ELECTRONIC PRODUCT ENGINEERING WORKSHOP

(ECP 307)

Supervised By-

Prof. S. Bhagat

Electronics & Communication Engineering

PROJECT WORK

PROJECT NAME: AIR QUALITY MANAGEMENT SYSTEM

TEAM MEMBERS:

BT18ECE035-- Aditya Wadichar

BT18ECE032-- Saurabh Kemekar

BT18ECE042-- Pranav Mehar

BT18ECE034-- Vishwesh Pillai

Department of Electronics & Communication Engineering
Visvesvaraya National Institute of Technology.

AIR QUALITY MANAGEMENT SYSTEM
Project Report

AIM

The project aims to develop a semi-automated air quality monitoring device that can alert the user on potentially dangerous situations and can take precautionary measures.

INTRODUCTION

Home automation is considered to be part of a luxurious lifestyle and it has become an important part of our lives. With the help of smart devices, we would be able to operate our heating and cooling as well as turn on and turn off lights with a single click from anywhere in our house. This is not just an efficient procedure but it will also help save electricity by avoiding the unnecessary use of electric appliances.
The device is based on the concept of a home automation system. It has temperature, humidity and various gas sensors. It gives information about the concentration of gases and temperature & humidity level to the administrator via an android app. It also notifies the user when levels increase above safe level and can take precautionary actions in dangerous situations according to need. (Here switching on/off exhaust fan).

COMPONENTS USED

1) **ESP32 -- >** ESP32 consists of Tensilica Xtensa 32-bit LX6 microprocessor with a clock speed of 2.4GHz. It is a low-cost, low-power system on a chip (SoC) series with Wi-Fi & dual-mode Bluetooth 4.2 capabilities.It consists of 802.11 b/g/n (802.11n @ 2.4 GHz up to 150 Mbit/s) as wifi protocol. ESP32 has various input/output peripherals that include analog to digital converters, digital to analog converters.

2) **DHT11 -- >** The DHT11 is a temperature and humidity sensor. It uses a humidity sensor and a temperature sensor to measure the surrounding air temperature and humidity and gives a digital signal on the data pin. Temperature ranges from 0°C to 50°C. For measuring humidity they use the humidity sensing component which has two electrodes with moisture holding substrate between them. So as the humidity changes, the conductivity of the substrate changes, or the resistance between these electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller. Similar to measuring the Temperature DHT11 use the thermistor

(Temperature - dependent resistor) and change in resistance value is used to calculate the temperature.

3) **MQ2 -- >** MQ2 is a Metal Oxide Semiconductor (MOS) type gas sensor. Gas Sensor (MQ2) module is useful for gas leakage detection. It is suitable for detecting H2, LPG, CH4, CO, Alcohol, Smoke, or Propane. The sensitivity of the sensor can be adjusted by a potentiometer. The detection is based upon a change of resistance of the sensing material when the Gas comes in contact with the material. Using a simple voltage divider network, concentrations of the gas can be detected.

4) **RELAY -- >** A relay is an electrically operated switch. Relays are used where it is necessary to control a circuit by an independent low-power signal. The relay can be operated in any of the NC (normally closed) or NO (normally open) mode. When a relay is off, the COMMON is connected to the NC and when a relay is on, the COMMON is connected to NO. Here relay is used in NO mode.

SOFTWARE USED

1) **ARDUINO IDE -- >** The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, etc. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

2) **MIT INVENTOR -- >** MIT App Inventor is a web application integrated development environment that allows us to build fully functional apps for smartphones and tablets. This cloud-based tool is used to develop an app that is a medium of communication with an administrator.

3) **FIREBASE -- >** Firebase is a mobile and web app development platform that provides developers with tools and services to help them develop high-quality apps. Firebase provides a real-time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored in Firebase's cloud.

4) **TINKERCAD -- >** Tinkercad is a free online circuit simulation software. The designed circuit including sensors and microcontroller were simulated in this software.

WORKING

The project has two parts-
1) Device (hardware)
2) App (software)

The device has ESP32 as a microcontroller which is powered by DC power supply. Two sensors are attached to it. 1) DHT11 measures humidity and temperature, 2) MQ2 measures smoke and other harmful gases. The analog input obtained from sensors is first converted to their respective units.

Due to the high sensitivity of sensors, we observed that the input has some discontinuity in the form of peaks even if conditions are normal. This can lead to a false alarm which is inconvenient for the user. To resolve this issue, we applied exponential smoothing to the past ten data values for a given parameter, with weight decreasing with time.

