SMART FAN WITH AUTOMATED TEMPERATURE CONTROL

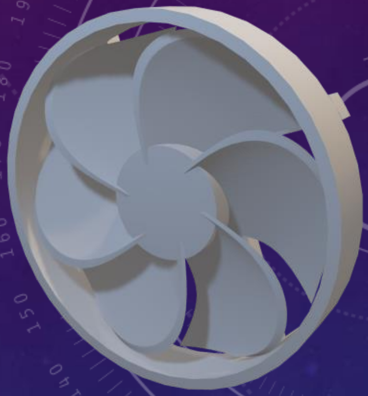
AS THE LAB OF IOT PROJECT

ZAKARYA BOUDRAF

INTRODUCTION

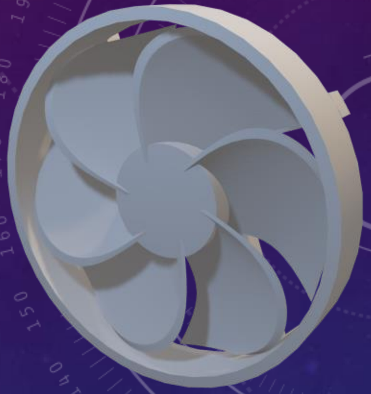
- **Concept of a Smart Fan:** This IoT project focuses on building a smart fan that automatically adjusts its speed based on the environment's temperature.

- This helps achieve an **energy-efficient** and **optimized** cooling solution for a wide range of purposes.

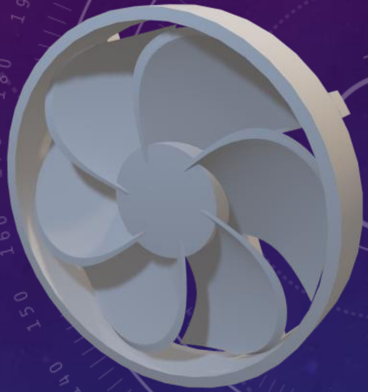


PROJECT OVERVIEW

- **Goals:**
  - Develop a fan that automatically adjusts its speed based on the ambient temperature.
  - Integrate the fan with a smart home automation system for monitoring.



