TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION

Fans are the integral part of many homes, industries, commercial places, and many electric and electronic appliances for cooling mechanisms. Our main focus is on ceiling fans in homes and delivering the automatic speed control of fans with respect to changes in temperature and humidity. According to a survey by Frost & Sullivan, there is 10% growth in fan market size in India each fiscal year which shows that there is still plenty of room for improvement in the fan technology.

The main factors for the growth of fan market size and the current trends are

- Innovation: Demand for premium fans with better aesthetics has been increasing for the past 3-5 years.
- Energy Saving Technology: BLDC fans are 50% more efficient compared to the conventional fans.[1]
- IoT: The smart fans are connected to the internet and they can be controlled from anywhere.

The automatic airflow control in the room is an important and developing technology. Being able to control the fan through apps, remote control is common nowadays. This technology is required because it helps in running the fan at low speeds on a dry and high temperature day. In winters, running the fans at high speed on a dry day causes skin irritation. So, the fan should be running at optimal speed to eliminate any discomforts that may arise due to running the fan at more or less than required speed. The study of the recorded temperatures and humidity of the room along with the feedback from the people, a proper selection of the fan speed can be done at a particular temperature and humidity. For example, if the temperature is more than 30 degree Celsius and the humidity is less than 15%, it denotes a dry day. The sweat evaporates very fast due to low vapor content in the room. In this case, increasing the fan speed beyond level 3 might cause sensation of heat and great discomfort. But if the fan is running at very low speeds like 1 is not fruitful. So, the ideal speed for the fan will be 3. This speed level keeps the required amount of air flow in the room along with not causing any heat sensation and discomforts.

There are many fans currently available in the market which can be controlled from the smartphone with the preinstalled apps[2]. Although they let the user control the speed of the fan manually, there is no auto support in most of the fans. The technology is still in development. The app created supports both auto and manual modes. In manual mode, the user can set the speed of his own. In case of auto mode, the user passes the control to the MCU. Most of the fans come with the MCU which is the brain of the fan. It retrieves the commands from the user

through Wi-Fi or Bluetooth connectivity. Currently, IR remote control and app control are being employed mostly.

The MCU can be programmed with higher capabilities to sense the temperature and humidity from its own sensors or it can retrieve the data from the thermostat in the room. After receiving the temperature and humidity values, the MCU can either process the corresponding optimal speed and set the same speed if the fan is running in auto mode. If the fan is running in manual mode, it runs at the speed given by the user.

1.2 OBJECTIVES

- Automatic speed control with the variations in the temperature and humidity of the surroundings.
- Smart control and personalized notifications
- Smart suggestions based on the room conditions.

CHAPTER 2

2.1 WORKING PRINCIPLE

NodeMCU is the brain of the project and it is integrated with the firebase. The temperature and humidity sensors are attached to the MCU. The working is explained as:

Step - I:

NodeMCU is connected to the internet through Wi-Fi module. The MCU starts reading the temperature and the humidity of the room from the sensors.

2.2 SENSING TEMPERATURE AND HUMIDITY

The LM35 sensor is used for sensing the temperature. DHT11 sensor is used for sensing the humidity[3]. Although DHT11 can measure both the temperature and humidity, the temperature sensing is slow and not accurate as LM35. The DHT11 sensors have their own single wire protocol used for transferring the data. This protocol requires precise timing. The DHT library takes care of almost everything. The optimal speed is calculated from the measured temperature and humidity values. The algorithm and the code are provided at the end of the step-I.

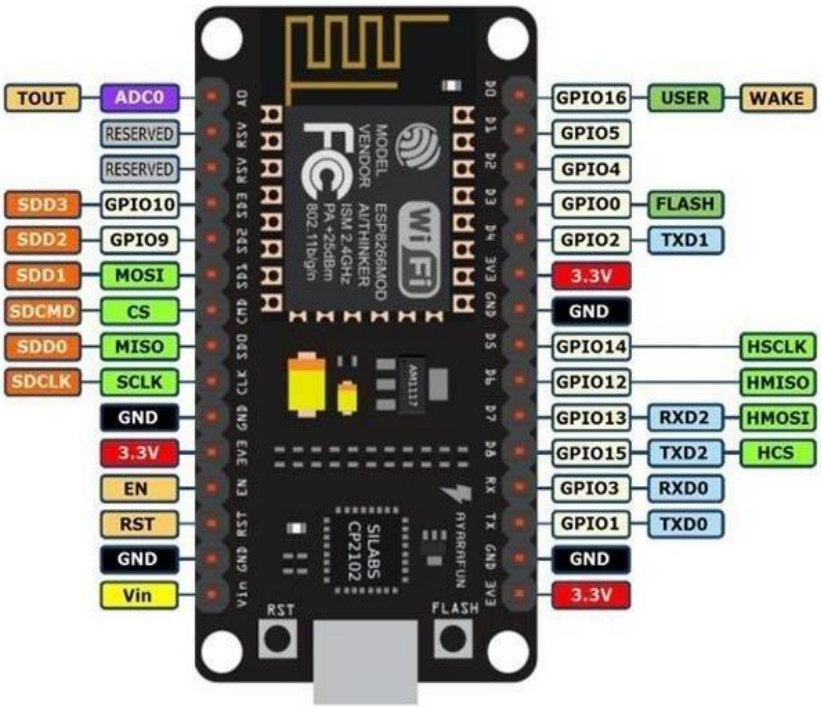


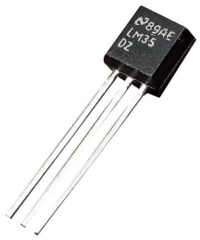
Fig: ESP8266 NodeMCU v1.0

2.2.1 ESP8266

The production board for the NodeMCU ESP8266 comes with an ESP-12E package with an ESP8266 chip and a Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and runs with a clock frequency adjustable from 80MHz to 160 MHz for data and applications, NodeMCU has 128 KB of RAM and 4 MB of Flash memory. It is suitable for IoT ventures due to its high computing capacity and in-built Wi-Fi / Bluetooth and Deep Sleep Operating functions. It is possible to power NodeMCU with the Micro USB jack and VIN pin (External Supply Pin). It supports the interface between UART, SPI, and I2C.

