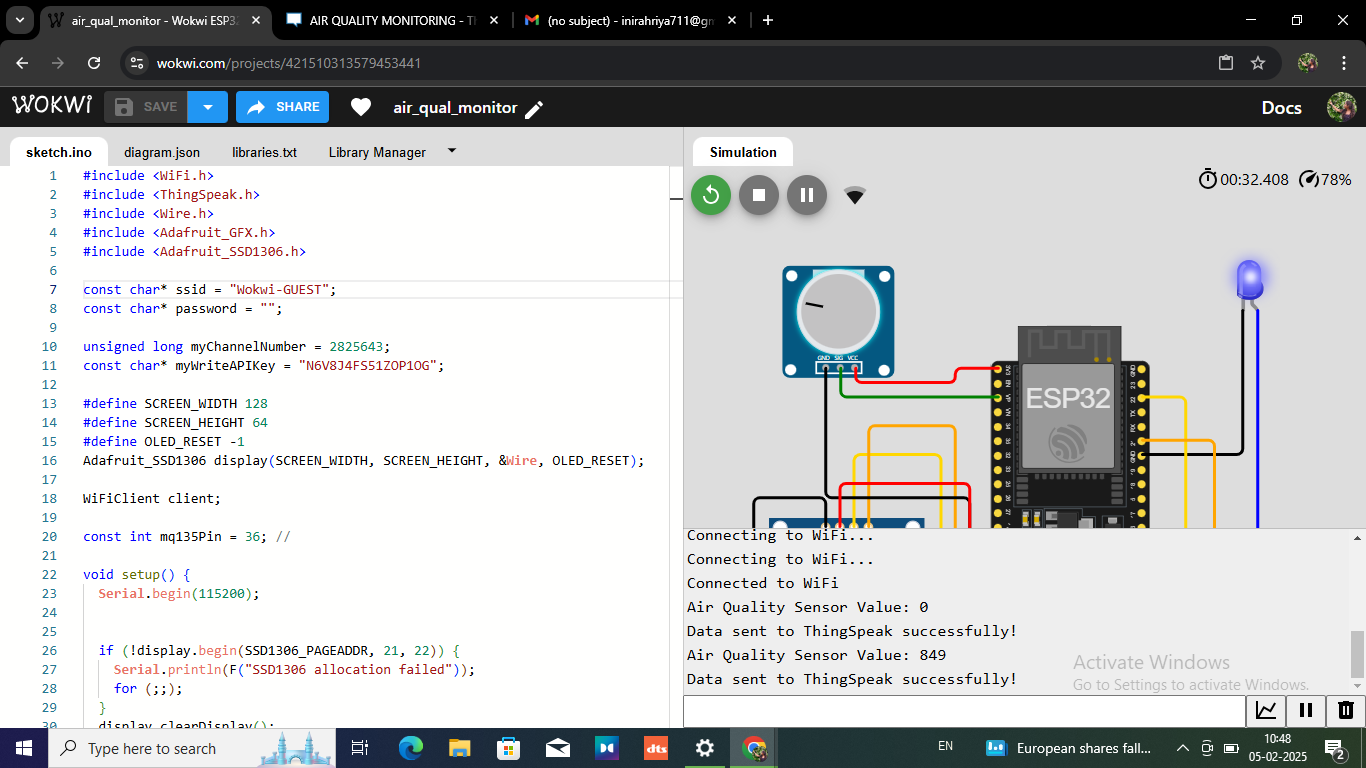
1. \*High precision
2. \*Capacitive type
3. \*Full range temperature compensated
4. \*Relative humidity and temperature measurement
5. \*Calibrated digital signal
6. "Outstanding long-term stability