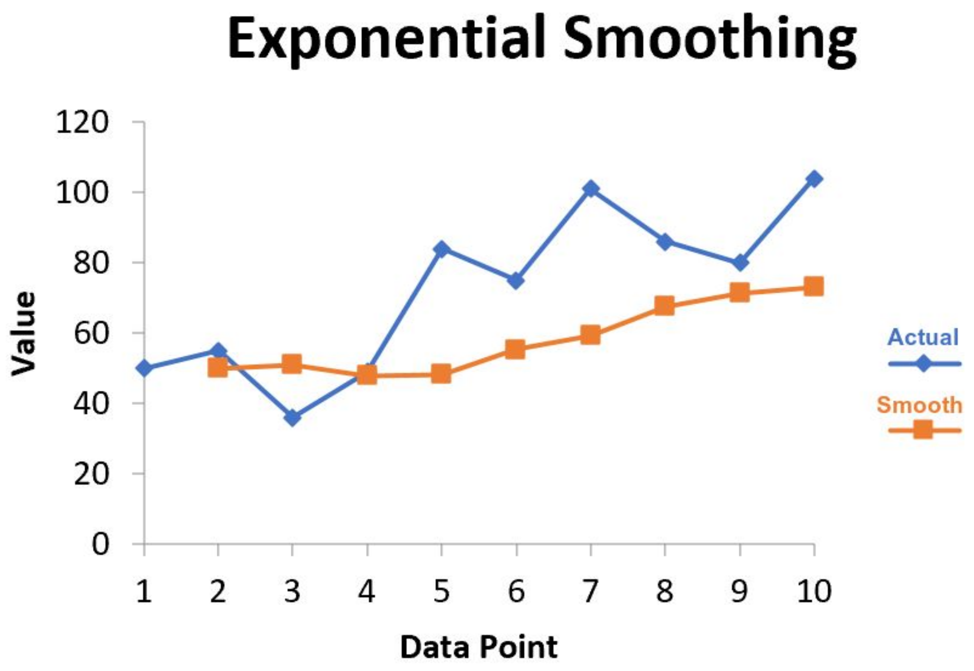


Fig 1: In the figure above, we observe that the raw data has quite a number of peaks. After applying exponential smoothing, much of the irregularity decreases while still keeping the general upward trend of the data.

Formula:

$$f(n) = \frac{\sum_{i=0}^{9} 0.8^{i} f(n-i)}{\sum_{i=0}^{9} 0.8^{i}}$$

Now that we have the final data values ready, we categorize them as safe, unsafe and dangerous. In the safe region, we keep the exhaust fan off. In the unsafe region, we notify the user that conditions are unsafe and ask to switch on the fan if it was off initially. In the dangerous region, we switch on the fan as a precautionary measure. If conditions are normal again, we switch off the fan.
Here if thresholds for switching on and switching off the fan are the same, the fan may continuously toggle its state which may damage the circuit. Hence the threshold for switching off the fan is kept slightly below than switching on. The case of notifying the user has similar logic. This can be understood by the following graph.