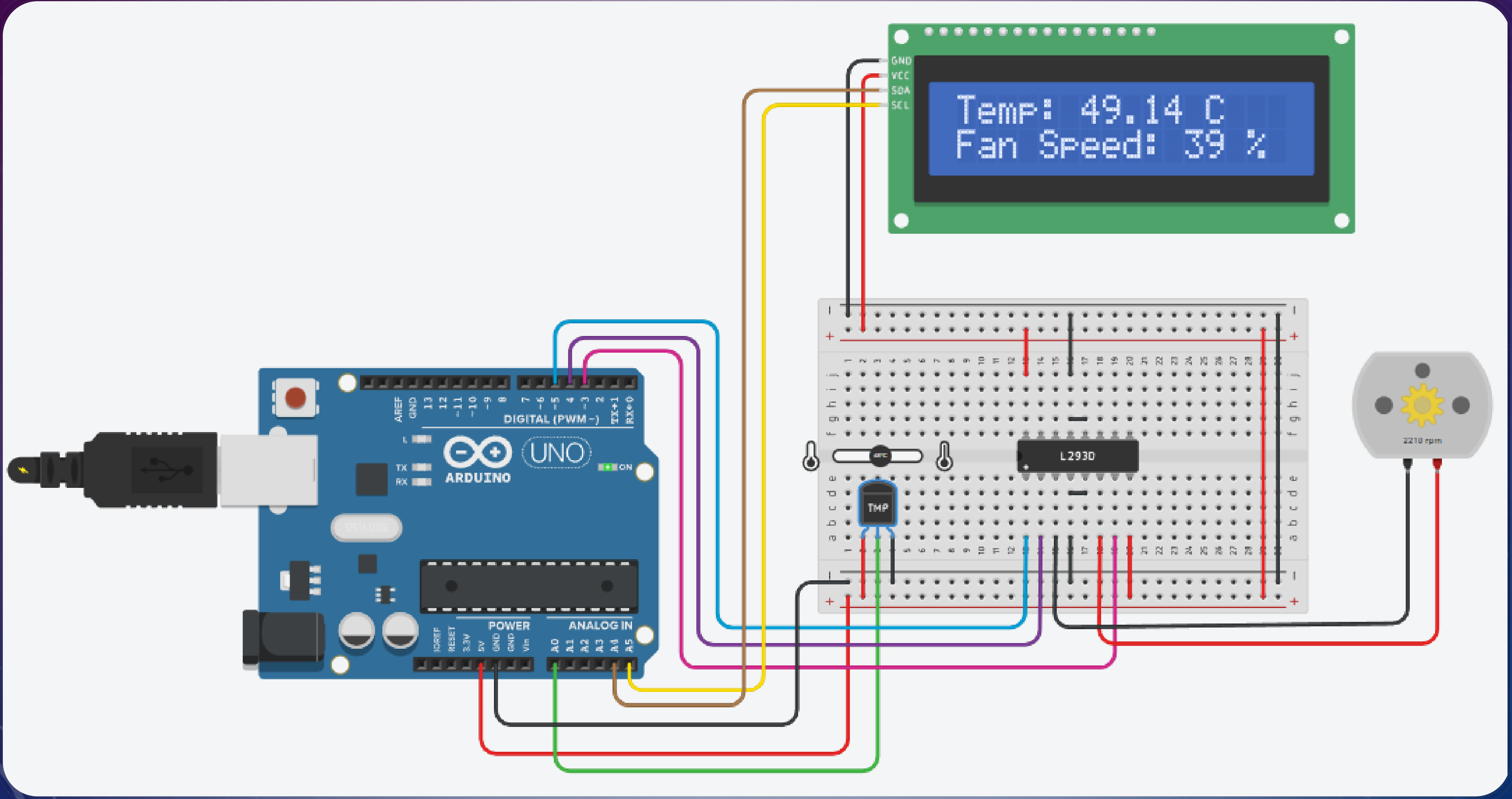
COMPONENTS USED



- The circuit for the smart fan project includes the following components:

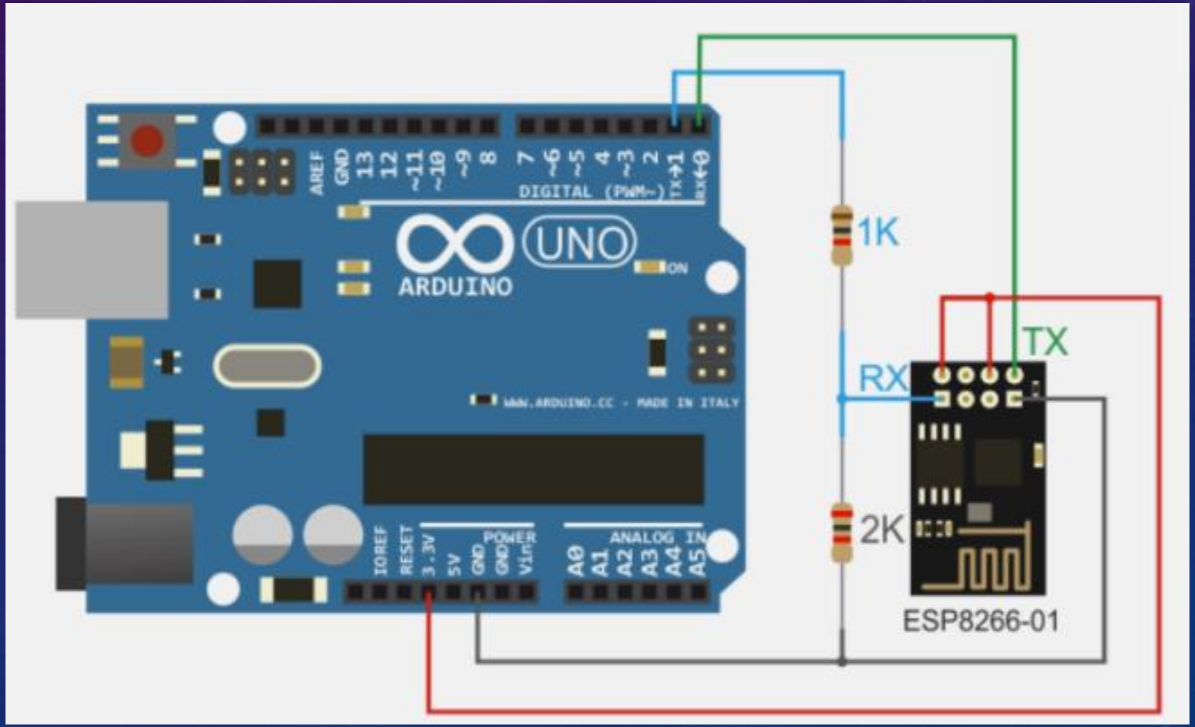
Quantity	Component
1	Arduino Uno R3
1	DHT22 Temperature and Humidity Sensor
1	5V 40x40mm DC Fan
1	L298N H-bridge Motor Driver
1	LCD 20 x 4 (I2C) Screen
1	ESP8266-01 module (or another WI-FI expansion module)

