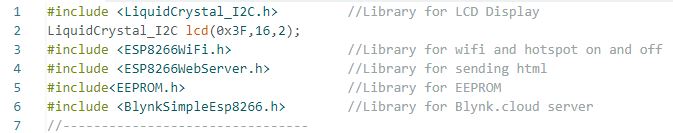
1. 

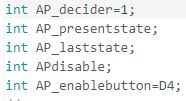
These are the necessary libraries which would be required in our system.

2.for captive portal we require following library:

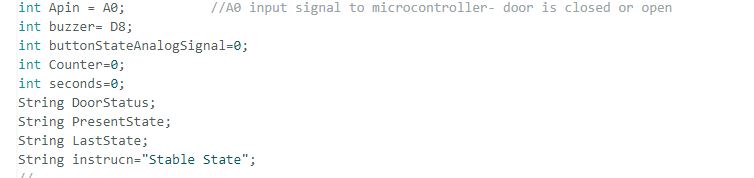


3.

Setting ip address for access point(hotspot) and user defined variable for DNSServer.

4.variables required for counting switch which is used to change the mode of system from wifi to access point and vice versa. D4 pin is declare to receive signal that whether the switch is pressed or not.

5.

A0 pin is used to receive an analog signal from photo coupler that it is interrupted by an opaque object , sending low values as emitter junction current is low no light is detected by photodiode at base of photo transistor or not interrupted then sending high value to the micro controller as photodiode receive an infra red light that causes to produce emitter current higher.

Low signal —>photo coupler is interrupted= door is closed

High signal —>photo coupler is not interrupted= door is open

D8 pin connected to buzzer .Sends high to buzzer when door is opened since long time. Time for this can be adjusted by code. By default we have set this time for 1 minute which can be explan in further code.

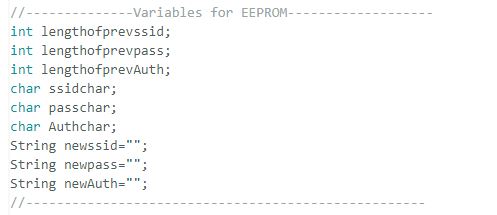
“String DoorStatus” stores the status of door whether it is open or closed.

“Int seconds” is variable used for timer.

“String instrucn” stores whether the system is stable or not by default it set to “stable state”.

Other variables used for requirement in code and can be explained later.

6.



These are the variables used for saving ssid,pass,authentication token to EEPROM when user try to change these parameters through captive portal.

7.



String used to store input parameters from web server.

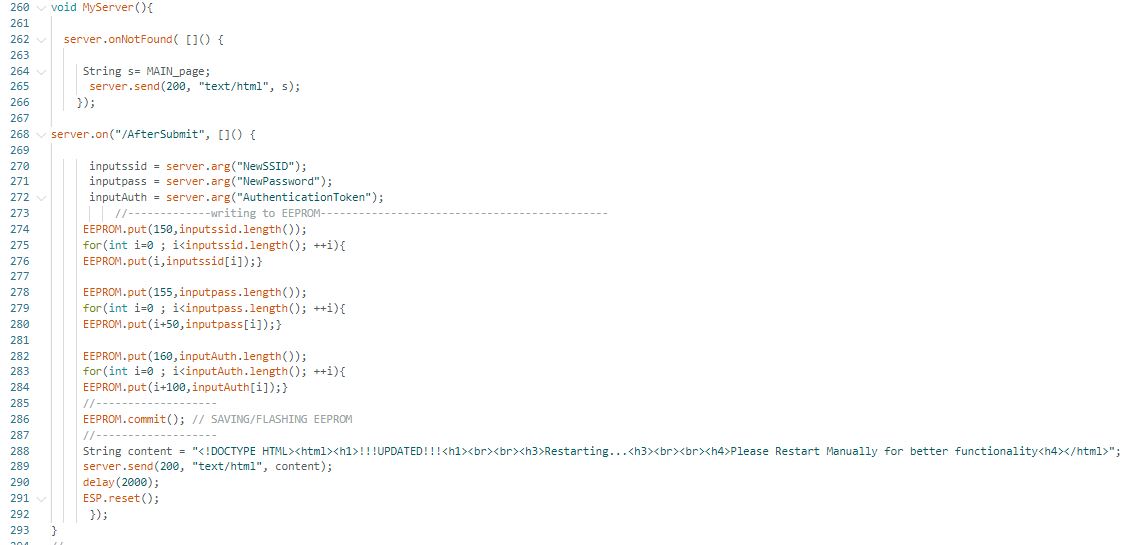
8.



