RV COLLEGE OF ENGINEERING®
BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

"SMART ROOM"

SELF STUDY REPORT
MICROCONTROLLERS AND EMBEDDED SYSTEM (18CS44)
IV SEMESTER

2020-21

Submitted by

MYTHRI NAIK	1RV19CS094
NAVNITH BHARADWAJ	1RV19CS098
NIDHI GK	1RV19CS100
NIKHIL VISHWANATH HEGDE	1RV19CS101
PURNODEEP RAJANKAR	1RV19CS122

Under the Guidance of
Dr. BADARINATH KB
Department of CSE, RVCE,
Bengaluru - 560059

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the **SELF STUDY-**project work titled “SMART ROOM” has been carried out by **MYTHRI NAIK (1RV19CS094), NAVNITH BHARADWAJ (1RV19CS098) , NIDHI GK (1RV19CS100), NIKHIL VISHWANATH HEGDE (1RV19CS101), PURNODEEP RAJANKAR (1RV19CS122)** bona fide students of RV College of Engineering, Bengaluru, have submitted in partial fulfillment for the **Assessment of Course: MICROCONTROLLERS AND EMBEDDED SYSTEMS (18SC44)** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report.

Dr. BADARINATH KB
Department of CSE,
RVCE., Bengaluru –59

Head of Department
Department of CSE,
RVCE, Bengaluru–59

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059

(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **MYTHRI NAIK (1RV19CS094), NAVNITH BHARADWAJ (1RV19CS098), NIDHI GK (1RV19CS100), NIKHIL VISHWANATH HEGDE (1RV19CS101), PURNODEEP RAJANKAR (1RV19CS122)** the students of 4th Semester B.E., Department of Computer Science and Engineering, RV College of Engineering, Bengaluru hereby declare that the Mini-Project titled **"SMART ROOM"** has been carried out by us and submitted in partial fulfillment for the **Assessment of Course: MICROCONTROLLERS AND EMBEDDED SYSTEMS** during the year 2020-2021.

Place: Bengaluru

Date:

INTRODUCTION

In the present era of globalization, automation has been making major strides towards making lives more comfortable all around the globe. With the technological advancements in various fronts, home automation has become a major area of interest. With this in mind, we aim to provide the facility of-

- A smart lighting system that ensures automatic illumination with color and brightness control options to suit any occasion.
- A smart temperature based speed control system that automatically adjusts the speed of the fan according to the temperature of the room

WHY AUTOMATE

- A long life span - economic and easy to handle
- You choose - switch on or switch off the lights from any part of your home
- Safe and secure homes
- Less manual work - Helpful for differently abled people
- Suitable for both offices and houses
- Change the color of the lights based on your mood and aesthetic preferences.
- Learn about the current temperature details - decide whether to use the automatic mode or the manual mode.