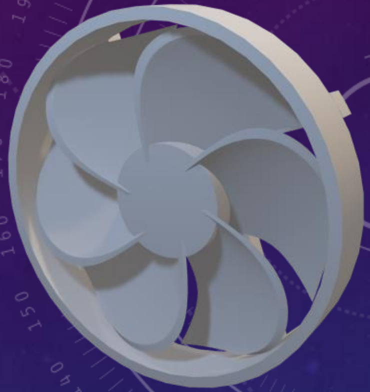
CIRCUIT DIAGRAM



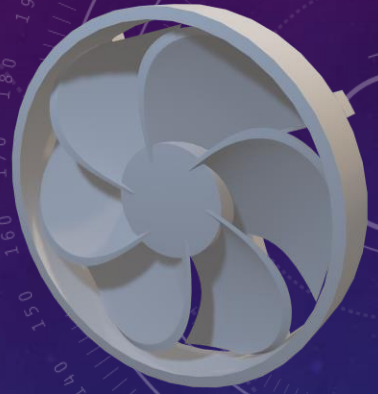
CIRCUIT DIAGRAM

Due to time constraints the following part of the circuit was not implemented but was simulated and used in the later parts of the project to demonstrate the MQTT server connection through WI-FI and the ability to view the data through a dashboard.



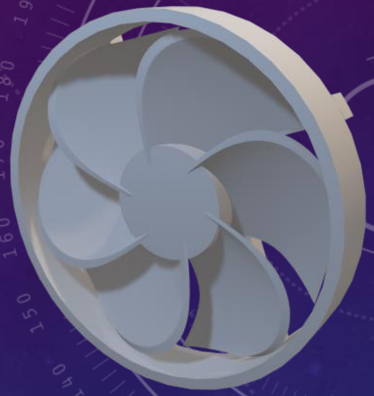


PROGRAMMING THE MICROCONTROLLER



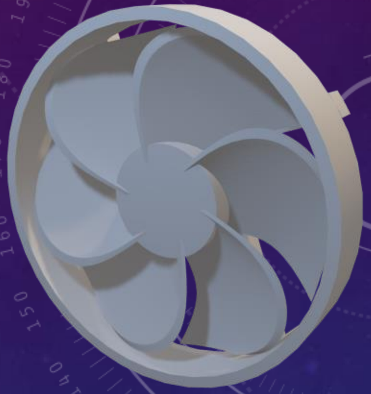
- **Steps for Temperature-based Fan Speed Control:**
  1. The microcontroller reads temperature data from the sensor.
  2. A control algorithm adjusts fan speed based on temperature readings.
  3. The microcontroller regulates fan speed according to predefined temperature thresholds.
  4. Continuous monitoring and adjustment ensure smooth and efficient operation.

THE MICROCONTROLLER READS TEMPERATURE DATA FROM THE SENSOR.



- Using the DHT22 temperature and humidity sensor we can get an accurate reading on the environment with the following line of code.
- `measure = DHT.read22(DHT22_pin);`
- Then we use the input data in the next steps to adjust the fan speed accordingly.