Constant char userdefined\_name[] PROGMEM= (parameters) :any thing written in the parameters of this function will considered as string, this function is used to store our html code as string. Because html has inverted commas in between the code that makes difficult to store in general way of declaring string and storing it in single variable.

9.

this is necessary to declare virtual server server port.

10.

myserver() a user defined function. Calling this function when Access point is enabled. Server.onNotFount() will execute data even if user not requested . this has parameter as send html through port 200 and html is stored in “MAIN\_page” which is now equals to “s” this all html code send by function “server.send”.

11.(HTML PAGE IMAGE)

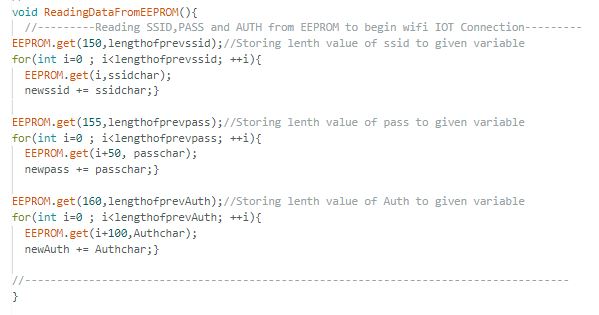
12. When user enter all the credentials and submit it by clicking on submit button, all input from server (html page) is now fetched and stored in respective variables.Size of parameters of Ssid will store at location 150 of EEPROM ,password at 155 and auth token at 160. These sizes have information how long our parameters are and saved at this location. And these data is used when we fetch these parameters for wifi connection.

For loop is used to write every character of parameters at specific location one by one. Ssid will store from 0 to its length. Password store from location 50 to its length. Auth token stores from location 100 to its length.

“EEPROM.commit” will save the data which was written in the process. Even if the system is off the data will not erase.

After 2 second . the ESP8266 will restart and begin wifi connection with new ssid,password,authentication token id.

13

.

“ReadingDataFromEEPROM” user defined function is used to read data from EEPROM. First length of ssid is reading from location 150, there we can find the size of our previous ssid that is stored. Now using loop from location 0 to its length we are storing character one by one to “newssid ” variable. Similarly we can read for other stored parameters.

14

Inside the void setup.some Necessary initialisation, EEPROM begin with 512 kb(ESP8266 has total 1kb memory in EEPROM). Calling function “ReadingDataFromEEPROM” for reading ssid,password,authentication token which was saved in EEPROM. Initialising server lcd, by“server.begin” and “lcd.begin”. Declaring modes of various pins. “Lcd.backlight” will turn on backlight of lcd . “lcd.clear” clear the Lcd screen. “Lcd.print” is used to print the characters on lcd. Lcd has dimension of 2 rows and 16 columns “lcd.setcursor” is used to set the cursor at specific location on lcd display. Then for loop is used to scroll lcd text which is currently on displayed.

15.

Here in void loop , the instructions in void loop will executed repeatedly.

Line 112 and 113: this code is to change type from char to char array because “blynk.begin” take char array as parameters.

Now we are setting push button for changing mode to Access Point or Wifi mode, working behind this to count push button if push button pressed for even times then it turn on hotspot and act as Access point . where user can connect the hotspot, if any device is connected to its access point ,captive portal will bring the html page automatically if not then user can type ip address of it on any browser, an web server will load where user can set the new credentials by filling the required field on web page. As we can see in the code “AP\_decider” where count is stored, if it is even it disconnect the current wifi connection and on hotspot and captive portal begin given ip address in the code. “Myserver” function calling at this point to send server instruction which was defined in the function.