7. \*Extra components not needed
8. \*Long transmission distance, up to 100 meters
9. "Low power consumption
10. \*4 pins packaged and fully interchangeable
11. **SKETCH CODE**
12. #include <ArduinoIoTCloud.h>
13. #include <Arduino\_ConnectionHandler.h>
14. #include <DHT.h>
15. #define DHTPIN 2
16. #define DHTTYPE DHT22
17. DHT dht(DHTPIN, DHTTYPE);
18. const char DEVICE\_ID[] = "ab05accd-427a-4158-884d-f21106735cb9";
19. const char SSID[] = "Wokwi-GUEST";
20. const char PASS[] = "";
21. const char SECRET\_KEY[] = "q9VIOk4pqCJWB9r@mMO57Ot!e";
22. CloudTemperatureSensor temperature;
23. void setup() {
24. **Serial**.begin(9600);
25. dht.begin();
27. ArduinoCloud.setBoardId(DEVICE\_ID);
28. ArduinoCloud.setSecretDeviceKey(SECRET\_KEY);
29. ArduinoCloud.begin(SSID, PASS);
30. }
31. void loop() {
32. ArduinoCloud.update();
33. float temp = dht.readTemperature();
35. if (!isnan(temp)) {
36. temperature = temp;
37. **Serial**.print("Temperature: ");