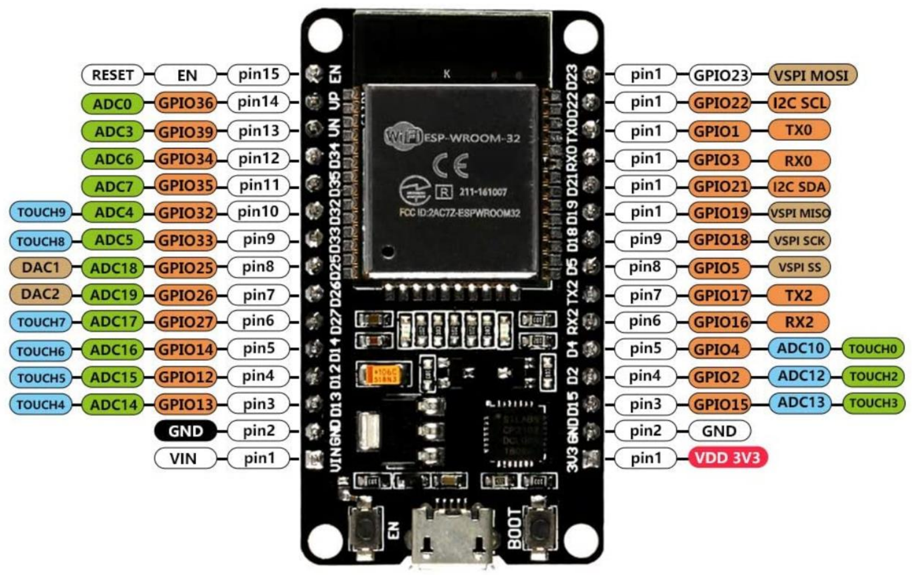
MICROCONTROLLER USED: ESP-32

Whenever a project requires WiFi connectivity, ESP-32 is the better choice, for the following reasons:

- The ESP32 is a very feature centric development board
- The microprocessor used is **Tensilica Xtensa 32-bit LX6** with clock frequency up to 240MHz and performs up to 600 DMIPS
- Both single and dual core variants are available
- The microprocessor of ESP-32 has got a 32 bit dual core CPU
- ESP-32 is cost-effective
- ESP-32 has got 38 programmable pins
- The clock speed being 240kHz, making it faster than the alternatives

COMPONENTS REQUIRED

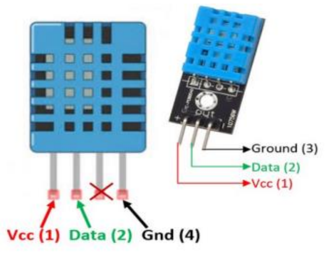
a. **Microcontroller (ESP32)**



b. **Connecting Wires**

c. **Relays**: Relay is used to connect our microcontroller to the appliances i.e lights. Connection is done through relay driver IC ULN 2003 as we want to use a low voltage circuit to control appliances which are connected to 220 V mains supply.

d. **Digital temperature and Humidity sensor (DHT11) :**

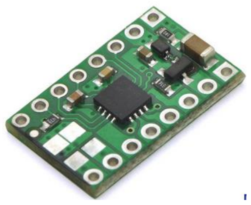


The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. Operating Voltage: 3.5V to 5.5V. Temperature Range: 0°C to 50°C. Resolution: Temperature and Humidity both are 16-bit. Accuracy: ±1°C.

e. **Light Dependant Resistors**

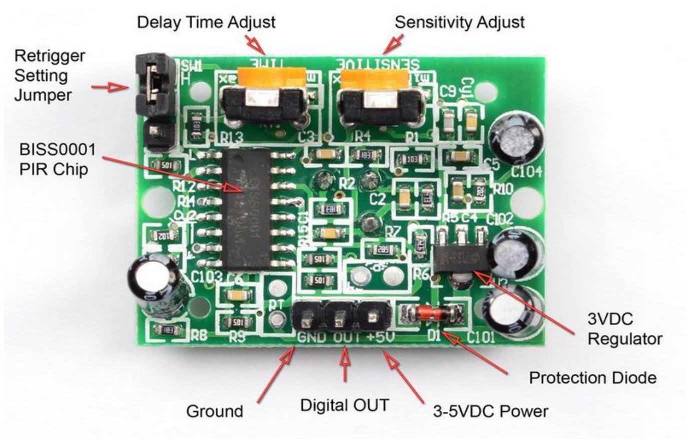
f. **DRV10293 - Motor Driver**

Motor Driver acts as an interface between the appliance i.e. fan and the control circuits which is the microcontroller. The function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor.



g. **Passive Infrared sensor**

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensor's range. They are small, inexpensive, low-power, easy to use and don't wear out. Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected) is the output obtained. Detection range is up to 20 feet (6 meters) 110° x 70° detection range. 5V-12V input voltage.