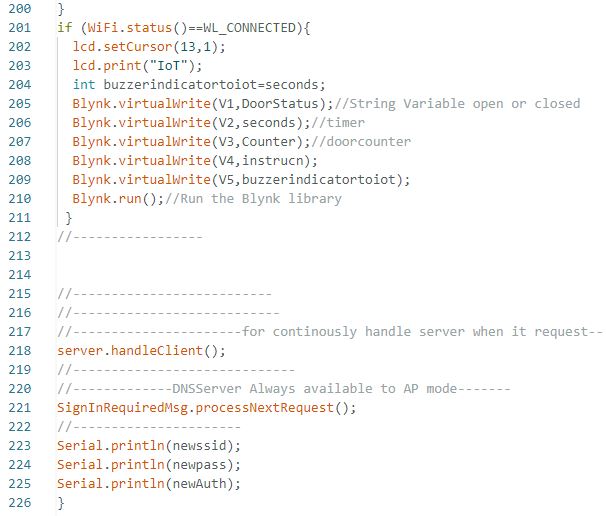
If the the push button is pressed odd times then it turn off the Access point and start wifi connection. Setting lcd cursor at bottom corner. Clearing “AP” mode indicator from the lcd. In the wifi mode we need to establish connection with blynk server to send data on internet “Blynk,config” is used with authentication token as its parameters (This authentication token is available on www.blynk.cloud when an user sign up there)

16.

Line 148: this code reads the analog signal from photo coupler from the pin A0. if the analog signal is high then door is open other wise the signal is low it means the photo coupler is interrupted and door is closed. The status is stored in “DoorStatus” variable. This variable can be used to send data to internet, and count the number of times door is open. If door is open then it counts in counter variable also display this on lcd. If it is closed then timer(“seconds” variable in the code) which is started when door is open is now stoped by the equation on line 169 in code, make buzzer off clearing the lcd and display on lcd counter and door status.

17.

Other thing is explained here, the timer started here by increment seconds variable in delay with 1000 milli seconds. Line 187: When door is open for 60 seconds or more the buzzer is start beeping. This buzzer will stop when door will closed. We know that when door is open timer starts and delay of 1000 milli seconds to balance this delay we have delayed 1000 milli seconds in closed state also. Line 197: if wifi is not connected then iot indicator is cleared from lcd.

18.

Line 201: if wifi is connected , iot indicator is displayed on lcd.”blynk.virtualwrite” is used to send data to Blynk server on different virtual pins(V1,V2,V3,V4,V5). “Blynk.run” is necessary to send data continuously to the server and run all the blynk function. “server.handleClient” for continuously handle server when it request in Access point mode. “userdefine\_name.processNextRequest” for DNS server always available to Access point.

“Serial.println” is used for debugging can see on serial monitor 9600 to check which ssid,password and authentication id is used by ESP8266 or we can say our system name “RMS7811i”.

**FULL CODE:**

#include <LiquidCrystal\_I2C.h> //Library for LCD Display

LiquidCrystal\_I2C lcd(0x3F,16,2);

#include <ESP8266WiFi.h> //Library for wifi and hotspot on and off

#include <ESP8266WebServer.h> //Library for sending html

#include<EEPROM.h> //Library for EEPROM

#include <BlynkSimpleEsp8266.h> //Library for Blynk.cloud server

//--------------------------------

#include <DNSServer.h> //For Captive Portal Auto Connect to the server in AP Mode

//--------------------------------

IPAddress ManuallysetIP(192,168,4,1); //setting manual ip address

DNSServer SignInRequiredMsg; // user define variable for DNSServer

//-----------------------------------------

int AP\_decider=1;

int AP\_presentstate;

int AP\_laststate;

int APdisable;

int AP\_enablebutton=D4; //Pin D4 for input signal

//--------------------------------------

int Apin = A0; //A0 input signal to microcontroller- door is closed or open

int buzzer= D8;

int buttonStateAnalogSignal=0;

int Counter=0;

int seconds=0;