THE MICROCONTROLLER REGULATES FAN SPEED ACCORDING TO PREDEFINED TEMPERATURE THRESHOLDS.



- Based on the previous readings we use the following code in combination with the H-bridge motor controller to increase/decrease the fan speed accordingly.
-
  ```
   int MapTemp = map(DHT.temperature,20,80,0,255);
  ```
- The line of code above has a range of 20°C to 80°C where the fan speed will be 0% at 20°C and 100% at 80°C.

A CONTROL ALGORITHM ADJUSTS FAN SPEED BASED ON TEMPERATURE READINGS.

- The following code controls the fan motor speed :

```
  if(MapTemp>0 && MapTemp<255){
    analogWrite(MOTOR_ENABLE,MapTemp); //control motor driver
  }
  else if(MapTemp<=0){
    analogWrite(MOTOR_ENABLE,0); //control motor driver
  }
  else{
    analogWrite(MOTOR_ENABLE,255); //control motor driver
  }
```

WI-FI CONNECTION

The following part of the project was realized through a simulator to demonstrate the ability of sending data through WI-FI and enabling us to monitor the temperature, humidity and fan-speed remotely using Mosquitto as a MQTT-broker and

a Node-RED dashboard.

mosquitto

MOSQUITTO

- In this project **Mosquitto** is used as the MQTT-broker of choice. Using the following code allows us to **connect** to our MQTT Server :

```
    if (client.connect(clientId.c_str())) {
      Serial.println("Connected");
      client.publish("/ThinkIOT/Publish", "Welcome");
      client.subscribe("/ThinkIOT/Subscribe");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
```

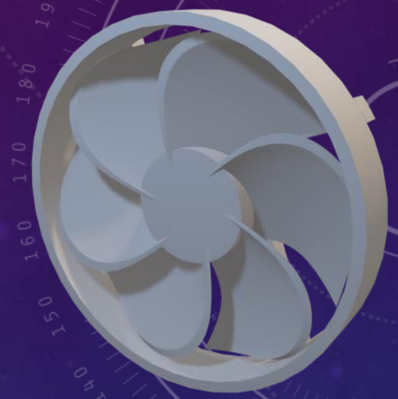
NODE-RED FLOW (PUBLISHER)

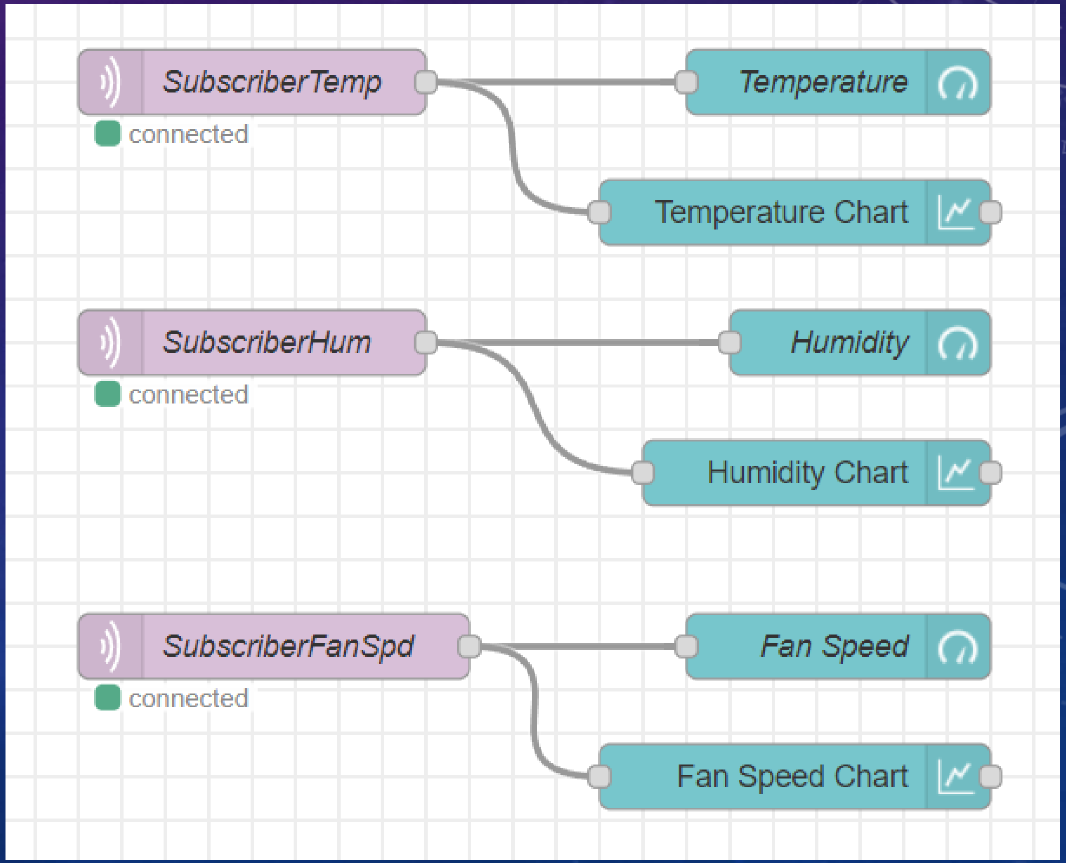
- The following code allows us to **publish** the data from our circuit through an **MQTT** broker to our **Node-RED subscriber** “nodes”.

```
String temp = String(data.temperature, 2);
client.publish("/FanSystem/temp", temp.c_str());
String hum = String(data.humidity, 1);
client.publish("/FanSystem/hum", hum.c_str());
String fanP = String(fanPercent);
client.publish("/FanSystem/fanPercent", fanP.c_str());
```