The following is the code that is used to connect NodeMCU to Wi-Fi

```
#include <ESP8266WiFi.h>
#include<WiFiClientSecure.h>
#define WIFI_SSID “SSID”
#define WIFI_PASSWORD “password”

void setup()
{
  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();
}
```



2.2.2 LM35

This temperature sensor can measure temperature in the range of -55 deg C to 150 degree C. It is a 3-terminal device that provides an analog voltage proportional to the temperature sensed. The output voltage increases with the temperature. The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it.

Fig: LM35 sensor

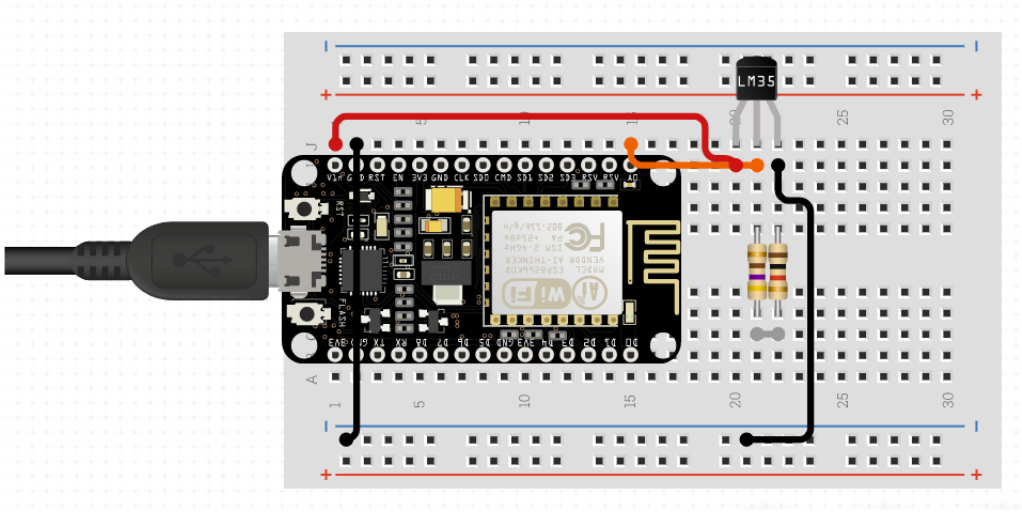


Fig: LM35 interfacing with NodeMCU

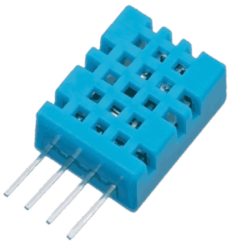
NodeMCU ADC can measure the analog voltage from LM35. The output is given to analog pin A0 of NodeMCU. The following programme is the arduino sketch for interfacing the LM35 with NodeMCU. The full circuit diagram can be found at https://www.circuito.io/app?components=513,333429,360216

```
float  vref = 5.0;
float  resolution = vref/1023;

void setup() {
 Serial.begin(115200);  /* Define baud rate for serial communication */
}

void loop() {
 float  temperature = analogRead(A0);
 temperature = (temperature*resolution);
 temperature = temperature*100;
 Serial.println(temperature);
}
```

2.2.3 DHT11

This sensor detects water vapor by measuring the electrical resistance between two electrodes. The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. The change in resistance between the two electrodes

is proportional to the relative humidity. The resistance betweens the electrodes decreases under higher relative humidity and the resistance increases for lower relative humidity. It can measure the relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50 deg C.



The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.It has 4 pins. They are Vcc, data pin, no connection and GND. The data pin is connected to the GPIO pin of the NodeMCU to record temperature and humidity.