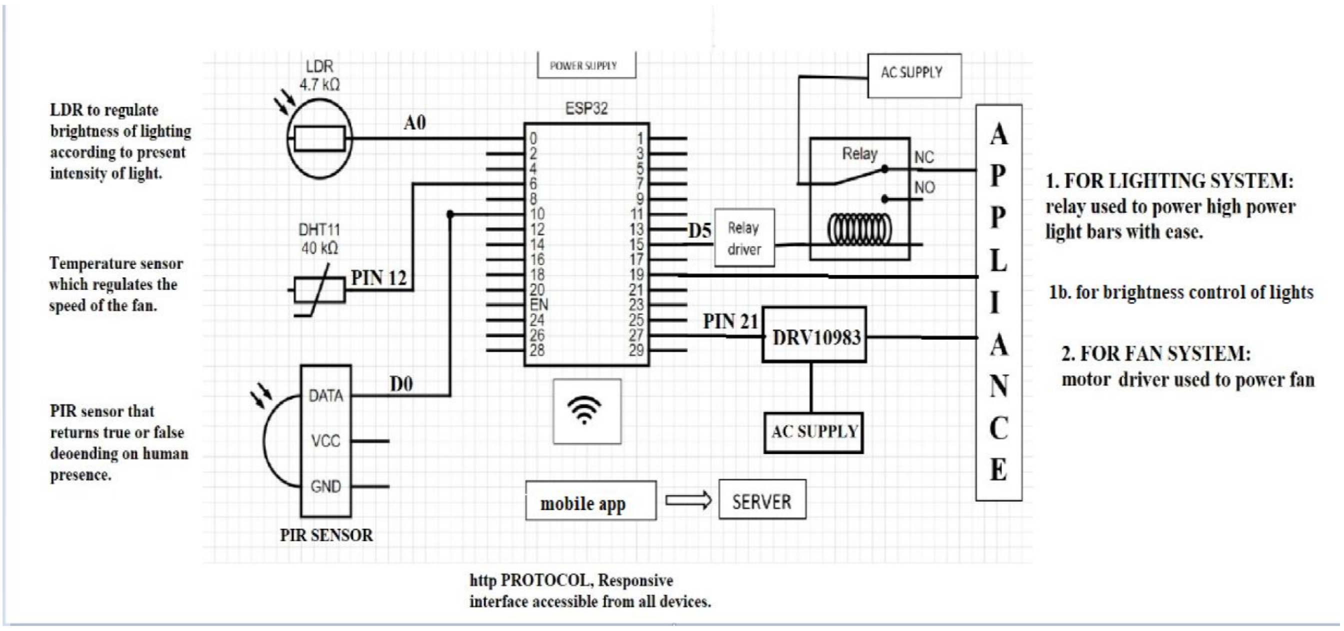
h. **12V adapter**

The adapter is used for power supply for components like relay driver and motor driver.