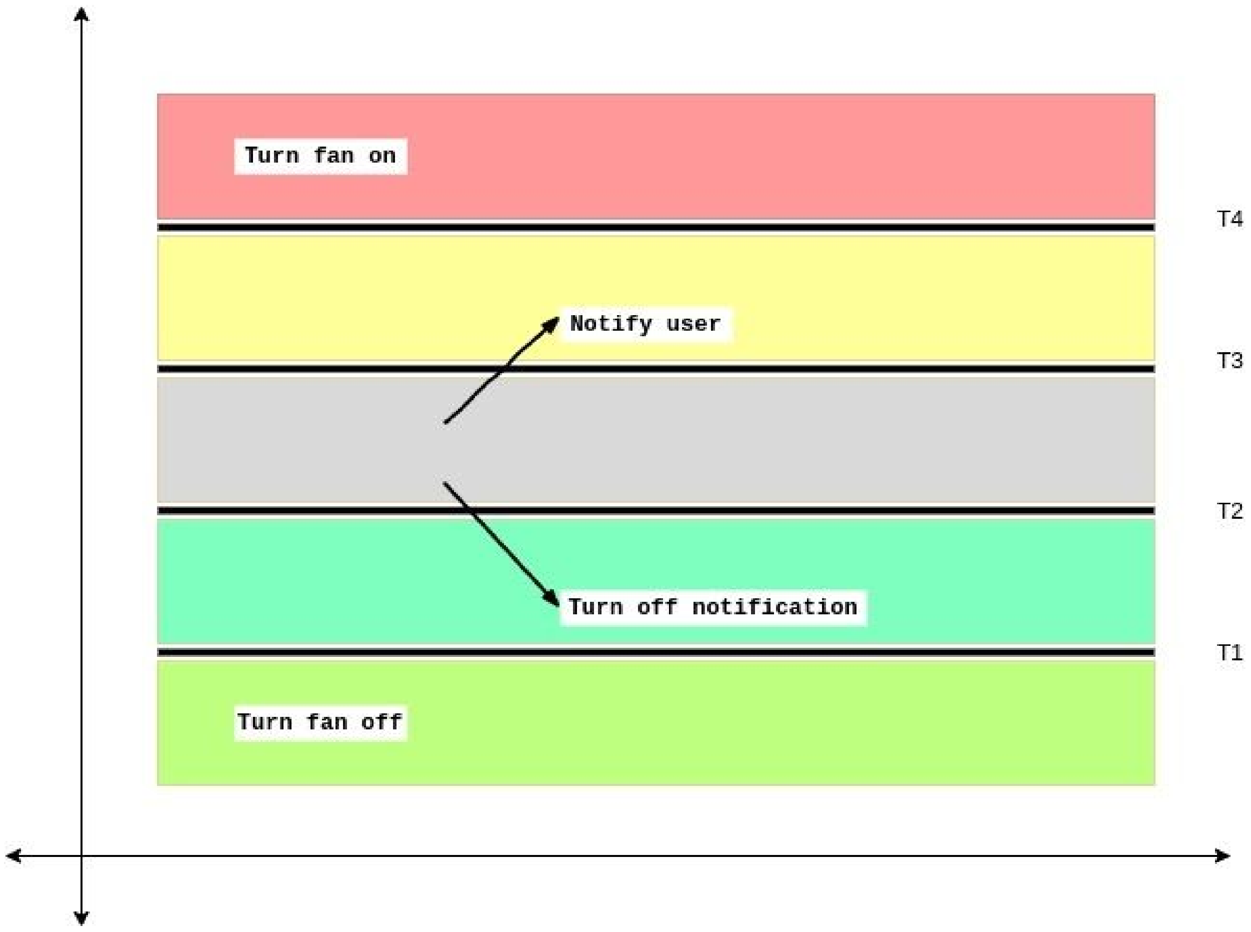
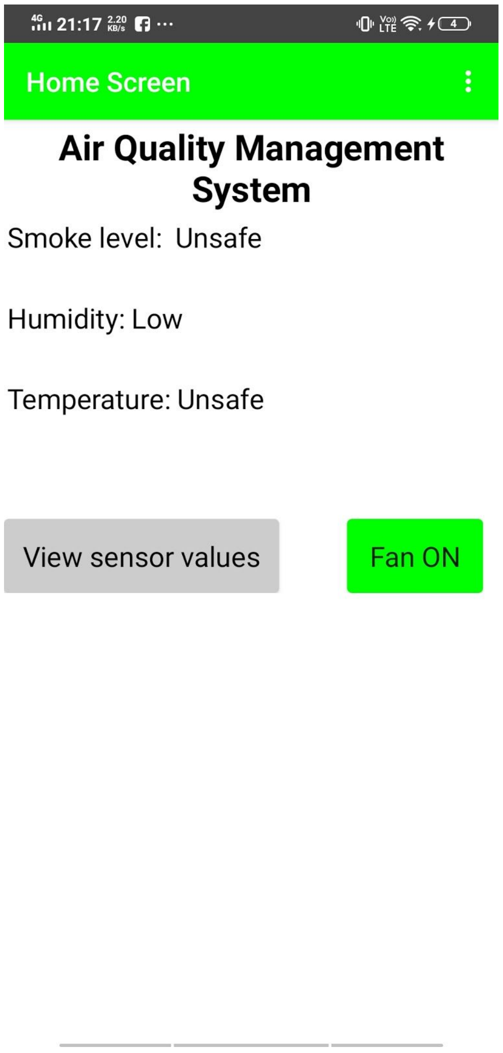


Fig 2: In the graph, we see that if levels are above *threshold 4 (T4)*, the fan is switched off automatically. When the level crosses *T3* and has an *increasing* nature, the user is notified. When the level crosses *T2* and has a *decreasing* nature, the notification is turned off. If the level falls below *T1*, then the fan is turned off.