String DoorStatus;

String PresentState;

String LastState;

String instrucn="Stable State";

//--------------------------------------

//--------------Variables for EEPROM-------------------

int lengthofprevssid;

int lengthofprevpass;

int lengthofprevAuth;

char ssidchar;

char passchar;

char Authchar;

String newssid="";

String newpass="";

String newAuth="";

//----------------------------------------------------

//------------------------Variables to store Credentials from Web Server--------

String inputssid;

String inputpass;

String inputAuth;

//-------------------------------------------------------------------------------

//------------HTML Code Store in Variable----------------------------------------

const char MAIN\_page[] PROGMEM = R"=====(

<!DOCTYPE html>

<html>

<body>

<center><h1>RMS7811i</h1></center>

<center><table border="1" bordercolor="blue" bgcolor="skyblue" height="400" width="400">

<td>

<center>

<h3>UPDATE CREDENTIALS</h3>

<form action="/AfterSubmit">

New SSID:<br>

<input type="text" name="NewSSID" placeholder="Enter your New SSID" required>

<br>

Password:<br>

<input type="password" name="NewPassword" placeholder="Enter your SSID Password">

<br>

Authentication Token<br>

<input type="text" name="AuthenticationToken" placeholder="Enter your BLYNK Auth" required>

<br><br>

<input type="submit" value="Submit">

</form><br><br>

</div>

</table>

</center>

</td>

</center>

</body>

</html>

)=====";

//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------

ESP8266WebServer server(80); //Server on port 80

//-------------------------------------------------------------------------------

void setup() {

EEPROM.begin(512);

ReadingDataFromEEPROM();

server.begin();

Serial.begin(9600);

pinMode(AP\_enablebutton,INPUT\_PULLUP);

pinMode(Apin,INPUT);

pinMode(buzzer,OUTPUT);

lcd.backlight();

lcd.begin();

lcd.clear();

lcd.print("--System Is ON--"); // initial display of lcd

lcd.setCursor(3,1);// LCD Cursor set at- 3rd coloumn 1st row

lcd.print("RMS7811i");

delay(2000);

lcd.clear();

lcd.setCursor(0,0);// LCD Cursor set at- 0th coloumn 0th row

lcd.print("Your System is Now Monitered with IoT ");

delay(1000);

for (int pos=0; pos<24;pos++)

{

lcd.scrollDisplayLeft();

delay(500);

}

}

void loop() {

//-----------------------AP Decider------------------------------------------

char newAuthCharrArray[newAuth.length()+1];

newAuth.toCharArray(newAuthCharrArray,newAuth.length()+1); //string to charr array

if(digitalRead(AP\_enablebutton)== HIGH){

AP\_presentstate=1;

}

if(digitalRead(AP\_enablebutton)== LOW){

AP\_presentstate=0;

}

if(AP\_decider %2==0){

lcd.setCursor(14,1);

lcd.print("AP");

APdisable--;

if (APdisable==0){

AP\_presentstate=0;

}

}

if(AP\_presentstate!= AP\_laststate){

if(AP\_presentstate==0){

AP\_decider++;}

if(AP\_decider %2==0){

APdisable=120;

WiFi.disconnect();

WiFi.softAP("RMS7811i","12345678");

SignInRequiredMsg.start(53, "\*", ManuallysetIP);

MyServer();

}

if(AP\_decider %2 != 0){

WiFi.softAPdisconnect();

WiFi.begin(newssid,newpass);

lcd.setCursor(14,1);

lcd.print(" ");//Clearing AP mode display

Blynk.config(newAuthCharrArray,"blynk.cloud", 8080);

}

}

AP\_laststate=AP\_presentstate;

//--------------------------------------------------------------------------

buttonStateAnalogSignal= analogRead(Apin); //Analog Signal from photoCoupler

if (buttonStateAnalogSignal>700){

DoorStatus="Open";

PresentState=DoorStatus;

}

else{

DoorStatus="Close";

PresentState=DoorStatus;

instrucn="Stable State";