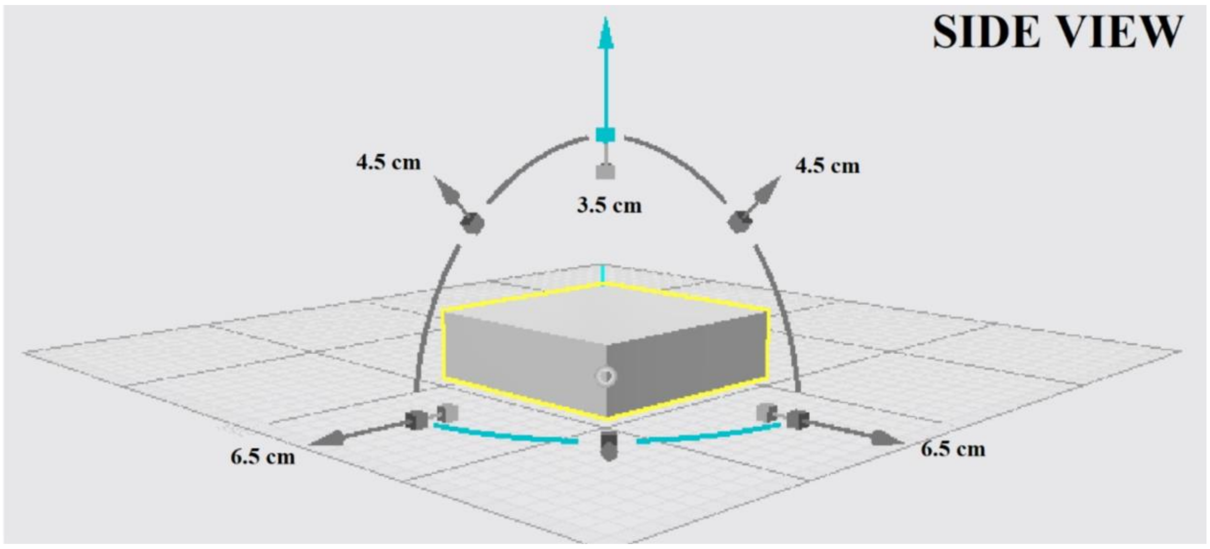
DESCRIPTION OF WORKING

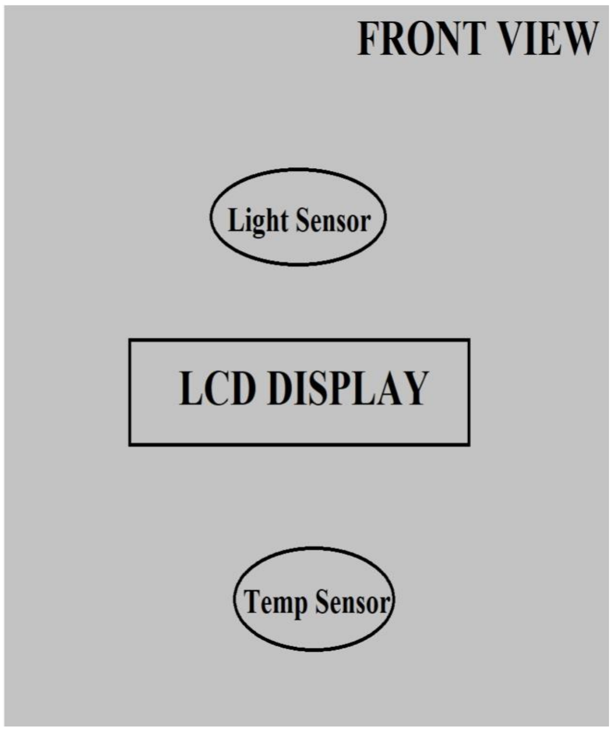
The Microcontroller along with the PIR sensor facilitates the lights and fans in a room turning on automatically on recognizing human presence in the room. Further, the temperature sensor takes into account the present temperature and humidity in the room via which the speed of the fan is regulated to maintain optimum temperature at all times. Through the readings of the LDRs in place, the brightness of the lights are adjusted according to the time of day and present intensity of light in the room.
A centralized Microcontroller system is used for several rooms grouped together while the sensors are present in every room. The PIR sensor returns a true or false value based on the presence of individuals in the room. Once presence is confirmed, the LDR sensor measures the present light intensity of the room and is related to the brightness of the lighting system present in inverse proportion. So the light intensity produced by the smart light is less in the morning and high during the evening. The DHT11 sensor gives the temperature and humidity readings. The speed of the fan is adjusted according to these. The current scale used to map readings to speed of the fan is based on the study of the average temperature of Bangalore city across years.
We also provide a web interface via which the user can switch between use and manual modes and can also adjust the hue of light by controlling RGB values. As a future scope, we plan to store the temperature and light intensity data of the user in the ThingSpeak cloud and further use this data to predict future electricity usage and hence help the user to efficiently reduce electricity usage.