Fig: DHT11 sensor

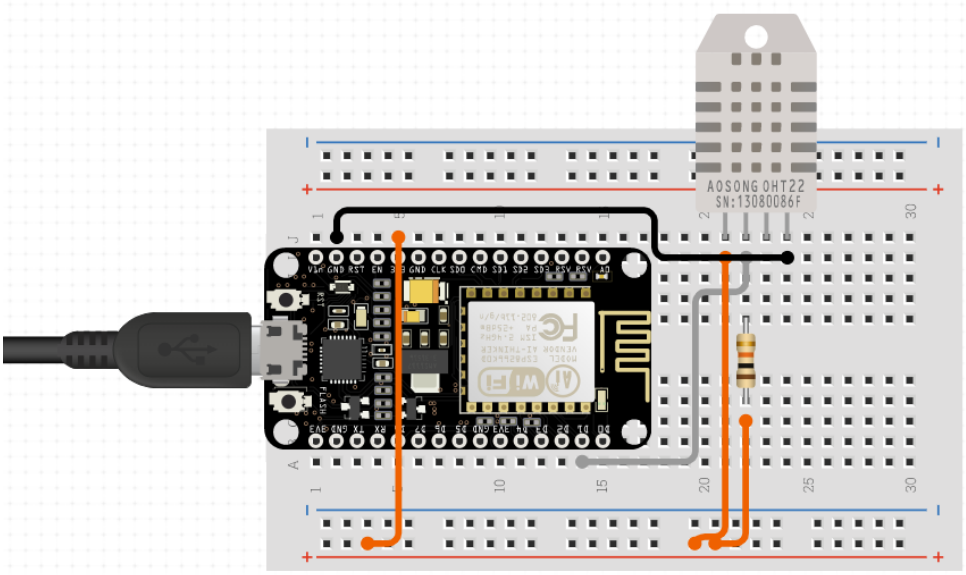


Fig: DHT11/22 sensor interfacing with NodeMCU

To use the DHT11 with NodeMCU, DHTLib library is to be installed. The following sketch is used for interfacing DHT11 with NodeMCU. The complete circuit diagram can be found at https://www.circuito.io/app?components=513,10167,360216

```
#include "DHT.h"

DHT dht;

void setup()
```

```
{
  Serial.begin(9600);
  Serial.println();
  Serial.println("Humidity (%)\tTemperature (C)");
  dht.setup(D1);    /* D1 is used for data communication */
}

void loop()
{
  float humidity = dht.getHumidity();     /* Get humidity value */
  float temperature = dht.getTemperature();    /* Get temperature value */
 }
```

2.2.4 Optimal Speed Calculation

The correspondingSpeed function is declared to take temperature and humidity as input variables. It returns the optimal speed for the fan.

```
int correspondingSpeed(int temp, int hum){
  int ideal = 5;
  //temperature greater than 37
  if(temp >=37 && hum<15){
    ideal = 3;
    return ideal;
  }
  else if(temp>=37 && hum<55){
    ideal = 4;
    return ideal;
  }
  else if(temp>=37 && hum>=55){
    ideal = 5;
    return ideal;
  }
  //temperatures between 37 and 30
  if(temp>=30 && temp<37 && hum<15){
    ideal = 3;
    return ideal;
  }
```

```
  else if(temp>=30 && temp<37 && hum<55){
    ideal = 4;
    return ideal;
  }
  else if(temp>=30 && temp<37 && hum>=55){
    ideal = 5;
    return ideal;
  }
  //temperatures less than 30
  if(temp < 30 && hum<15){
    ideal = 1;
    return ideal;
  }
  else if(temp<30 && hum<45){
    ideal = 2;
    return ideal;
  }
  else if(temp<30 && hum<75){
    ideal = 3;
    return ideal;
  }
  else if(temp<30 && hum>= 75){
    ideal = 4;
    return ideal;
  }
}
```

Step - II:

NodeMCU retrieves the data from the real-time database. It checks whether the fan is ON/OFF, whether the mode is MANUAL/AUTO. Also checks the speed input given by the user in case of MANUAL mode. NodeMCU processes all these signals and provides the corresponding output signals to the electronic speed controller.

2.3 FLUTTER AND FIREBASE INTEGRATION

The flutter app is integrated with the firebase real-time database. With the same real-time database, the MCU is integrated. Two roots are created in the database. One root is Android root with children: speed, test, mode, switch. The other is the Controller root with children: test, temp, hum.

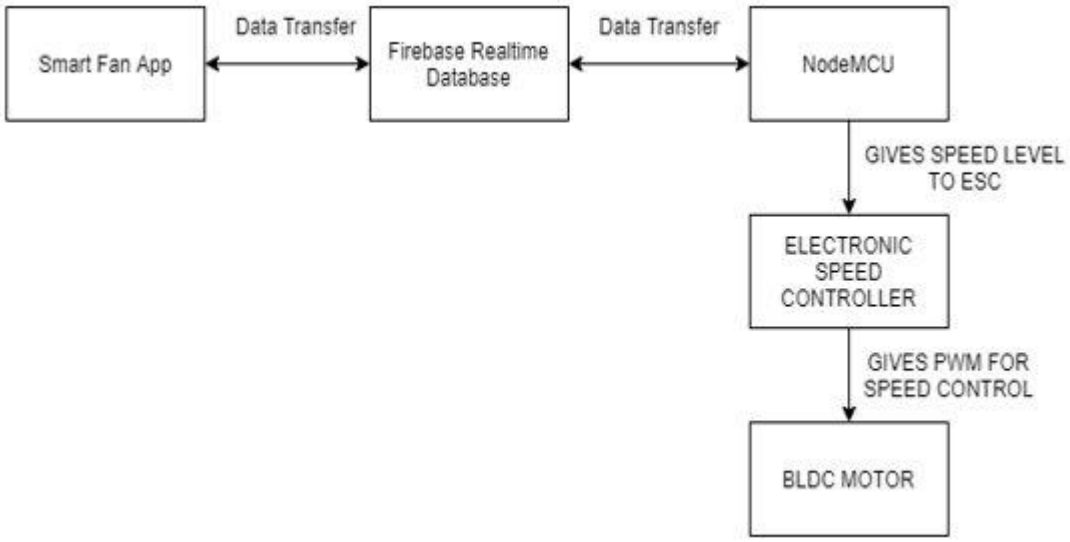


Fig: Flutter App and nodeMCU working

The MCU constantly reads the children of the android and produces output according to the commands given. The android app constantly monitors the changes in the controller root and updates the same in the SMART FAN application.[4]

Fig: Firebase Real-time database

2.3.1 Flutter

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. It allows one to create a native mobile application with only one codebase which means that one can use one programming language and one codebase to create two different apps (for iOS and Android). It is cheaper to develop a mobile application with Flutter because one doesn't need to create and maintain two mobile apps. Most importantly it is performant, that is one won't notice the difference between a native application and a Flutter app.[5]

2.3.2 Firebase

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. **Firebase** is categorized as a NoSQL database program, which stores data in JSON-like documents. Firebase is easy to integrate into Flutter app and also NodeMCU. The firebase library from "FirebaseESP8266.h".

