Home Automation Using Arduino and ESP8266

Samir Samanta, Koushik Kr. Khan, Arghya Bhattacharyya, Sounak Das, Atul Barman, Mr. Koushick Mathur

University Institute of Technology, The University of Burdwan, Golapbag (North), Burdwan, West Bengal, India

Abstract— *With the advent of technology, life has become faster in pace and shorter in interactions, with others, as well as with the surroundings. In such a scenario, there is a need to have an endeavor to have everything at the push of a button away, and more importantly, automated. Home Automation is such an endeavor, in which, all the electrical appliances present at home are connected to each other, having interactions with sensors placed at strategic positions in a closed loop manner in order to perform meager tasks automatically, leaving less burden on the humans. With this project we are promoting the fact that Home Automation can greatly contribute to energy conservation too.*

Keywords— Home Automation, ESP8266, sensors.

I. INTRODUCTION

A brief definition of Home Automation may be given in this way as, "Home Automation is the technology, in which every electrical appliance present inside a particular house are connected to one another and to a set of "*sensors*" placed at particular strategic positions, reading specific data in a closed loop fashion to serve the purpose to automate all the connected home appliances."

The most important part in a fully automated system are the sensors, be it [1]IR motion sensors, heat sensors, smoke detectors. The data they acquire are then sent to the microcontroller unit, which then processes the data and performs specific switching of the Home Appliances in a real time fashion.

In this project we have set these specific objectives for our version of the automated system.

1. Turn ON all the appliances when a person enters the room.
2. He/They can then turn them ON or OFF using an IR remote, or use a web application for the purpose as convenient.
3. Turn off everything whenever everyone leaves the room, in other words, when there are no one inside.
4. A person can remotely access the home appliances if he feels the need be.

In addition to these there are many more possibilities that we can explore with Home Automation systems.

We are using [2]Arduino Uno development board as the brain or the microcontroller unit of the system, [3]ESP8266 as the World Wide Web interface and IR sensors to count the number of people entering or leaving the room.

1.1Arduino Uno

According to the description given in the Arduino Website, "Arduino/Genuino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button."

Fig.1.1: Arduino Uno Development Board

Arduino is an open source project aiming at developing greater curiosity towards DIY projects among students and enthusiasts. Along with the development board, comes the Arduino IDE based on a programming language called *Processing* for programming the microcontroller.

1.2 ESP8266 Wi-Fi Module

The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and microcontroller capability produced by Espressif. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using AT commands.

The very low price and the fact that there were very little external components on the module which suggests that it could eventually be very inexpensive in volume make it the component of choice for our needs.

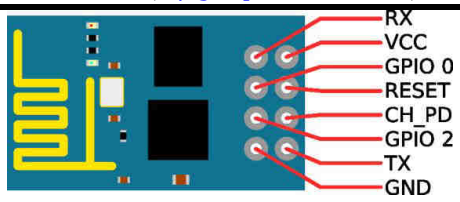

Fig.1.2: Pin diagram of ESP8266

1.2.1 AT (Hayes) Commands

The **Hayes command set** is a specific command language originally developed by Dennis Hayes for the Hayes Smartmodem 300 baud modem in 1981. The command set consists of a series of short text strings which can be combined to produce commands for operations such as dialing, hanging up, and changing the parameters of the connection. The vast majority of dial-up modems use the Hayes command set in numerous variations.

[4] Some AT Commands used in our project with the ESP8266 and there functions

AT Command	Response	Description
AT	OK	
AT+RST	Description about the device followed bt "OK"	To reset the module and refresh all the settings.
AT+GMR	Description about the device followed by "OK"	
AT+CWMODE=<1,2,3>	OK	1. The module acts as a Wi-Fi enabled device that must be connected to a router. 2. Acts as a Wi-Fi Hotspot 3. Acts as both
AT+CWLAP	Information about every Wi-Fi router present in the vicinity separated by newlines.	To get to know if any AP is present so that to connect to it. Note: Doesn't work in Mode 1.
AT+CWJAP=<SSID>,<password>	WIFI CONNECTED WIFI GOT IP OK	To connect to a specific access point. Note: Doesn't work in Mode 1.
AT+CIOBAUD=<baud rate>	OK	To set the Baud rate of the module in accordance of the baud rate of the microcontroller with which it is used.
AT+CIPMUX=1	OK	To use the module in multiplexed mode.
AT+CIPSERVER=1,80	OK	To get the message from port number 80.