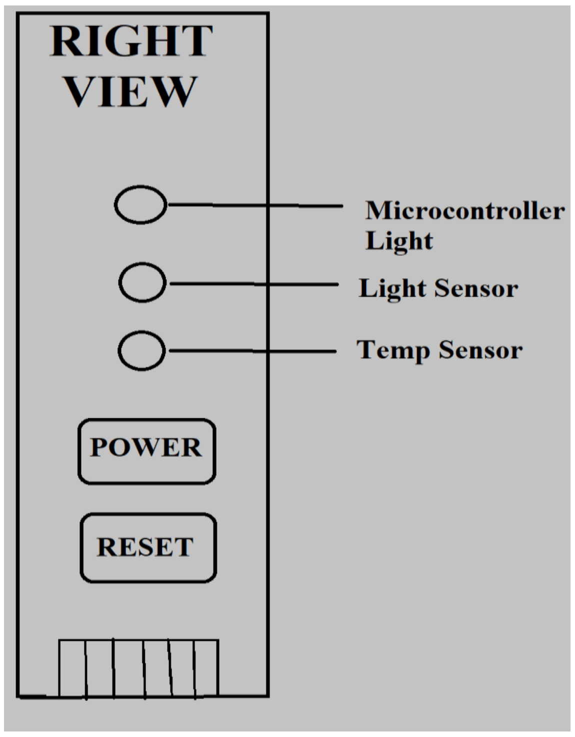
BLOCK DIAGRAM


LDR to regulate brightness of lighting according to present intensity of light.
Temperature sensor which regulates the speed of the fan.
PIR sensor that returns true or false deoending on human presence.
LDR 4.7 kΩ
A0
DHT11 40 kΩ
PIN 12
DATA
VCC
GND
D0
PIR SENSOR
POWER SUPPLY
ESP32
EN
D5
Relay driver
Relay
NC
NO
AC SUPPLY
PIN 21
DRV10983
AC SUPPLY
mobile app
SERVER
APPLIANCE
1. FOR LIGHTING SYSTEM: relay used to power high power light bars with ease.
1b. for brightness control of lights
2. FOR FAN SYSTEM: motor driver used to power fan
http PROTOCOL, Responsive interface accessible from all devices.