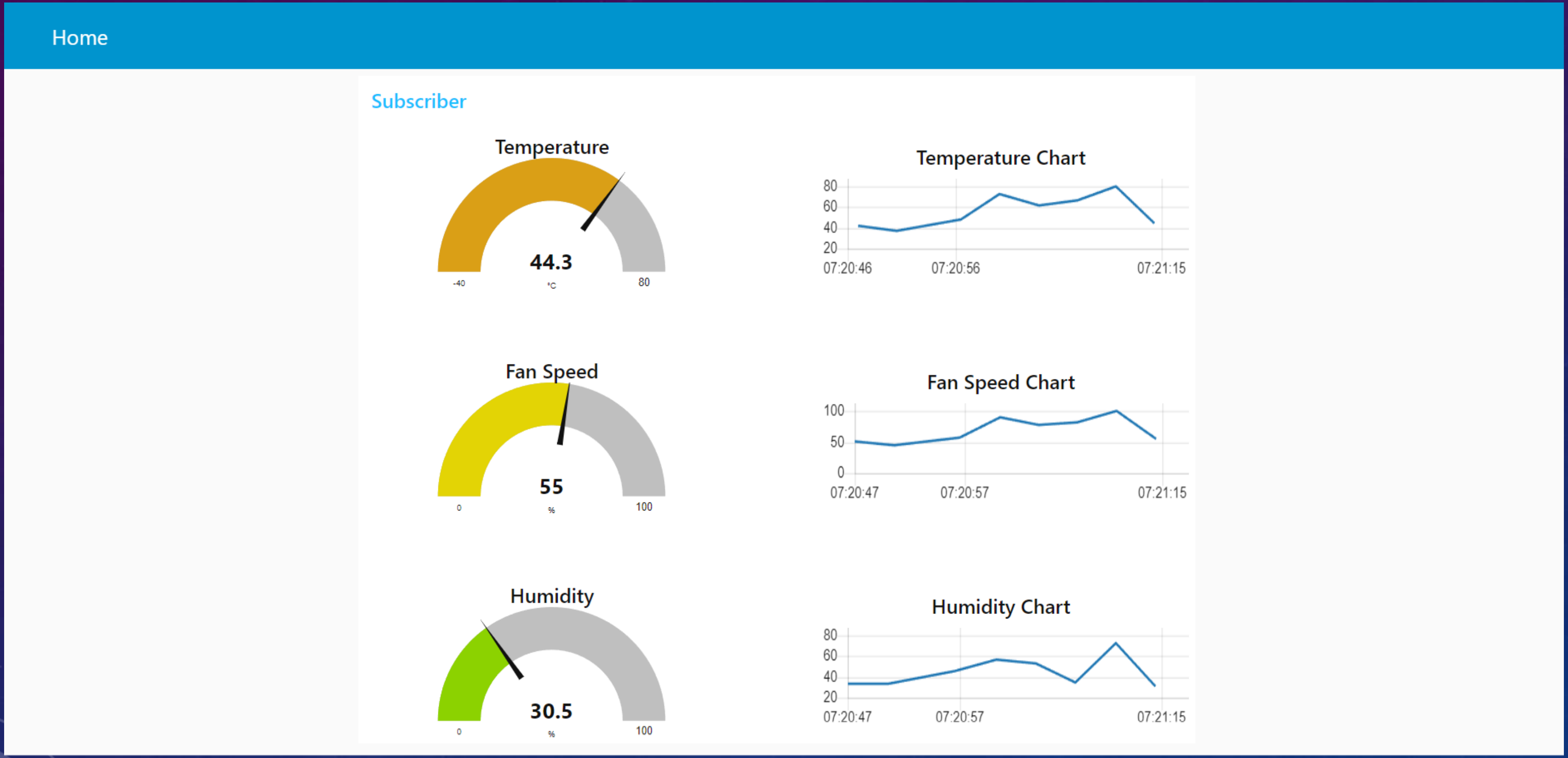
NODE-RED FLOW (SUBSCRIBER)