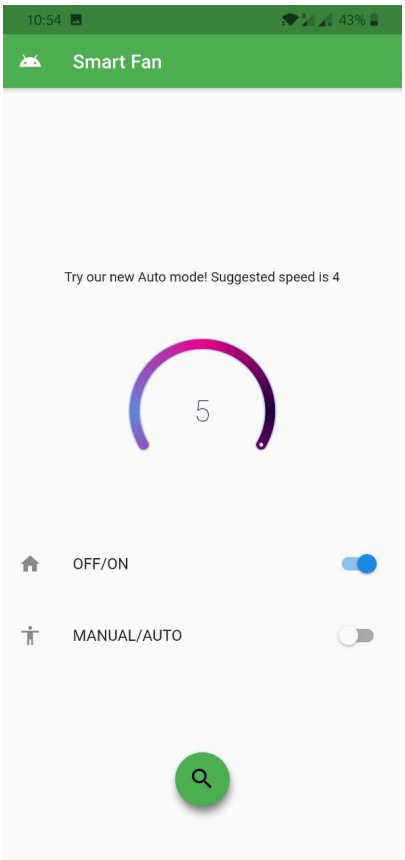


Fig: Smart Fan app by Flutter App interface

The following is the part of Arduino sketch to integrate firebase with the NodeMCU with firebase. The temperature and the humidity values are updated and the switch command is read from the user are included in the code.

```
#include "FirebaseESP8266.h"
#include <FirebaseESP8266.h>

//Firebase credentials update
#define FIREBASE_HOST "firebasehostaddress"
#define FIREBASE_AUTH "firebaseauthid"

//Defining FirebaseESP8266 data object for data sending and receiving
FirebaseData fbdo;

void setup()
{
   //Setting the Firebase info
   Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
   Firebase.reconnectWiFi(true);
}

void loop()
{
   //converting temperature and humidity values into strings for updating into rtdb
   String temp = String(temperature);
   String hum = String(humidity);
   //updating temperature in database
   if(Firebase.setString(fbdo, "/controller/temp", temp)){
   Serial.print(temp + " ");
   Serial.println("Temperature Updated");
   digitalWrite(LED_BUILTIN, HIGH);
 }
 else{
   //printing error in case of failure
   Serial.print("Error in set, ");
   Serial.println(fbdo.errorReason());
```

```
}
digitalWrite(LED_BUILTIN, LOW);
//updating humidity in database
if(Firebase.setString(fbdo, "/controller/hum", hum)){
  Serial.print(hum + " ");
  Serial.println("Humidity Updated");
  digitalWrite(LED_BUILTIN, HIGH);
}
else{
  //printing error in case of failure
  Serial.print("Error in setting humidity, ");
  Serial.println(fbdo.errorReason());
}
//calculate corresponding or optimal speed
int idealspeed = correspondingSpeed(temperature, humidity);
String ideal = String(idealspeed);

//updating suggested speed
if(Firebase.setString(fbdo, "/controller/test", ideal)){
  Serial.print(ideal + " ");
  Serial.println("Suggested speed updated");
  digitalWrite(LED_BUILTIN, HIGH);
}
else{
  //printing error in case of failure
  Serial.print("Error in setting suggested speed, ");
  Serial.println(fbdo.errorReason());
}
//reading whether fan is ON/OFF
//switcch is a variable that is already defined to show status
//switch state
if(Firebase.getString(fbdo, "/android/switch")){
  switcch = fbdo.stringData();
  Serial.print(switcch + " ");
  Serial.println("Switch position");
  digitalWrite(LED_BUILTIN, HIGH);
}
```

```
  else{
    //printing error in case of failure
    Serial.print("Error in reading switch, ");
    Serial.println(fbdo.errorReason());
    return;
  }
}
```

Step-III

Electronic speed controllers provide the necessary three phase PWM signals to the BLDC motor in the fan. Thus, the speed is controlled by the MCU.

2.4 CONTROLLING THE FAN SPEED

The BLDC fans are powered from the home supply with the necessary AC-DC converter[6]. The voltage ratings of the BLDC fans vary from 12V to 24V. An electronic speed controller(ESC) controls the input to the BLDC motor thus by providing the three phase PWM voltages. The selected ESC works with the PWM signal frequency of 50Hz which is the same as the Servo motor PWM signal. Hence, the servo library can be used to control the speed. As shown in the figure, the low speed and high speed are selected. The other speeds are taken at other angles of the Servo.write(deg) function.

The servo library is defined to generate the PWM signals of 50Hz frequency. The interrupt is attached to the GPIO4(D2) of the MCU along with the maximum and minimum pulse width. The minimum pulse width is given zero for zero speed configuration but the speed level 1 starts from the pulse width of 1millisecond. The code is simple as shown below for controlling the speed of the motor.

2.4.1 ESC (ELECTRONIC SPEED CONTROLLER)

All the BLDC fan manufacturers like Orient electric fans, Atomberg, Havells use their own electronic speed controller which are controlled by the MCU integrated with both the switchboard and IR remote control. In this project, 30A BLDC ESC which is specifically made for quadcopter and multi-rotors and provides faster and better motor speed control giving better flight performance compared to other available ESCs. This is selected for running the 12V BLDC motor without any high load. In the real situations, the load and the power requirements of the BLDC motor in the fan might vary from the motor used in this project which is a quadcopter motor.