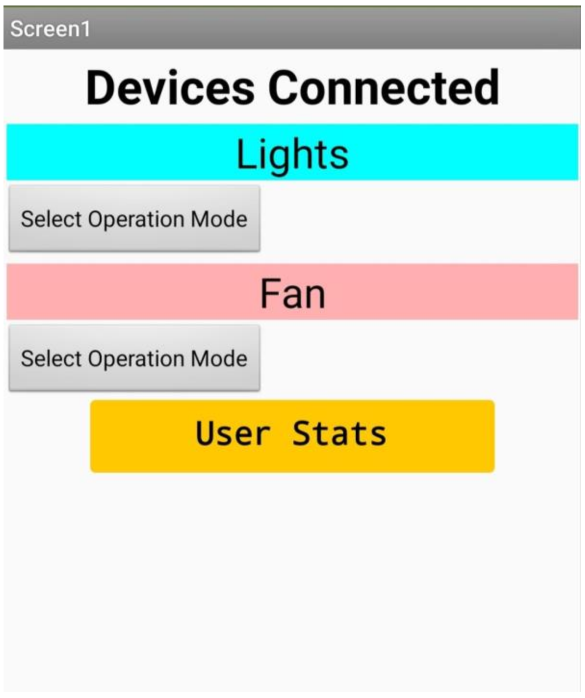
3-D PRODUCT VIEW



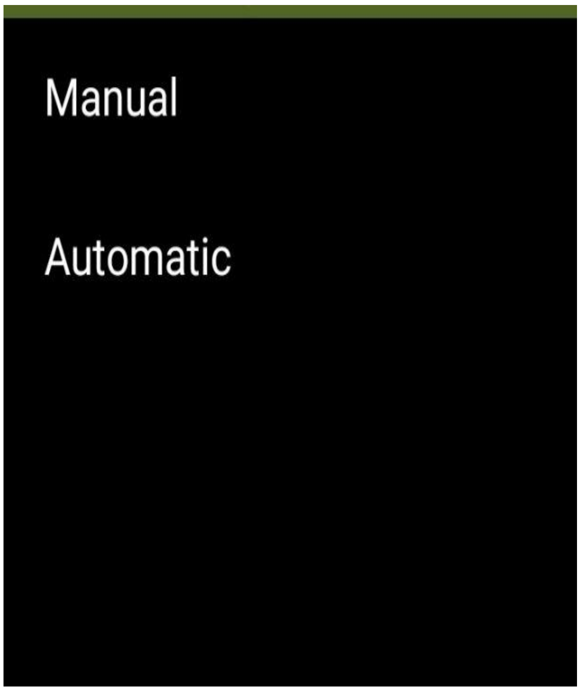




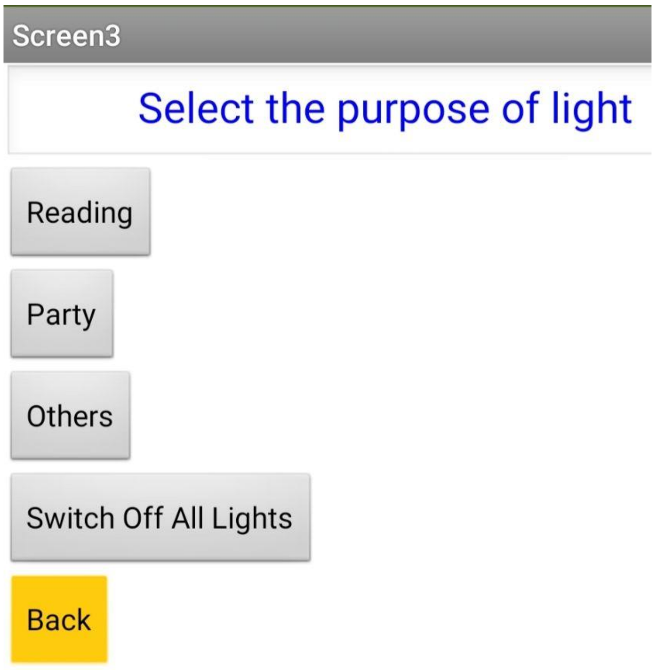
APP INTERFACE WITH THINGSPEAK ANALYTICS



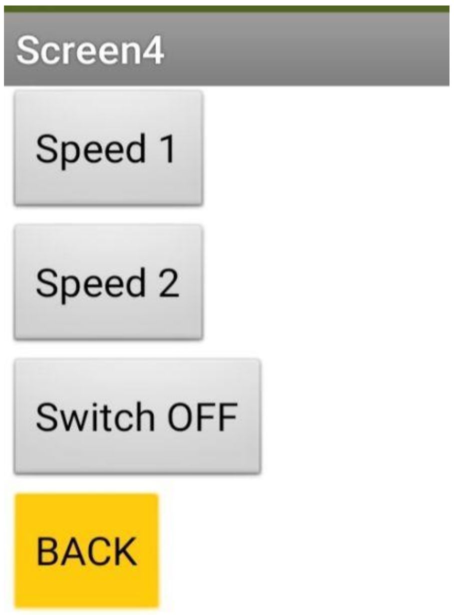
- This is the Home Page of the app.
- Shows the devices connected and navigation to their operating modes can be done through buttons "Select Operation Mode"
- "User Stats" is a button to navigate to usage graph window