1.3 IR Sensors

We are using TSOP1738 IR sensor. The TSOP 1738 is a member of IR remote control receiver series. This IR sensor module consists of a PIN diode and a pre amplifier which are embedded into a single package. The output of TSOP is active low and it gives +5V in off state.

When IR waves, from a source, with a centre frequency of 38 kHz incident on it, its output goes low. TSOP module has an inbuilt control circuit for amplifying the coded pulses from the IR transmitter. A signal is generated when PIN photodiode receives the signals. This input signal is received by an automatic gain control (AGC). For a range of inputs, the output is fed back to AGC in order to adjust the gain to a suitable level. The signal from AGC is passed to a band pass filter to filter undesired frequencies. After this, the signal goes to a demodulator and this demodulated output drives an NPN transistor. The collector output of the transistor is obtained at Pin 3 of TSOP module.

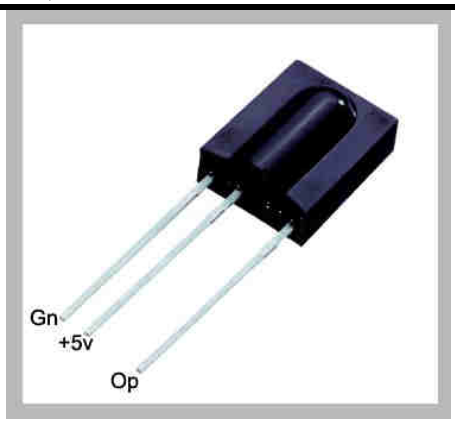

Fig.1.3: TSOP1738 Pin Diagram

II. SCHEMATICS AND CODES

2.1The IR Sensor Circuit

To make use of the IR sensors, we have placed two IR LEDs at each side of the door, i.e., one is kept on the inner side and the other on the outer side. Along with them the two TSOPs are placed opposite to them for detection of presence.

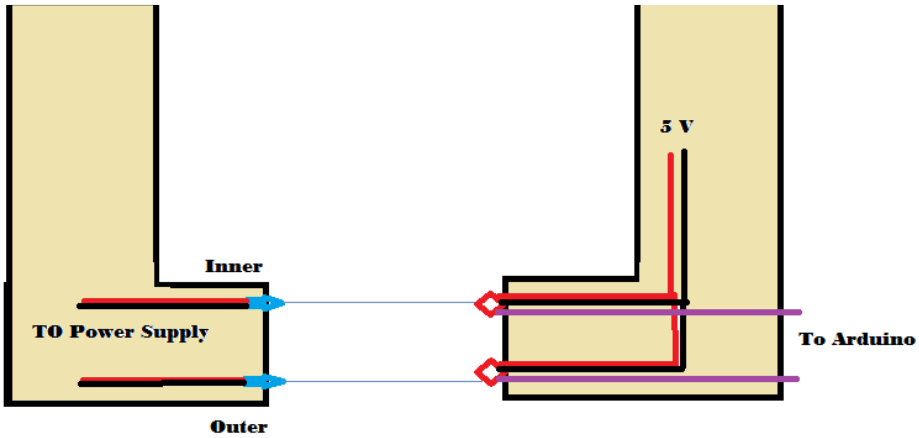

Fig.2.1: The Sensors Positioned At The Door

2.1.1Transmitter Part

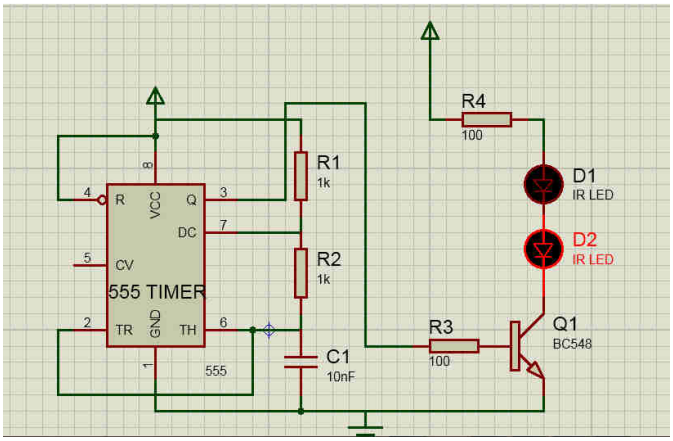

Fig.2.2: 38kHz IR Transmitter

We have used a 555 astable multivibrator to produce near about 38kHz output and used it drive two IR LEDs as shown in the above figure. These LEDs would then go about to be used as the 38kHz IR sources for the two TSOPs to work with.