We store the data into our Firebase database which can be accessed by the mobile app for the user. The app has been created using MIT App Inventor. It has two screens, a home screen and sensor values screen. The home screen gives information about concentration levels of smoke, humidity and temperature. The sensor data screen shows the sensor data of sensors in their respective units. The notification sent by the device is also flashed on the app screen. Finally, the user also has complete control over the exhaust fan so they can switch on/off the fan as and when desired. A switch is provided on the home screen for this purpose which also shows the current state of the fan.

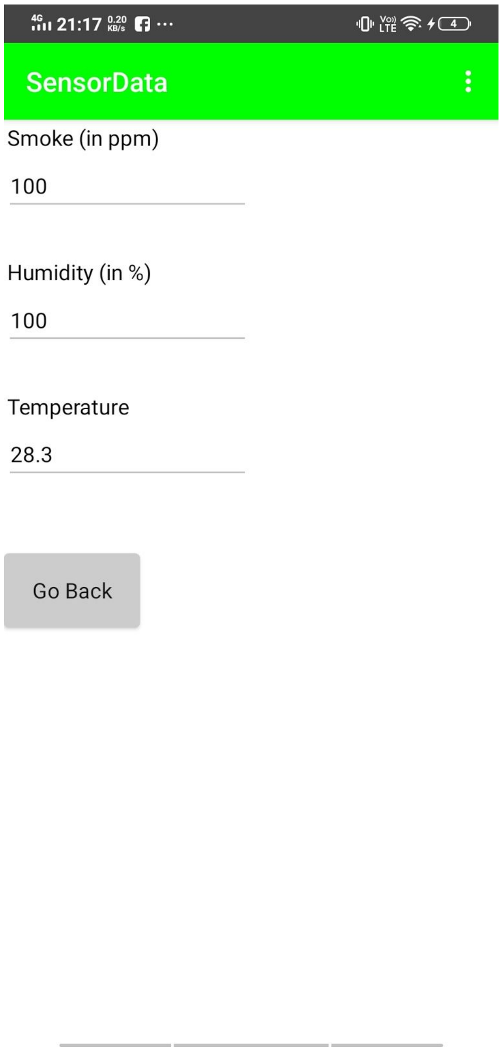


Air Quality Management System

Smoke level: Unsafe

Humidity: Low

Temperature: Unsafe



Smoke (in ppm)

100

Humidity (in %)