Fig: 30A BLDC ESC The considered motor is A2212 - 1400KV BLDC Brushless motor.

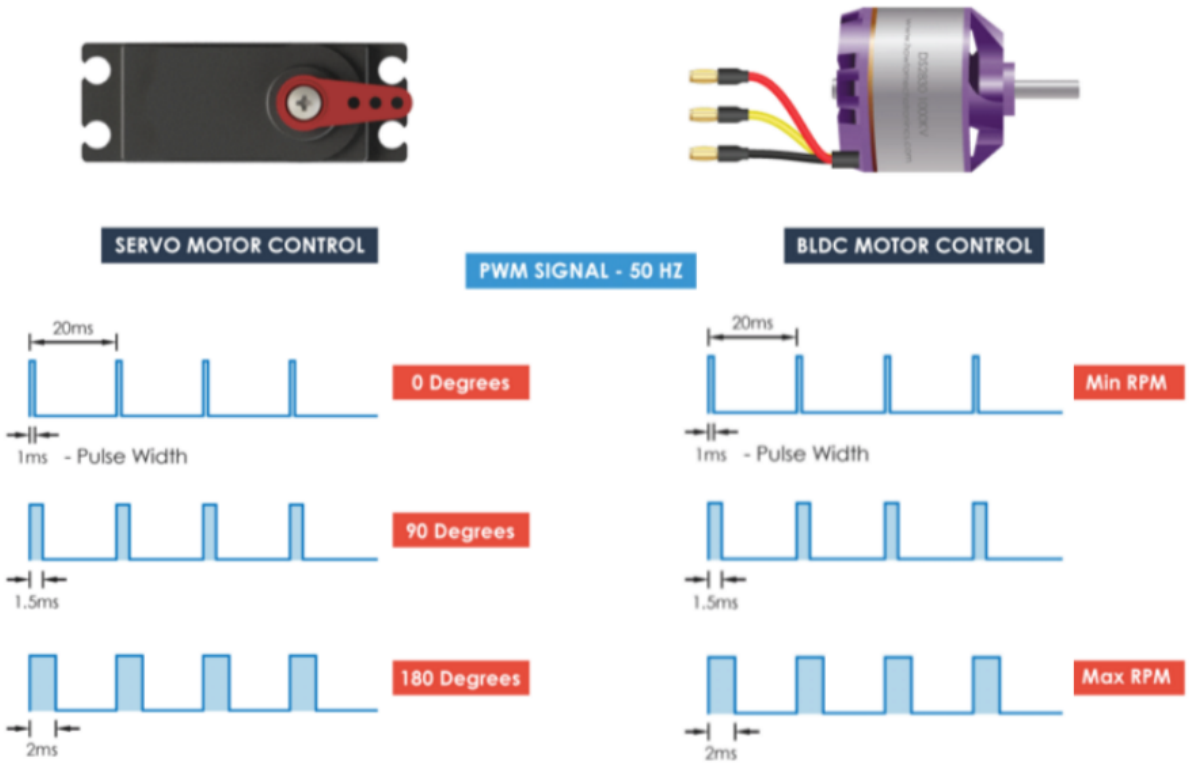


Fig. Servo motor control and BLDC motor control

Simple code for running BLDC motor at high speed and low speed:

```
#include <Servo.h>

//declaring the servo variable for fan
Servo servo;

//speed setting function
int setFanspeed(int fanspeed) {
   if(fanspeed == 0){
      servo.write(0);
   }
   else if(fanspeed == 1){
      servo.write(90);
   }
```

```
    else if(fanspeed == 3) {
        servo.write(110);
    }
    else if(fanspeed == 4) {
        servo.write(135);
    }
    else if(fanspeed == 5) {
        servo.write(180);
    }
}

void setup()
{
servo.attach(4,0,2000); //(pin, min pulse width, max pulse width in microseconds)
servo.write(0);
delay(2000);
}

void loop() {
    //fanspeed is calculated before calling the setFanspeed function
    //fanspeed variable is filled by either manual mode or auto mode
    setFanspeed(fanspeed); //sets the fan speed to the required level
}
```

CHAPTER 3

3.1 BLOCK DIAGRAM

The following block diagram depicts the outline of the working of the project(as explained in the previous section).

The block diagram is explained as:

I. The temperature and the humidity measurements are taken by MCU.
II. NodeMCU is connected to the internet and it exchanges the data with the real-time database.
III. MCU generates the required PWM signals to control the ESC.
IV. The power electronics converter shown in the figure is an AC-DC converter generally used in the BLDC fans for constant DC voltage.