2.2.2 Reciever Part

The TSOPs need a power source of about 5V which can be provided by the in built power source present in the UNO board. The outputs of the TSOPs are analog values, which are then connected to two of the six analog inputs present in the Arduino Board.

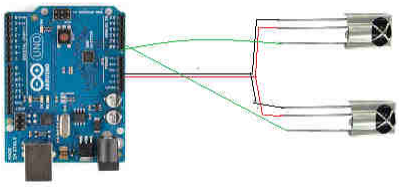

Fig.2.3: TSOP1738 Connected To Arduino Uno

The analog input to the Arduino is decoded into an integer value between 0-1024. Based on that we can judge if a particular set of sensors has been triggered or not.

2.2 ESP8266 Interfacing To Arduino

Connecting an ESP8266 Wi-Fi module to the Arduino is quite simple. We just need to connect the Tx of ESP to the Rx of Arduino and vice-versa. There are hardware serial ports in Arduino, marked Tx and Rx but we have used the *softwareserial* header file to make 2 and 3 as Tx and Rx for our purposes.

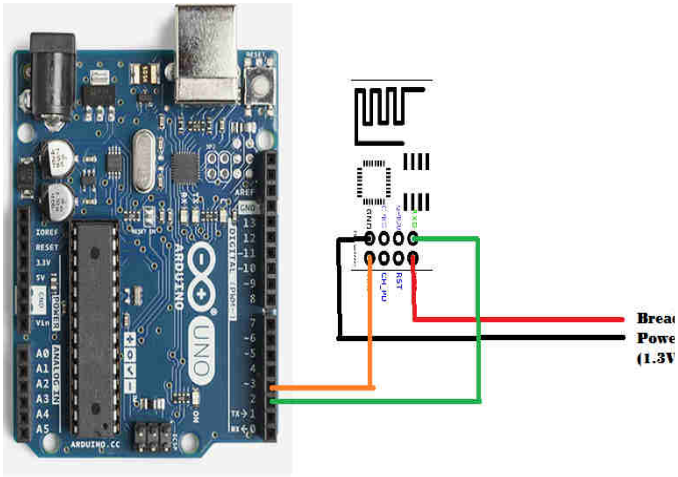

Fig.2.4: ESP Connected To Arduino Uno

The ESP takes about 300mA current. So, we had to use a generic Breadboard Power supply, which has a 12V input and can provide 12V,5V and 1.3V needed for the ESP8266.

2.3 Driving Electrical Appliances

To drive high voltage electrical appliances, we have used 12V relays.

2.4 Calculating the Number Of Persons Inside The Room

Let us assume two variables IN and OUT for the respective sensors. IN is providing a fictional value 0 when no one is obstructing its source an1 otherwise and same for out. Then if OUT=1 given that IN was already 1 then its understood that the person is going outside and thus a counter is decremented accordingly. And vice versa is applicable for a person entering the premises.

Following is the Arduino Code used in our project.

```
int c=0;
int inner=0;
int outer=0;
int A=0,B=0;
int fa=0,fb=0;
void setup() {
 pinMode(4,OUTPUT);
 pinMode(5,OUTPUT);
 pinMode(6,OUTPUT);
 Serial.begin(9600);
}
void loop() {
  inner=analogRead(A4);
outer=analogRead(A5);
if(inner<500)
{
  B=0;
 }
else
{
  B=1;
 }
if(outer<500)
{
  A=0;
}
else
{
  A=1;

}
if(A==0)
{
  fa=1;
  if(fb==1){c--;}
}
else
{
```

```
 fa=0;
}
if(B==0)
{
 fb=1;
 if(fa==1){c++;}
 }
 else
 {fb=0;
 }
 Serial.println(inner);
 Serial.println("\n");
 Serial.println(outer);
 Serial.println("\n\n");
 delay(500);

}
```

2.5 Controlling Arduino Pins Using Webpage

It should be noted that to use a webpage in the local domain or in the Worldwide Web, the ESP8266 and in turn the Arduino must be connected to the internet through a Wi-Fi enabled router. The ESP works only when we enter the AT commands for various functions like sending a message or receiving one. Thus, we make the Arduino do all those chores like opening up a link, connecting to the Access Point, etc..