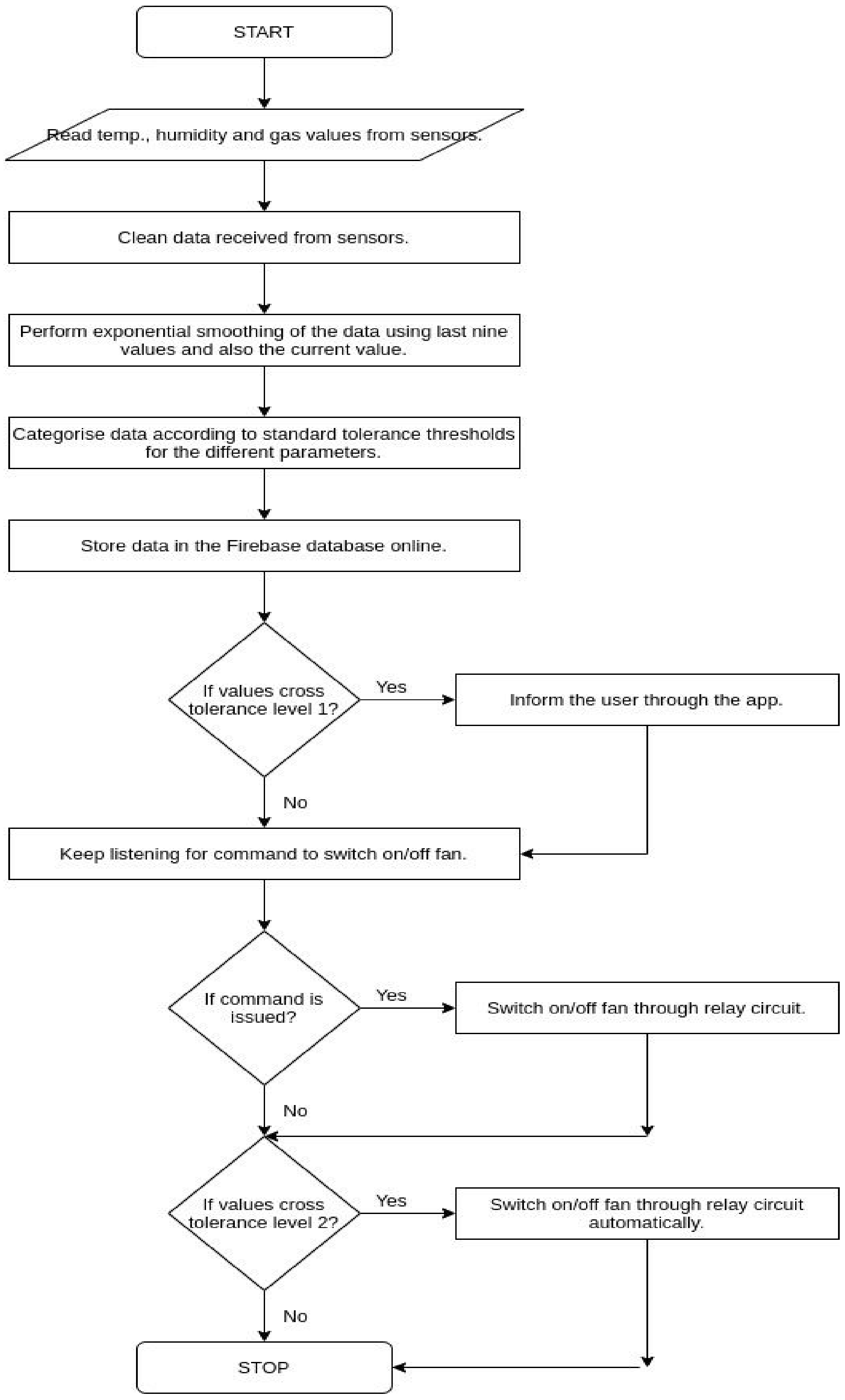
100

Temperature

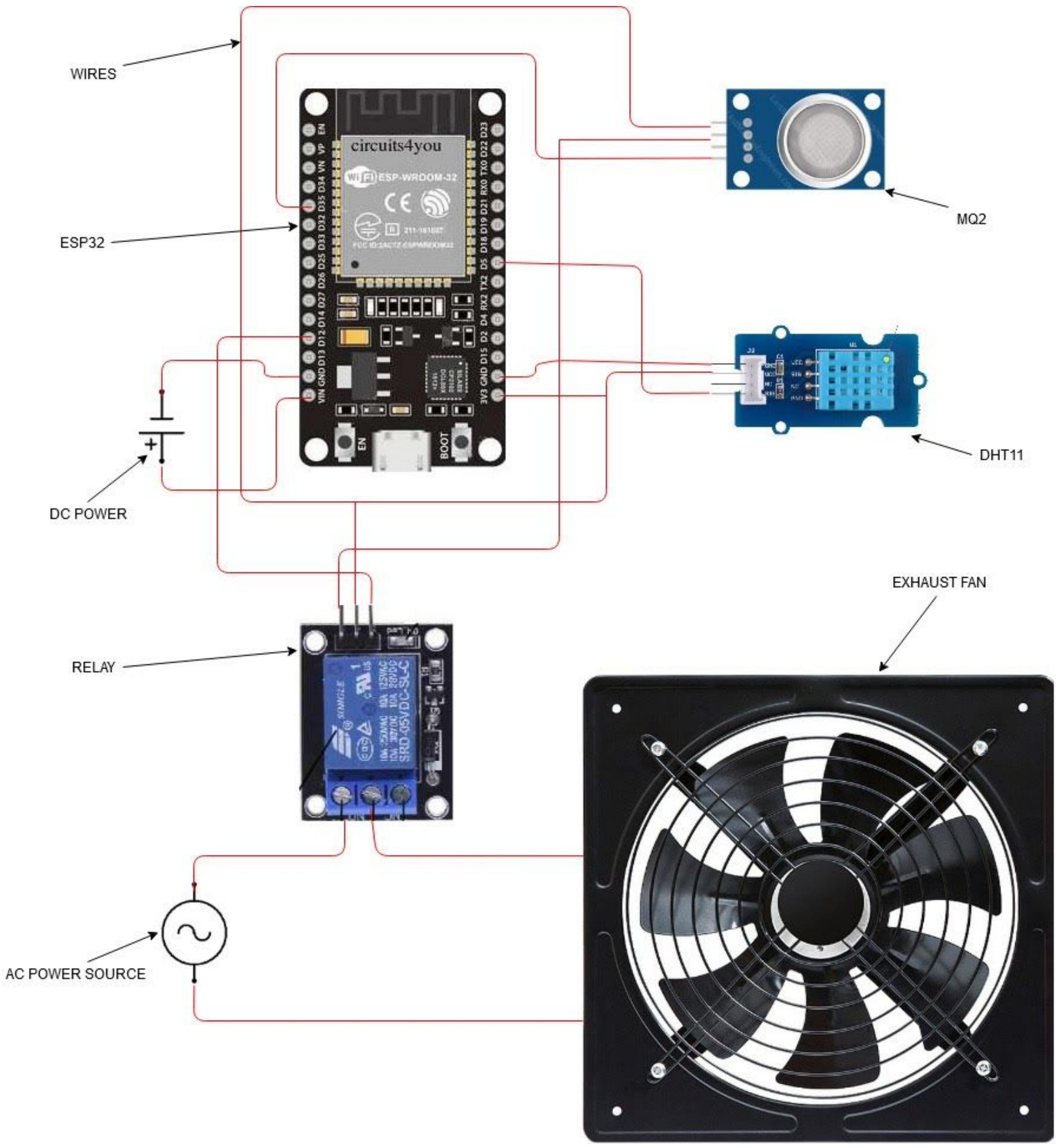
28.3

Then according to user input and concentration level, the state of the fan is decided and output is given to relay accordingly. The relay supplies AC power to the exhaust fan according to the input given. Here relay is operated in NO mode. Hence the fan is off unless logic high input is given to the relay.

FLOWCHART


START
Read temp., humidity and gas values from sensors.
Clean data received from sensors.
Perform exponential smoothing of the data using last nine values and also the current value.
Categorise data according to standard tolerance thresholds for the different parameters.
Store data in the Firebase database online.
If values cross tolerance level 1?
Yes
Inform the user through the app.
No
Keep listening for command to switch on/off fan.
If command is issued?
Yes
Switch on/off fan through relay circuit.
No
If values cross tolerance level 2?
Yes
Switch on/off fan through relay circuit automatically.
No
STOP

CIRCUIT DIAGRAM / SYSTEM DESIGN



STATUS OF WORK DONE

Measuring and processing of sensor data has been completed. The data was labeled into levels according to set thresholds and the output was shown with help of an LED arrangement as a prototype. The app has also been built and the Firebase database has been set up. The app was tested by supplying dummy data through the database.

Therefore, the device can now measure and process sensor data, and the app can fetch sensor data from the online database.

The third and last part of the project,ie, the connection of the ESP32 device to the online Firebase database was interrupted by the unforeseen COVID-19 situation and the nationwide lockdown and is incomplete as of now.

FUTURE SCOPE

For now, a single device is controlled by the administrator. We can develop a network of sensors placed in large areas like factories or an intersection in a city. Depending on needs, the outputs can be taken in many forms like an exhaust fan, alarm, air conditioners, etc.

We can also add a feature which notifies the user if a particular sensor malfunctions and needs to be replaced. This can be done by analysing data from the sensors and checking them against expected values. Since the device is modular in nature, implementing this feature will also be cost-efficient.