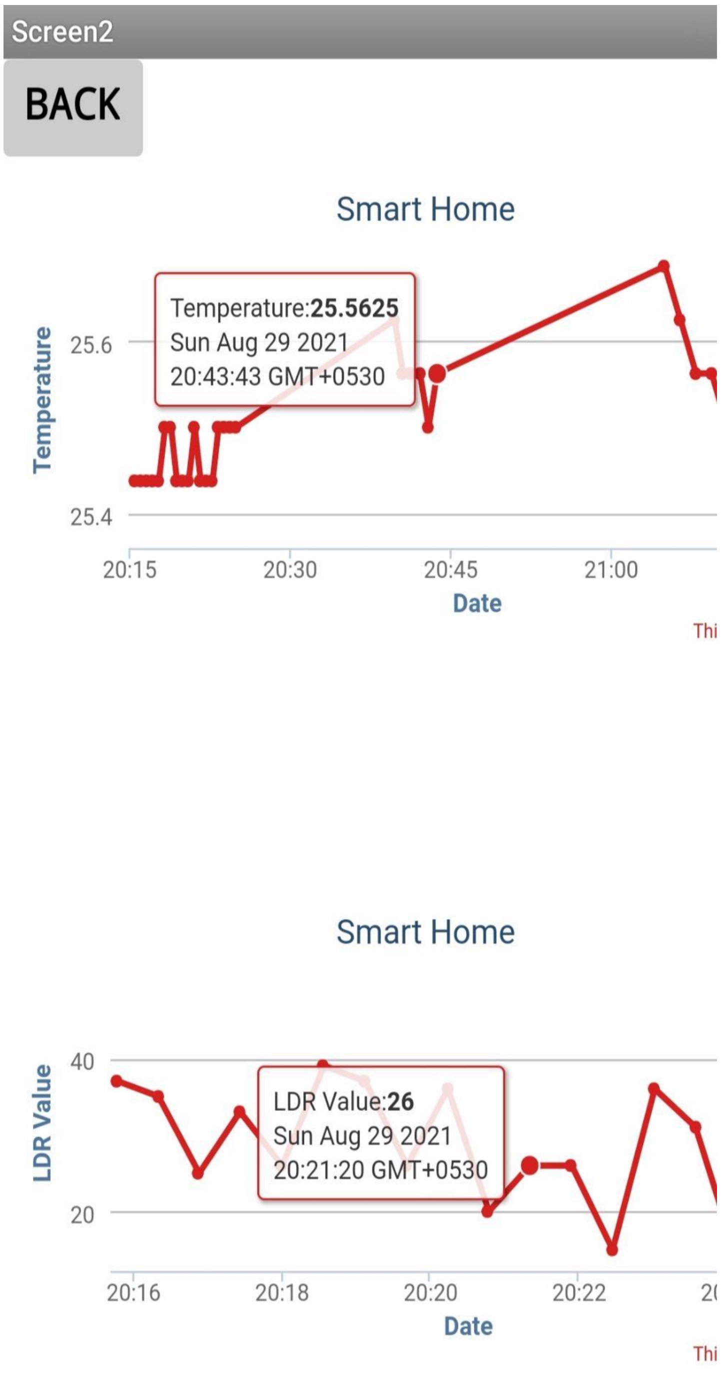
- This is the window that enables the user to select the operation mode for the devices.
- This control is provided separately for both fans and lights
- Based on the user selection the device operates in corresponding mode.



- When the user selects manual as the operation mode, control is navigated to this window.
- Here the user can select which light the user wants to use.
- This provides more flexibility along with the fully automatic mode.



- This page gives user to control the fan speeds when manual mode is selected.
- Two speeds are provided.
- In automatic mode the speeds are automatically decided based on



- This is the User Statistics page.
- First graph shows the temperature recorded and different intervals of time.
- Second graph shows the ldr values and different time-stamps

REFERENCES

- Speed-Control Techniques in AC-DC Operated BLDC Applications, Texas Instruments.
- https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf
- https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_client.html