The following Arduino sketch does just that

```
#include <SoftwareSerial.h>
#define DEBUG true
SoftwareSerial esp8266(2,3); // make RX Arduino line is pin 2, make TX Arduino line is pin 3.
                             // This means that you need to connect the TX line from the esp to the Arduino's pin 2
                             // and the RX line from the esp to the Arduino's pin 3
void setup()
{
 Serial.begin(9600);
 esp8266.begin(9600); // your esp's baud rate might be different
 pinMode(4,OUTPUT);
 digitalWrite(4,LOW);
 pinMode(5,OUTPUT);
 digitalWrite(5,LOW);
  pinMode(6,OUTPUT);
 digitalWrite(5,LOW);
 senddat("AT+RST\r\n",10000,DEBUG);
 senddat("AT+CWMODE=2\r\n",1000,DEBUG);          // configure as access point
 senddat("AT+CIFSR\r\n",1000,DEBUG); // get ip address
 senddat("AT+CIPMUX=1\r\n",1000,DEBUG);          // configure for multiple connections
 senddat("AT+CIPSERVER=1,80\r\n",1000,DEBUG);    // turn on server on port 80
}
 void loop()
{
 if(esp8266.available()) // check if the esp is sending a message
 {

   if(esp8266.find("+IPD,"))
   {
    delay(1000); // wait for the serial buffer to fill up (read all the serial data)
    // get the connection id so that we can then disconnect
    int connectionId = esp8266.read()-48; // subtract 48 because the read() function returns
                                          // the ASCII decimal value and 0 (the first decimal number) starts at 48
    esp8266.find("pin="); // advance cursor to "pin="
     int pinNumber = (esp8266.read()-48)*10;
    pinNumber+=(esp8266.read()-48);// get first number i.e. if the pin 13 then the 1st number is 1, then multiply to get 10
    control(pinNumber);// toggle pin
     // make close command
    String closeCommand = "AT+CIPCLOSE=";
    closeCommand+=connectionId; // append connection id
    closeCommand+="\r\n";
    senddat(closeCommand,1000,DEBUG);      // close connection
   }
 }
}
 /*
Function senddat is used to make the ESP8266 execute the required AT commands. It returns a string as prescribed in the previous table of AT commands and their responses.
*/
String senddat(String command, const int timeout, boolean debug)
{
   String response = "";
  esp8266.print(command); // send the read character to the esp8266
  long int time = millis();
  while( (time+timeout) > millis())
   {
    while(esp8266.available())
```

```
    {
     // The esp has data so display its output to the serial window
      char c = esp8266.read(); // read the next character.
      response+=c;
    }
  }
  if(debug)
  {
   Serial.print(response);
  }
  return response;
}

/*Function control, to do the required actions on the Arduino's pins when it gets the required responses from the exp8266 module through the serial wire.
*/
void control(int a)
{
 switch(a)
 {
  case 10:digitalWrite(4,!digitalRead(4));
  break;
  case 11:digitalWrite(5,!digitalRead(5));
  break;
  case 12:digitalWrite(6,!digitalRead(6));
  break;
  case 13:digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  delay(1000);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  delay(1000);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  delay(1000);
  break;
  default: break;

 }
}
```