- Using the following “Flow” we use the three **subscriber** nodes each following a topic we published to display on **gauges** and **charts** on the **dashboard**.



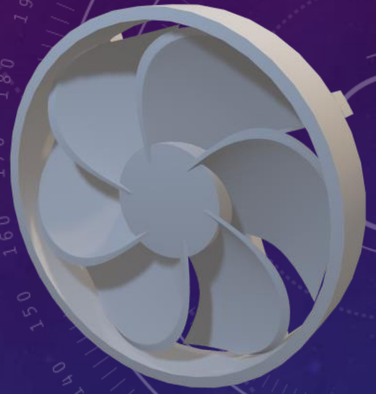


DASHBOARD

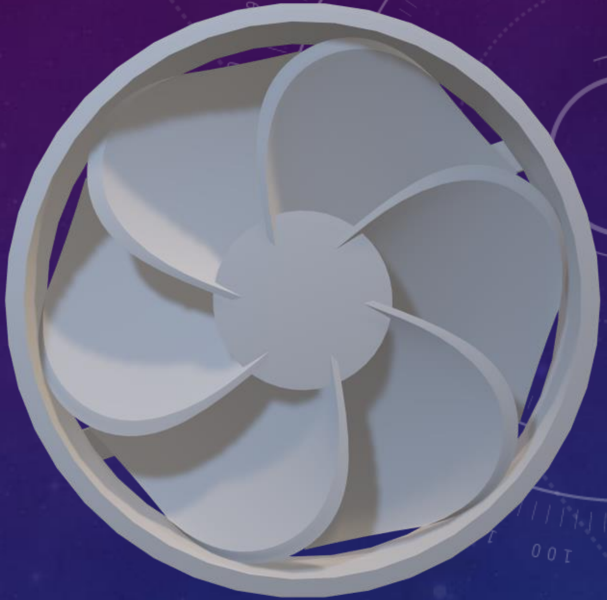


Home

Subscriber



CONCLUSION



- **Achievements:**
  - **Developed** and **implemented** a **smart fan** with temperature control.
  - The fan adjusts speed based on the **surrounding** temperature.
  - Provides a **comfortable** and **energy-efficient** cooling experience.
  - Allows monitoring of the **temperature** and **fan speed** through a dashboard.
- **Future Improvements:**
  - Implementing the **demonstrated** circuit through a **WIFI** module (ESP8266) which allows **monitoring** and manual fan speed **control** through a mobile app.
  - Expand compatibility with other smart home devices such as humidifiers, central cooling panels, etc.
  - Enhance fan connectivity and responsiveness.