APPENDIX: SOURCE CODE

```
#include <WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ThingSpeak.h>

#define DS18B20PIN 16

#define CHANNEL_ID 1490374
#define WRITE_API_KEY "E4NPOX101H66IZE5"
#define READ_API_KEY "AIOTW3N310SUU94Q"

WiFiClient client;

OneWire oneWire(DS18B20PIN);

DallasTemperature sensor(&oneWire);
const char* ssid = "TP-Link_5A94";
const char* password =  "10977798";

unsigned int lightField = 3;
unsigned int fanField = 4;

int pir_data = 10;
int ldr = 36;
int lightInit;  // initial value
int lightVal;   // light reading
float tempinC;

int led1 = 2;
int led2 = 23;
int led_reading = 22;
int led_party = 19;

int led3 = 21;
int led4 = 4;

void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);

  WiFi.begin(ssid, password);
  ThingSpeak.begin(client);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("WiFi connected with IP: ");
  Serial.println(WiFi.localIP());

  pinMode(pir_data, INPUT);
  pinMode(ldr , INPUT);
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led_reading, OUTPUT);
  pinMode(led_party, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  lightInit = analogRead(A0);
  Serial.print("light sensor init = ");
  Serial.print(lightInit, DEC);
  Serial.println("");
  sensor.begin();
}

float readTSData( long TSChannel, unsigned int TSField ) {

  float data =  ThingSpeak.readFloatField( TSChannel, TSField, READ_API_KEY );
  //  Serial.println( " Data read from ThingSpeak: " + String( data, 9 ) );
  return data;

}

void loop() {
  delay(40000);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
}

float lightFlag = readTSData(CHANNEL_ID, lightField);
delay(15000);
float fanFlag = readTSData(CHANNEL_ID, fanField);
delay(15000);

//pir sensor *
bool PIR_status = digitalRead(pir_data);

if (lightFlag == 0){
  if (PIR_status) {
    Serial.println("PIR motion detected");
    digitalWrite(led1, HIGH);

    //ldr sensor***
    lightVal = ldrSensor();
    Serial.print("light sensor = ");
    Serial.print(lightVal - lightInit);
    Serial.println("");
}
else {
    Serial.println("NO PIR motion detected");
    digitalWrite(led1, LOW);
    Serial.println("");
  }
ThingSpeak.writeField(CHANNEL_ID, 2, lightVal - lightInit, WRITE_API_KEY);
delay(15000);
}
else if(lightFlag == 1){
  Serial.println("Manual: reading");
  digitalWrite(led2, LOW);
  digitalWrite(led_party, LOW);
  digitalWrite(led_reading, HIGH);
  }
else if(lightFlag == 2){
  Serial.println("Manual: party");
  digitalWrite(led_reading, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led_party, HIGH);
  }
else if(lightFlag == 3){
  Serial.println("Manual: other");
  digitalWrite(led_reading, LOW);
  digitalWrite(led_party, LOW);
  digitalWrite(led2, HIGH);
  }
else{
  Serial.println("Manual: switch off");
  digitalWrite(led_reading, LOW);
```

```
    digitalWrite(led_party, LOW);
    digitalWrite(led2, LOW);
    }

  if(fanFlag == 0){
    if (PIR_status) {
      Serial.println("PIR motion detected");
      digitalWrite(led1, HIGH);

      //temperature sensor **
      tempinC = temperatureSensor();
      Serial.print("Temperature = ");
      Serial.print(tempinC);
      Serial.println("ºC");
      Serial.println("");
  }
  else {
      Serial.println("NO PIR motion detected");
      digitalWrite(led1, LOW);
      Serial.println("");
    }

  ThingSpeak.writeField(CHANNEL_ID, 1, tempinC, WRITE_API_KEY);
  delay(15000);
   }
   else if(fanFlag == 1){
    Serial.println("Manual: speed 1");
      digitalWrite(led4, LOW);
        digitalWrite(led3, HIGH);
    }
    else if(fanFlag == 2){
      Serial.println("Manual: speed 2");
      digitalWrite(led3, LOW);
      digitalWrite(led4, HIGH);
    }
    else{
      Serial.println("Manual: switch off");
      digitalWrite(led3, LOW);
      digitalWrite(led4, LOW);
    }
}

float temperatureSensor(){
      sensor.requestTemperatures();
      float tempinC1 = sensor.getTempCByIndex(0);
      if (tempinC1 > 28 && tempinC1 < 32) {
        digitalWrite(led4, LOW);
        digitalWrite(led3, HIGH);
      }
      else if (tempinC1 >= 32) {
```

```
      digitalWrite(led3, LOW);
      digitalWrite(led4, HIGH);
    }
    else {
      digitalWrite(led3, LOW);
      digitalWrite(led4, LOW);
    }
    return tempinC1;
 }

 int ldrSensor(){
    int lightVal1 = analogRead(A0);
    if (lightVal1 - lightInit < 100) {
      Serial.println("Bright");
      digitalWrite(led2, LOW);
    }
    else {
      Serial.println("Dark");
      digitalWrite(led2, HIGH);
    }
  return lightVal1;
}
```