38. **Serial**.println(temp);
39. } else {
40. **Serial**.println("Sensor error!");
41. }
43. delay(10000);
44. }
    1. **SETUP**

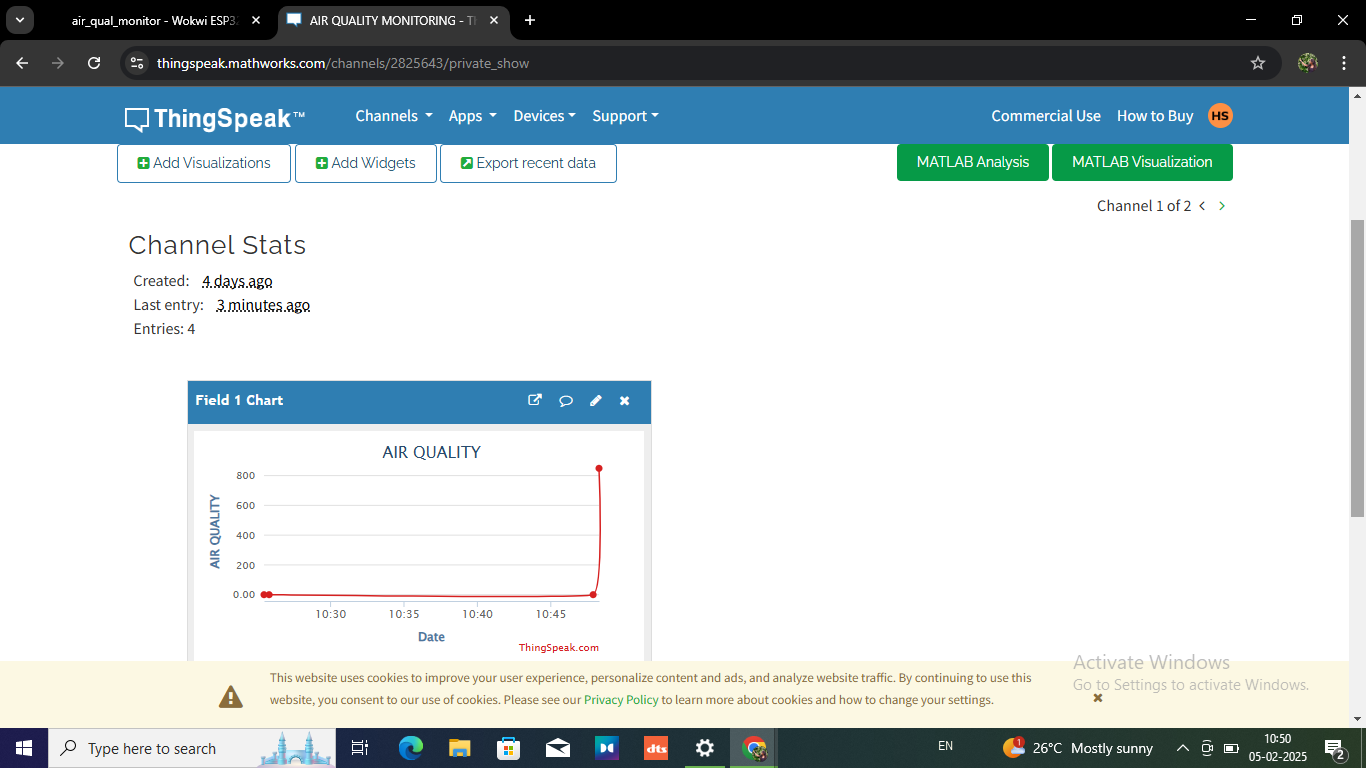
****

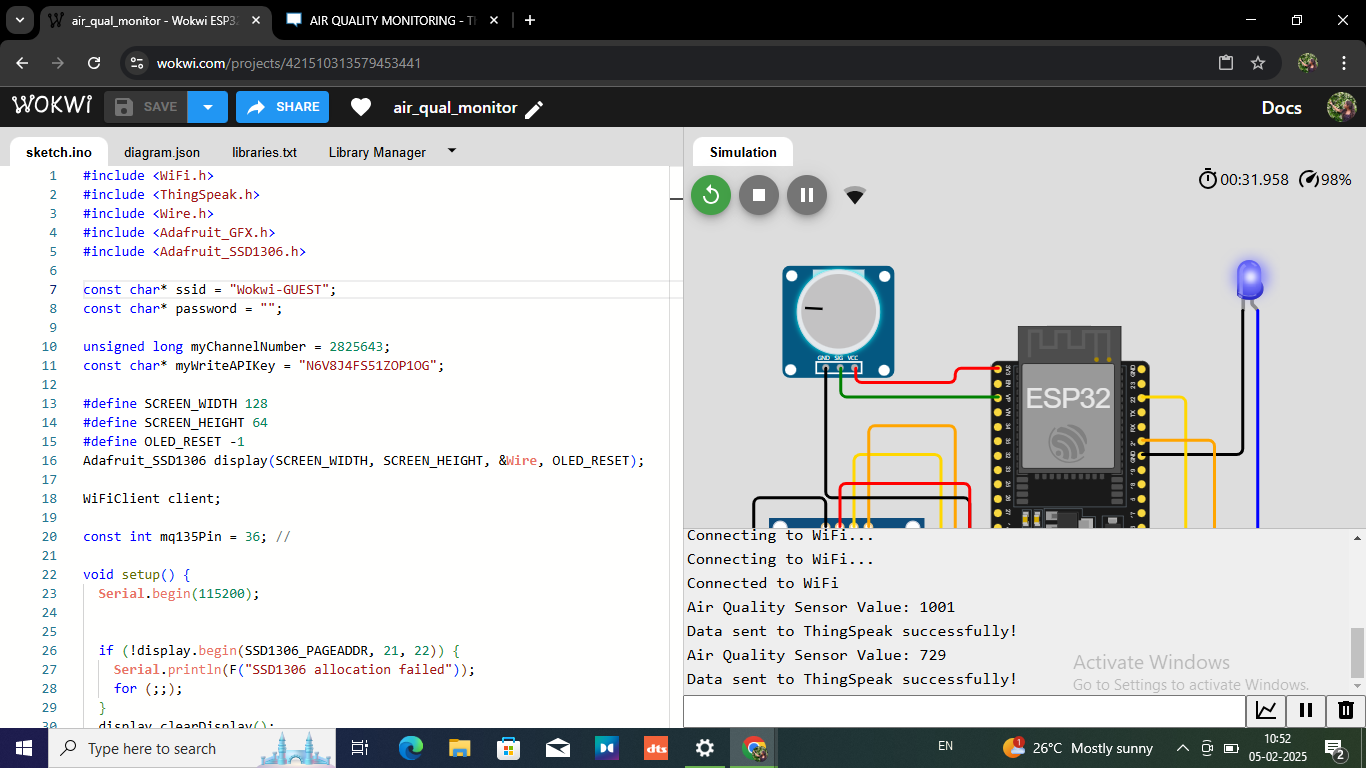
****

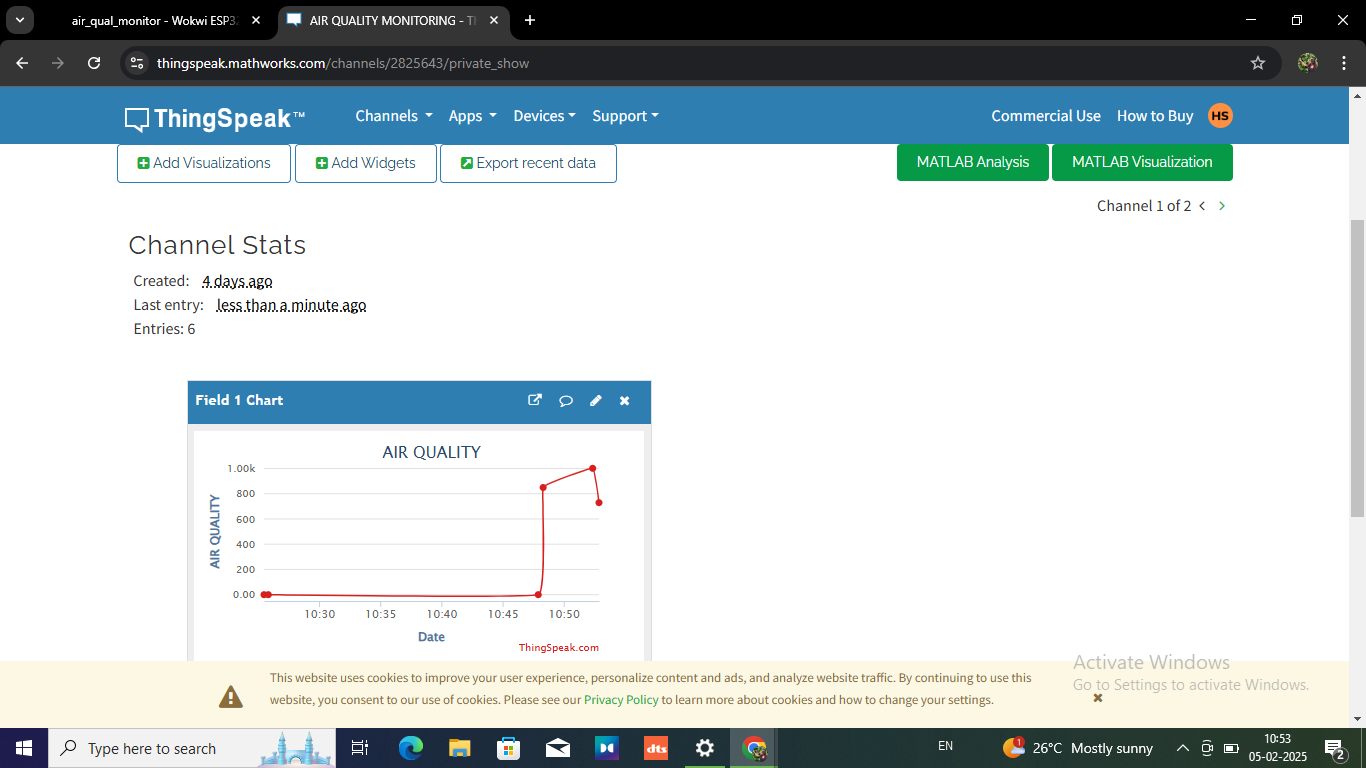