2.6 [5]Controlling Arduino Through Generic IR Remote

We have also kept a provision to use an IR remote control for controlling the home appliances when need be.

```
#include <IRremote.h>
int IRpin = 11,pin3=0,pin4=0,pin2=0,pattern=0;
IRrecv irrecv(IRpin);
decode_results results;
void setup()
{
 pinMode(12,OUTPUT);
 pinMode(13,OUTPUT);
 pinMode(4,OUTPUT);
 pinMode(5,OUTPUT);
 pinMode(6,OUTPUT);
 digitalWrite(13,HIGH);
 digitalWrite(12,LOW);
 Serial.begin(9600);
 irrecv.enableIRIn(); // Start the receiver
}
void loop()
{
 if (irrecv.decode(&results))
   {
    Serial.println(results.value, DEC); // Print the Serial 'results.value'
     irrecv.resume();   // Receive the next value
   }
 switch(results.value)
 {
  case 33441975:pin4=1;

  break;
  case 33456255:pin4=0;
  break;
  case 33446055:pin3=1;
  break;
  case 33439935:pin3=0;
  break;
  case 33454215:pin2=1;
  break;
  case 33472575:pin2=0;
  break;
 default:break;
   }
 if(pin4==1)
  digitalWrite(4,HIGH);
 else
  digitalWrite(4,LOW);
 if(pin3==1)
  digitalWrite(5,HIGH);
 else
  digitalWrite(5,LOW);
  if(pin2==1)
  digitalWrite(6,HIGH);
```

```
 else
   digitalWrite(6,LOW);
 }
```

2.7 The Final Sketch Including Above Features

```
#include <SoftwareSerial.h>
#include <IRremote.h>
 #define DEBUG true
 SoftwareSerial esp8266(2,3); // make RX Arduino line is pin 2, make TX Arduino line is pin 3.
                              // This means that you need to connect the TX line from the esp to the Arduino's pin 2
 int c=0;
int inner=0;
int outer=0;
int A=0,B=0;
int fa=0,fb=0;
int pin=0;
int flag=1;
int flg=0;
int IRpin = 11,pin3=0,pin4=0,pin2=0,pattern=0;
IRrecv irrecv(IRpin);
decode_results results;
void setup()
{
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);
  digitalWrite(12,LOW);
  Serial.begin(9600);
  esp8266.begin(9600); // your esp's baud rate might be different
  pinMode(4,OUTPUT);
  digitalWrite(4,LOW);
   pinMode(5,OUTPUT);
  digitalWrite(5,LOW);
   pinMode(6,OUTPUT);
  digitalWrite(5,LOW);
  senddat("AT+RST\r\n",10000,DEBUG);
  senddat("AT+CWMODE=2\r\n",1000,DEBUG);          // configure as access point
  senddat("AT+CIFSR\r\n",1000,DEBUG); // get ip address
  senddat("AT+CIPMUX=1\r\n",1000,DEBUG);          // configure for multiple connections
  senddat("AT+CIPSERVER=1,80\r\n",1000,DEBUG);    // turn on server on port 80
  irrecv.enableIRIn();
}
 void loop()
{
   useWifi();
 useIR();
  inner=analogRead(A4);
outer=analogRead(A5);
if(inner<500)
{
  B=0;
 }
else
{
  B=1;
 }
if(outer<500)
{
  A=0;
}
else
{
  A=1;

}
if(A==0)
{
  fa=1;
  if(fb==1){c--;}
}
else
{
  fa=0;
}
if(B==0)
{
  fb=1;
  if(fa==1){c++;}
  }
  else
  {fb=0;
  }

  if(c>0)
  {
    if(flag == 1)
    {
      pin=1;
      control(pin);
    }
   }
  else
  {
    if(flg==0)
```

```
   {
   pin=0;
   control(pin);
   }
   flag=1;
  }
   Serial.println(c);
  }
void useIR()
{