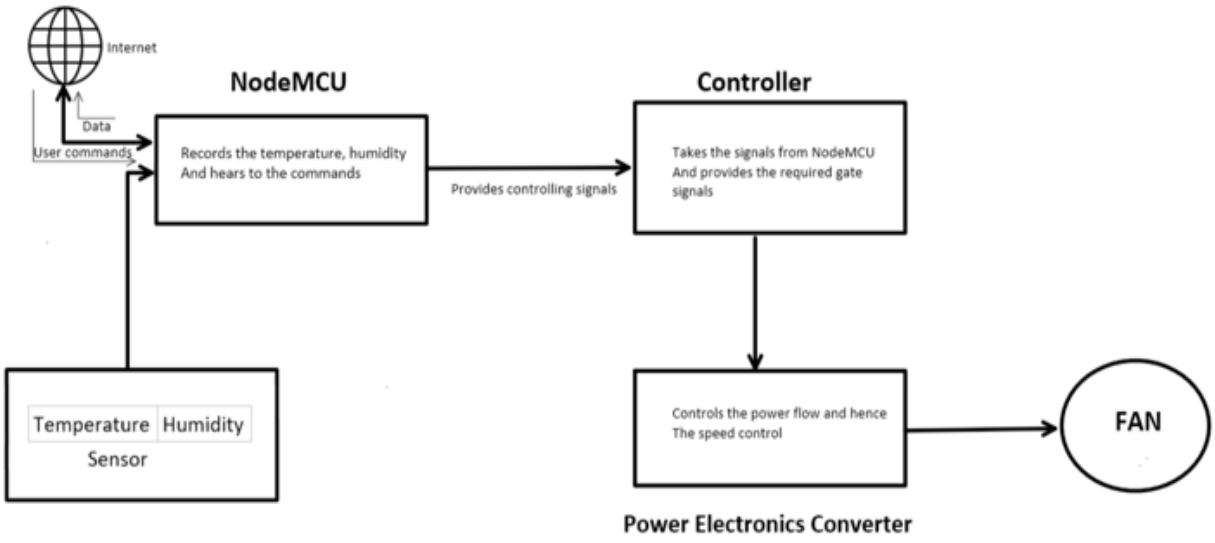


Fig: Block diagram showing working of the system

3.2 FLOWCHART

The flowchart is explained as:

I. NodeMCU is connected to the firebase real-time database via Wi-Fi.
II. Senses and updates the temperature and humidity of the room.

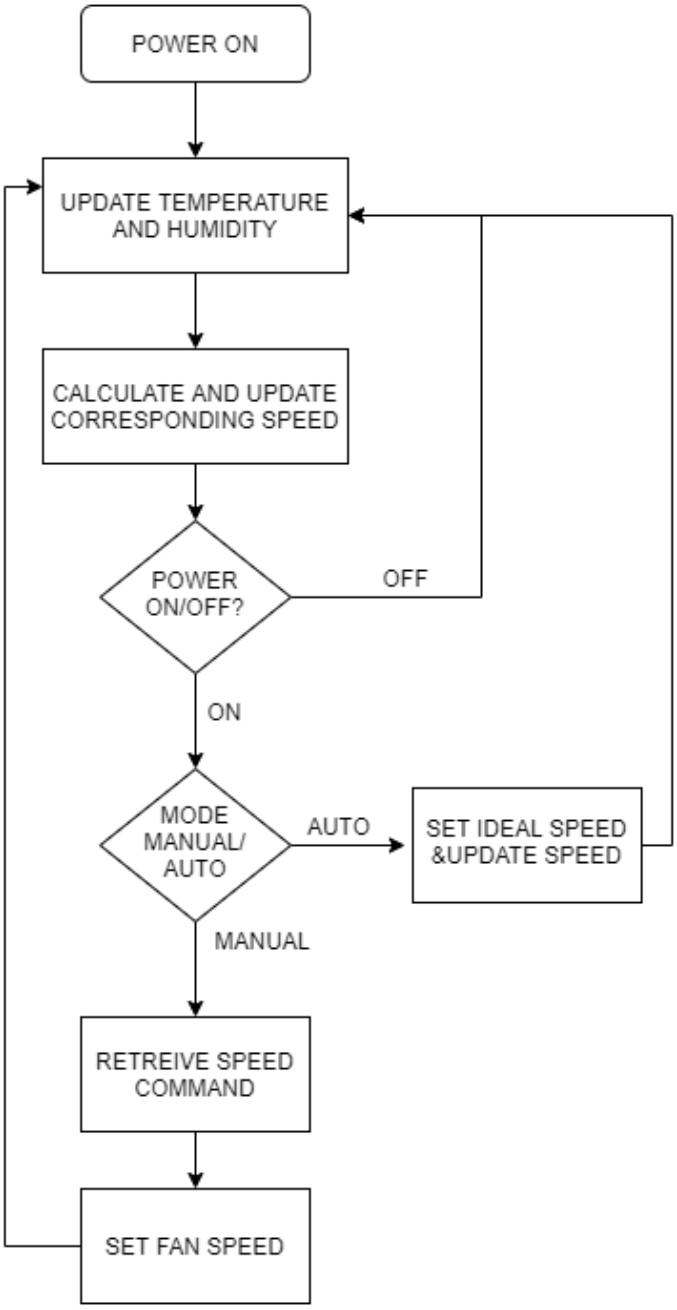


Fig: Flowchart of MCU Working

III. Calculates and updates the optimal speed for the measured temperature and humidity values.

IV. Check whether the switch state is ON/OFF. If the switch is OFF, then the execution goes again to the beginning, that is it again starts sensing temperature and humidity values. Otherwise, it proceeds to the next step

V. Checks whether the mode is AUTO/MANUAL. If the mode is AUTO, then the execution goes to the programme of setting the speed by generating required PWM signal. If the mode is MANUAL, the programme goes to the next step of execution.

VI. Reads the input speed level updated by the user in the real-time database. Now, the execution goes to the programme of setting the speed by generating required PWM signal.