}

if(PresentState != LastState){

if(PresentState=="Open"){

Counter++;

lcd.clear();

lcd.setCursor(1,0);

lcd.print("Door is open");// display that door is currently open

}

if(PresentState =="Close"){

digitalWrite(buzzer,LOW);

seconds=seconds\*0;

lcd.clear();

lcd.setCursor(1,0);

lcd.print("Door Closed!");

lcd.setCursor(0,1);

lcd.print("DoorOpened:"); //display door counter- part1

lcd.print(Counter); //display door counter- part2

delay(1000);

}

LastState=PresentState;

}

if(PresentState=="Open"){

seconds++;

delay(1000);

lcd.setCursor(0,1); // second row of display

lcd.print("Since:");

lcd.print(seconds);

lcd.print("s");

if(seconds>=60){

instrucn="Buzzer is ON";

digitalWrite(buzzer,HIGH);

}

}

if(PresentState =="Close"){

delay(1000);

}

//----------------------

if (WiFi.status()!=WL\_CONNECTED){

lcd.setCursor(13,1);

lcd.print(" ");

}

if (WiFi.status()==WL\_CONNECTED){

lcd.setCursor(13,1);

lcd.print("IoT");

int buzzerindicatortoiot=seconds;

Blynk.virtualWrite(V1,DoorStatus);//String Variable open or closed

Blynk.virtualWrite(V2,seconds);//timer

Blynk.virtualWrite(V3,Counter);//doorcounter

Blynk.virtualWrite(V4,instrucn);

Blynk.virtualWrite(V5,buzzerindicatortoiot);

Blynk.run();//Run the Blynk library

}

//-----------------

//--------------------------

//---------------------------

//----------------------for continously handle server when it request--

server.handleClient();

//-----------------------------

//-------------DNSServer Always available to AP mode-------

SignInRequiredMsg.processNextRequest();

//----------------------

Serial.println(newssid);

Serial.println(newpass);

Serial.println(newAuth);

}

//--------------functions---------------------------------------

void ReadingDataFromEEPROM(){

//---------Reading SSID,PASS and AUTH from EEPROM to begin wifi IOT Connection---------

EEPROM.get(150,lengthofprevssid);//Storing lenth value of ssid to given variable

for(int i=0 ; i<lengthofprevssid; ++i){

EEPROM.get(i,ssidchar);

newssid += ssidchar;}

EEPROM.get(155,lengthofprevpass);//Storing lenth value of pass to given variable

for(int i=0 ; i<lengthofprevpass; ++i){

EEPROM.get(i+50, passchar);

newpass += passchar;}

EEPROM.get(160,lengthofprevAuth);//Storing lenth value of Auth to given variable

for(int i=0 ; i<lengthofprevAuth; ++i){

EEPROM.get(i+100,Authchar);

newAuth += Authchar;}

//-------------------------------------------------------------------------------------

}

void MyServer(){

server.onNotFound( []() {

String s= MAIN\_page;

server.send(200, "text/html", s);

});

server.on("/AfterSubmit", []() {

inputssid = server.arg("NewSSID");

inputpass = server.arg("NewPassword");

inputAuth = server.arg("AuthenticationToken");

//-------------writing to EEPROM---------------------------------------------

EEPROM.put(150,inputssid.length());

for(int i=0 ; i<inputssid.length(); ++i){

EEPROM.put(i,inputssid[i]);}

EEPROM.put(155,inputpass.length());

for(int i=0 ; i<inputpass.length(); ++i){

EEPROM.put(i+50,inputpass[i]);}

EEPROM.put(160,inputAuth.length());

for(int i=0 ; i<inputAuth.length(); ++i){

EEPROM.put(i+100,inputAuth[i]);}

//-------------------

EEPROM.commit(); // SAVING/FLASHING EEPROM

//-------------------

String content = "<!DOCTYPE HTML><html><h1>!!!UPDATED!!!<h1><br><br><h3>Restarting...<h3><br><br><h4>Please Restart Manually for better functionality<h4></html>";

server.send(200, "text/html", content);

delay(2000);

ESP.reset();

});

}

//-------------------------------------------------------------------------------