  if (irrecv.decode(&results))
   {
    Serial.println(results.value, DEC);
    IR_decode(results.value);// Print the Serial 'results.value'
     irrecv.resume();   // Receive the next value
     flag=0;
   }
}
void useWifi()
{
 if(esp8266.available()) // check if the esp is sending a message
  {


   if(esp8266.find("+IPD,"))
   {
    delay(1000); // wait for the serial buffer to fill up (read all the serial data)
    // get the connection id so that we can then disconnect
    int connectionId = esp8266.read()-48; // subtract 48 because the read() function returns
                                          // the ASCII decimal value and 0 (the first decimal number) starts at 48

    esp8266.find("pin="); // advance cursor to "pin="

    pin = (esp8266.read()-48)*10;
    pin+=(esp8266.read()-48);// get first number i.e. if the pin 13 then the 1st number is 1, then multiply to get 10
   control(pin);// toggle pin

    // make close command
    String closeCommand = "AT+CIPCLOSE=";
    closeCommand+=connectionId; // append connection id
    closeCommand+="\r\n";
   sendData(closeCommand,1000,DEBUG);      // close connection
    flag=0;
   }
  }
 }

 void IR_decode(int a)
 {
  switch(a)
  {
    case 33441975:pin=10;
    break;
    case 33446055:pin=11;
    break;
    case 33454215:pin=12;
    break;
    case 33456255:pin=13;
    break;
    case 33439935:pin=14;
    break;
    case 33472575:pin=15;
    break;
    case 33431775:pin=0;
    break;
    case 33480735:pin=1;
    default:break;
  }
  control(pin);
 }
String senddat(String command, const int timeout, boolean debug)
{
   String response = "";

   esp8266.print(command); // send the read character to the esp8266

   long int time = millis();

   while( (time+timeout) > millis())
   {
     while(esp8266.available())
     {

       // The esp has data so display its output to the serial window
       char c = esp8266.read(); // read the next character.
       response+=c;
     }
   }
  if(debug)
   {
```

```
    Serial.print(response);
   }

   return response;
}
void control(int a)
{
 switch(a)
 {
  case 10:DecideAndUse(4,1);
  break;
  case 11:DecideAndUse(5,1);
  break;
  case 12:DecideAndUse(6,1);
  break;
  case 13:DecideAndUse(4,0);
  break;
  case 14:DecideAndUse(5,0);
  break;
  case 15:DecideAndUse(6,0);
  break;
  case 16:flg=1;
  break;
  case 17:flg=0;
  break;
  case 0: digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  break;
  case 1:digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  break;
  default: break;
 }
}
void DecideAndUse(int a,int b)
{
 if(c>0)
 {
  if(b==1)
    {
    digitalWrite(a,HIGH);
    }
    else
    {
    digitalWrite(a,LOW);
    }
 }
 else
 {
  if(flg==1)
  {
   if(b==1)
   {
   digitalWrite(a,HIGH);
   }
   else
   {
   digitalWrite(a,LOW);
   }
   flag=0;
  }
  else
  {
   digitalWrite(a,LOW);
  }
 }
}
```