VII. Continuously repeat from I.

3.3 CIRCUIT DIAGRAM

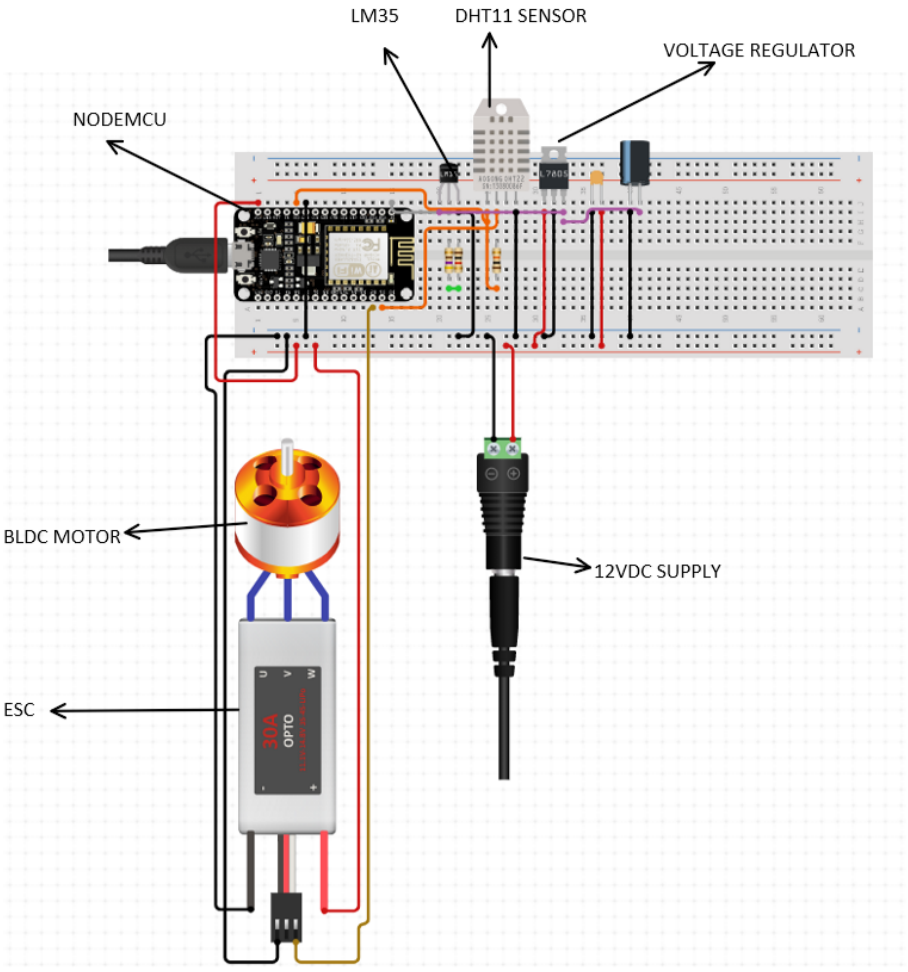


Fig: Circuit Diagram

The proper circuit diagram with all the explanations can be found here [How to wire DHT22/11 Humidity and Temperature Sensor, A2212 Brushless Motor 1000KV (with 30A ESC), LM35 to Node MCU (circuito.io)]

Circuit Diagram Explanation:

I. A 12V DC supply is taken from a constant DC source. In general, the homes are supplied with single phase or three phase AC which is converted into DC and stepped down to the required value.
II. The LM35 sensor and the DHT sensor both require Vcc of 5V. This is achieved by using a voltage regulator (L7805) which steps down the 12V supply to constant 5V supply.
III. The ESC is provided with 12V supply and the signal input is given from NodeMCU.
IV. The NodeMCU can also be powered from the 5V generated by the ESC or the voltage regulator.
V. The three phase output of the ESC is supplied to the BLDC motor. A heatsink is also provided to eliminate the heat produced.

The list of components used are tabulated in the table given below

PART	QUANTITY
NodeMCU 1.0v	1
A2212 Brushless Motor 1000KV	1
30A ESC	1
LM35 - Temperature Sensor	1
470 Ohm Resistor	1
1K Ohm Resistor	1
Female DC power adapter - 2.mm jack to screw terminal block	1
DHT11 Humidity and Temperature Sensor	1
10K Ohm Resistor	1
USB micro-B Cable	1
Voltage Regulator 5V	1
Capacitor Ceramic 100nF	1
Electrolytic Capacitor - 1uF/50V	1
BreadBoard	1
HeatSink T0-220	1
Jumper Wires Pack - M/M	2

CHAPTER 4

4.1 EXPECTED RESULTS

The speed control of the fan is achieved in two modes namely Manual and Auto. In Auto mode, the speed of the fan is automatically switched by the speed controller and it updates the current speed of the fan to the app. In case of Manual mode, the speed of the fan is as given by the user. The MCU constantly monitors the changes in the real-time database and acts accordingly by updating the ESC with the appropriate speed levels. The following table shows the expected results delivered over a span of 12 hours.

Table: Results showing comparison of Auto mode and Manual mode at different conditions

Switch State	T(˚C)	Humidity (%)	Mode	Ideal Speed LEVEL (Calculated by the MCU)	Speed LEVEL (Input by User)	Fan Speed LEVEL	Comment
OFF	24	94	MANUAL	4	0	0	Fan is in OFF state
OFF	24	94	MANUAL	4	0	0	Fan is in OFF state
ON	25	89	MANUAL	4	1	1	Running at User given speed
ON	26	83	MANUAL	4	1	1	Running at User given speed
ON	28	74	MANUAL	3	3	3	Running at User given speed
ON	28	70	MANUAL	3	3	3	Running at User defined speed
ON	31	55	AUTO	5	-	5	Fan speed is switched to 5
ON	32	49	AUTO	4	-	4	No change
ON	31	33	AUTO	4	-	4	No change
ON	29	40	AUTO	2	-	2	Fan speed is switched to 2
ON	27	45	AUTO	3	-	3	Fan speed is switched to 3
ON	26	54	AUTO	3	-	3	No Change

T- Room Temperature in Celsius Scale

All the data of temperature, humidity, ideal speed and the fan speed are saved in a csv file through google sheets. The csv file is uploaded into MATLAB and the simulation is processed

on the three phase BLDC model in simulink. The following MATLAB simulation shows the running of the BLDC fan with the comparison between the ideal and the manual speed.