****

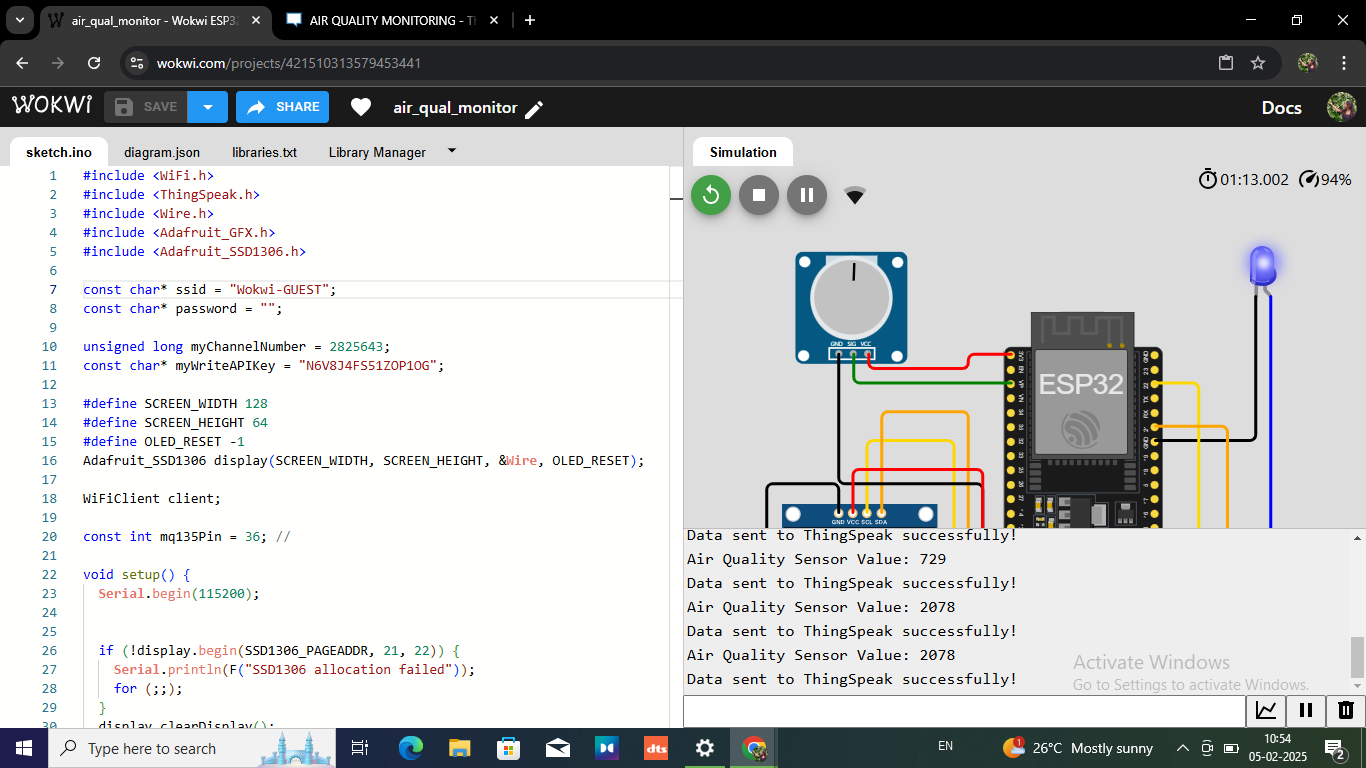
1. **OUTPUT**

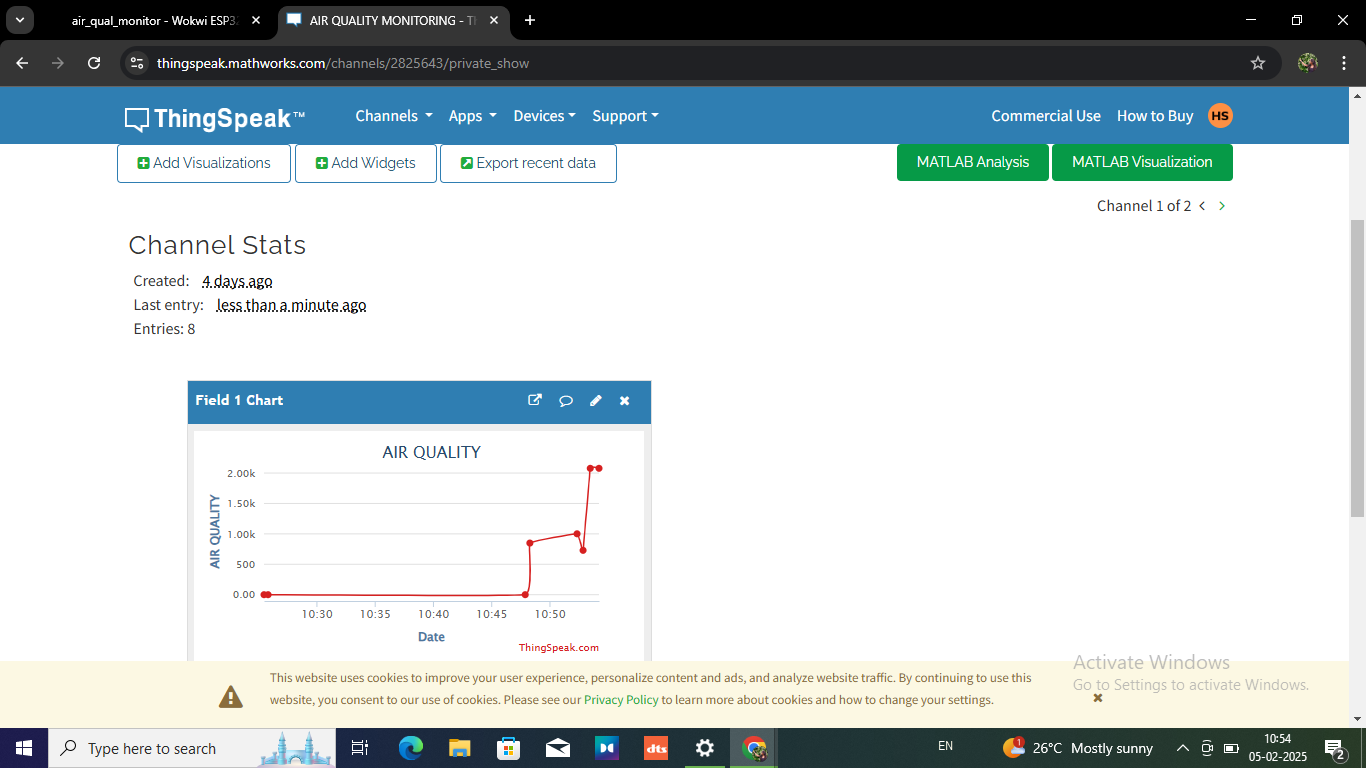


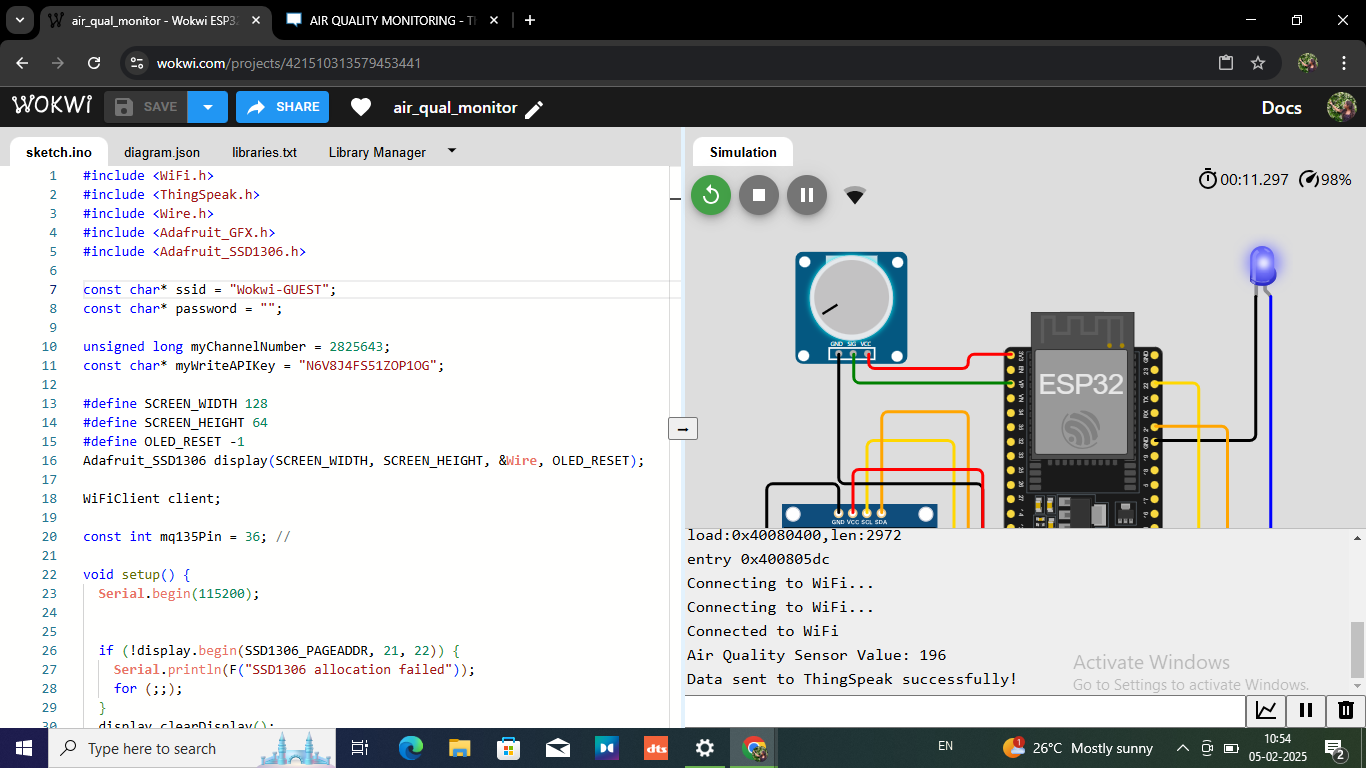
****

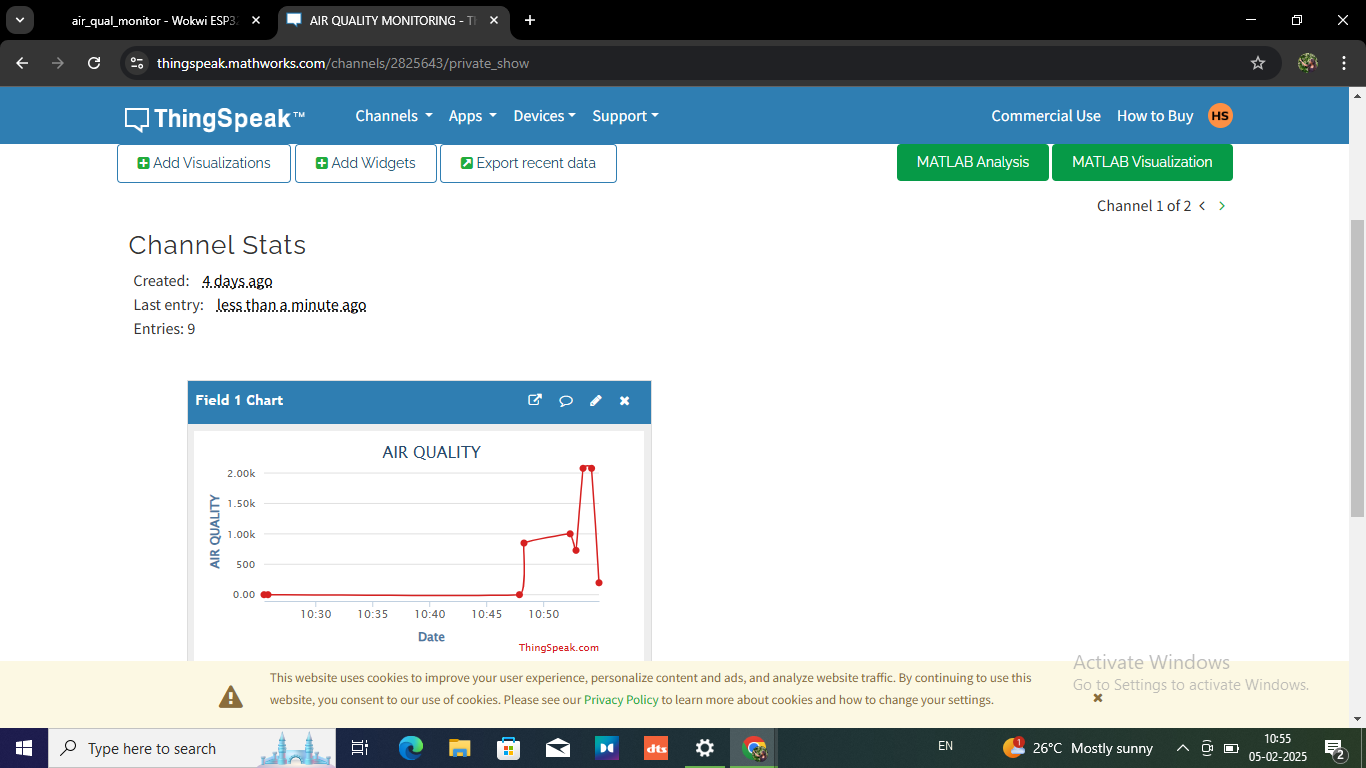
****

****

****

****

****

****