2.8 HTML Code

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8"/>
 <title>iCaRE Home Automation</title>
  <<link rel="stylesheet" type="text/css" href="style.css">
 <link rel="stylesheet" href="bootstrap.min.css"/>
 <link rel="stylesheet" href="font-awesome.min.css">

 </head>
 <body>

  <div class="menu">

   <!-- Menu icon -->
   <div class="icon-close">
    <img src="close.png">
   </div>

   <!-- Menu -->
   <ul>
   <li><a href="index.html">Home</a></li>
    <li><a href="about_us.html">About</a></li>
    <li><a href="our_vision.html">Our Vision</a></li>
    <li><a href="help.html">Help</a></li>
    <li><a href="contact.html">Contact</a></li>
   </ul>
  </div>
```

```
<!-- Main body -->
<div class="jumbotron">

 <div class="icon-menu">
  <i class="fa fa-bars"></i>
  Menu
  </div>
<section class="container">
 <div class="row text-center">
 <h2>Light</h2>
 <button id="10" class="led">ON</button>
 <button id="13" class="led">OFF</button>
 </div>
 <div class="row text-center">
 <h2>Fan</h2>
 <button id="11" class="led">ON</button>
 <button id="14" class="led">OFF</button>
 </div>
 <div class="row text-center">
 <h2>Air Conditioning</h2>
 <button id="12" class="led">ON</button>
 <button id="15" class="led">OFF</button>
 </div>
 <div class="row text-center">
 <h2>Remote Control</h2>
 <button id="16" class="led">ON</button>
 <button id="17" class="led">OFF</button>
 <p>Use this Button Only when you want to TURN ON or OFF your Devices <em>Remotely</em></p>
 </div>
 </section>
 <script src="jquery.min.js"></script>
 <script src="app.js"></script>
 </body>
</html>
```

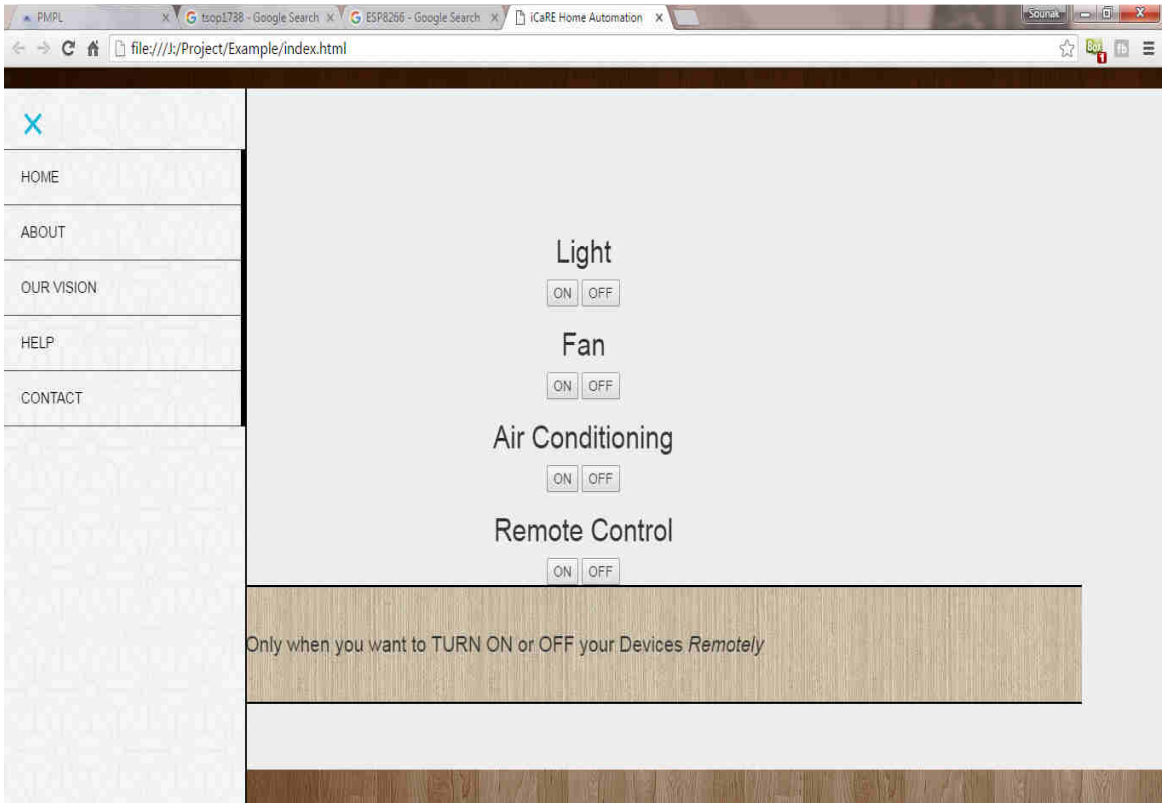

Fig.2.5: Snapshot Of The Webpage

2.9 JS Code

```
var main = function() {

 $('.icon-menu').click(function() {
  $('.menu').animate({
   left: "0px"
  }, 200);
 });
$('.icon-close').click(function() {
  $('.menu').animate({
   left: "-285px"
  }, 200);
```

```
  $('body').animate({
    left: "0px"
  }, 200);
 });
 $(".led").click(function(){
      var p = $(this).attr('id'); // get id value (i.e. pin13,
pin12, or pin11)
      // send HTTP GET request to the IP address with the
parameter "pin" and value "p", then execute the function
      $.get("http://192.168.4.1:80/", {pin:p}); // execute get
request
    });
};
$(document).ready(main);
```

III. CONCLUSION

In conclusion, our device works as expected. We are expecting to add more and diverse features to take "automated" to "smart" homes. Going by the trend, this is going to be the future of clean and smart living. These devices can run 24x7, 365 days and with a little tweak, we can also monitor the number of people present. Indirectly, this can also be used as anti-theft devices to monitor the home remotely.

REFERENCES

[1] Infra Red Sensor (IR), From the article "Passive Infrared Sensor", Wikipedia – The Free Encylopedia.
[2] Arduino, From Arduino Project's Official Website.
[3] ESP8266, From ESP8266 official website.
[4] AT Commands table as provided in an article published in itead.cc.
[5] The IR library was used as published in github directory by username "z3t0".