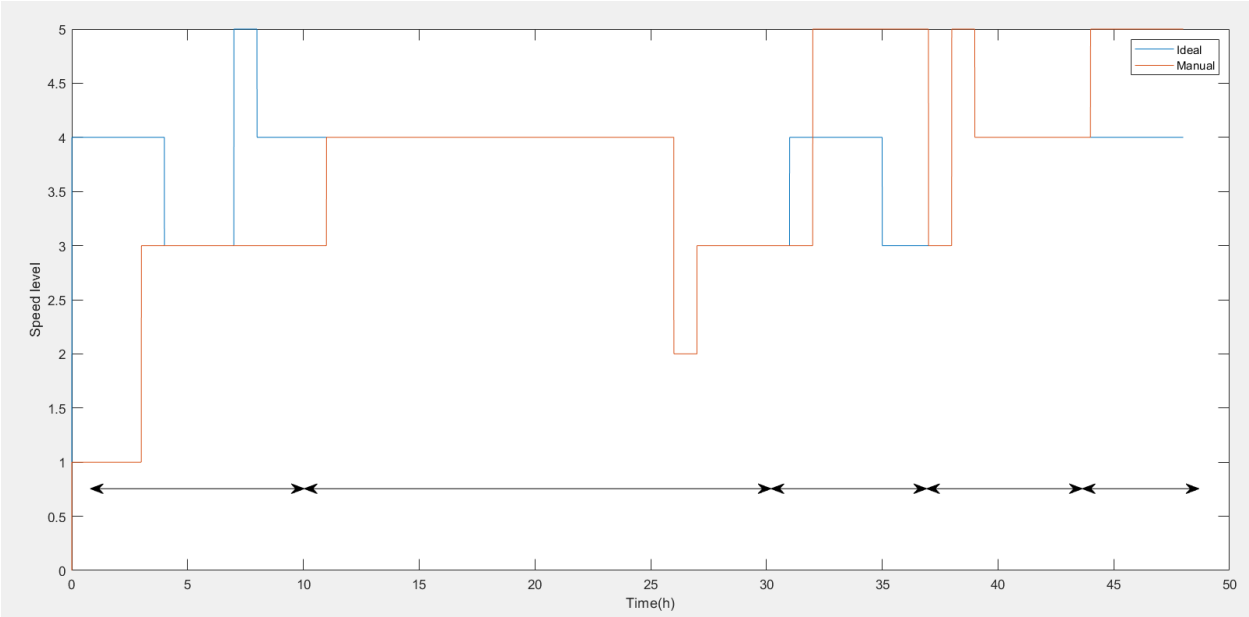


Fig: MATLAB simulation Result showing comparing ideal speed and manual speed

CHAPTER 5

5.1 APPLICATIONS

- Automatic airflow control in room results in comfort to the people inside the room
- The automatic airflow control can be integrated with Air Conditioners thus providing better cooling effect and helps in cooling the room real quick by switching the fan OFF and ON several times while AC is running.
- Integrating with Bluetooth Technology helps in switching ON and OFF the fan automatically. Also, this helps in alerting the user whenever one is leaving the room without switching OFF the fan/ switches ON the fan automatically when the user's mobile bluetooth is in the range.
- Low-cost solution for automatic airflow in the room that can be integrated with any fan.

5.2 CONCLUSION

Automatic fan speed control is achieved by adding this simple device to any fan. Although the project is proposed for BLDC fans which are being extensively used nowadays, the project can be extended to the existing Induction Motor fans. For the fans with the existing IoT support, the extra feature of temperature-controlled air flow can be incorporated easily by making use of the proposed solution. The existing electronic speed controllers can also be easily integrated with the project. The automatic air flow control in the room increases the comfort and also saves time in cooling the room in case of AC rooms.

REFERENCES

[1]. Sahid P.C et al. "Energy Efficient Ceiling Fan using BLDC Motor". International Journal of Engineering Research & Technology, ISSN: 2278-0181, Vol. 4 Issue 04, April-2015

[2]. DR. Suchart Yammen, Dr. Sureerat Tang and Mahesh Kumar Reddy Vennapusa. "IoT based speed control of Smart Fan". The 4th international conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON), 18613641.

[3]. Md. Abdullah-Al-Mashud et al. “Automatic Room Temperature Controlled Fan Speed Controller Using PT-100”. International Journal of Scientific and Engineering Research, August 2015.

[4]. Sonam Khedkar, Swapnil Thube. "Real Time Databases for Applications". International Research Journal of Engineering and Technology, Volume:04, Issue:06, June-2017, e-ISSN:2394-0056.

[5]. I S Siva Rao et al. "Shop GO : An IoT based solution for smart shopping". In 2020 International Conference on Computer Science, Engineering and Applications.

[6]. Naveen Bevara, Sanjay Dixit. "Speed-Control Techniques in AC-DC Operated BLDC Applications". Texas Instruments, Application Report, SLO, A203-August 2016