CODE

```
#include <FirebaseESP32.h>

#include "DHT.h"
#include <MQ2.h>
#include <WiFi.h>

#define FIREBASE_HOST "https://air-quality-a7167.firebaseio.com/"
#define FIREBASE_AUTH "DPikF7rxmpNCVJqfUzbDNuQHbyqIlJBooZjUDRAY"
#define WIFI_SSID "Gangs of WiFipur"
#define WIFI_PASSWORD "12345678"
FirebaseData firebaseData;
int gasin = A5;
int tempin = A4;
int pinmq2 = 35;
int temp;
int gasconc;
float smoke_arr[10]={0,0,0,0,0,0,0,0,0,0};
float temp_arr[10]={0,0,0,0,0,0,0,0,0,0};
float humidity_arr[10]={0,0,0,0,0,0,0,0,0,0};
float prev_smoke=0;
float prev_temp=0;
float prev_humidity=0;

//thresholds
int threshT1=30;
int threshT2=32;
int threshT3=35;
int threshT4=40;

int threshG1=60;
int threshG2=100;
int threshG3=120;
int threshG4=150;

int threshH1=40;
int threshH2=45;
int threshH3=50;
int threshH4=60;

#define DHTPIN 34
#define DHTTYPE DHT11   // DHT 11
MQ2 mq2(pinmq2);

int lpg, co, smoke;
DHT dht(DHTPIN, DHTTYPE);

int isFanRequired(float val, float prev_val, float T1, float T2, float T3,
float T4)
{
  if(val>T4)
  {
    //Turn fan on
    return 0;
  }
  else if(val>T3 && prev_val<=val)
  {
    //notify on
    return 1;
  }
  else if(val<T2 && prev_val>=val)
  {
    //notify off
    return 2;
  }
  else if(val<T1)
  {
    //Turn fan off
    return 3;
  }
  else
  {
    //Do nothing
    return -1;
  }
}

void setup() {
  // put your setup code here, to run once:
  pinMode(gasin,INPUT);
  pinMode(tempin,INPUT);
  dht.begin();
  mq2.begin();
  Serial.begin(115200);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

}

void loop() {
  // put your main code here, to run repeatedly:
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  delay(500);
   //Serial.print(temp);
   //Serial.print(gasin);
    float humidity = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float temp = dht.readTemperature();
  float* values= mq2.read(false);

  //lpg = values[0];
  lpg = mq2.readLPG();
  //co = values[1];
  //smoke = values[2];
  smoke = (mq2.readSmoke()+mq2.readCO())/2;
  if (smoke<0)
  {
    smoke=10000;
  }

FirebaseJson updateData;
updateData.set("data1","value1");
if (Firebase.updateNode(firebaseData, "/sensor-values", updateData)) {
  Serial.println(firebaseData.dataPath());

  Serial.println(firebaseData.dataType());

  Serial.println(firebaseData.jsonString());

} else {
  Serial.println(firebaseData.errorReason());
}
  if (isnan(smoke)){
    Serial.println(F("Failed to read from mq2 sensor!"));
  }
  if (isnan(humidity) || isnan(temp)) {
    Serial.println(F("Failed to read from DHT sensor!"));
  }
  for(int i=9;i>0;i--)
  {
    smoke_arr[i]=smoke_arr[i-1];
    temp_arr[i]=temp_arr[i-1];
    humidity_arr[i]=humidity_arr[i-1];
  }
  smoke_arr[0]=smoke;
  temp_arr[0]=temp;
  humidity_arr[0]=humidity;
  float norm_smoke=0;
  float norm_temp=0;
  float norm_humidity=0;
  float mult=1;
  float sum=0;
  int flag=0;
  for(int i=0;i<10;i++)
  {
    norm_smoke+=smoke_arr[i]*mult;
    norm_temp+=temp_arr[i]*mult;
    norm_humidity+=humidity_arr[i]*mult;
    sum=sum+mult;
    mult*=0.8;
  }
  norm_smoke/=sum;
  norm_temp/=sum;
  norm_humidity/=sum;


  //Serial.print(F("Humidity: "));
  //Serial.print(humidity);
  //Serial.print(F("%  Temperature: "));
  //Serial.print(temp);
  Serial.print(F("°C Norm Smoke: "));
  Serial.print(norm_smoke);
  Serial.print(" ppm Smoke: ");
  Serial.print(smoke);
  Serial.print(" ppm");
  Serial.println("");
  int smoke_status=
isFanRequired(norm_smoke,prev_smoke,threshG1,threshG2,threshG3,threshG4);
  int temp_status=
isFanRequired(norm_temp,prev_temp,threshT1,threshT2,threshT3,threshT4);
  int humidity_status=
isFanRequired(norm_humidity,prev_humidity,threshH1,threshH2,threshH3,threshH
4);

  boolean notifyON= (smoke_status==1 || temp_status==1 ||
humidity_status==1);
  boolean notifyOFF= (smoke_status==2 || temp_status==2 ||
humidity_status==2);

  boolean fanON = (smoke_status==0 || temp_status==0 || humidity_status==0);

  prev_smoke=norm_smoke;
  prev_temp=norm_temp;
  prev_humidity=norm_humidity;
  delay(500);

}
```