#include <ArduinoJson.h>

#include <PubSubClient.h>

#include <WiFi.h>

#include <WiFiClient.h>

#include <WebServer.h>

#include <Ticker.h>

#include <EEPROM.h>

#include <WiFiUdp.h>

#include <ESPmDNS.h>

#include <DNSServer.h>

const byte DNS\_PORT = 53;

DNSServer dnsServer;

WebServer webServer(80); //dns

#include "helpers.h"

#include "global.h"

//#include "WebSocketsServer.h"

//#include <Hash.h>

#include "Adafruit\_MQTT.h"//ada

#include "Adafruit\_MQTT\_Client.h"//ada

//RTC

//#include <RTClib.h>

#include <Wire.h>

long timer = 0;

#include "AT24CX\_DEMO.h"

//WebSocketsServer webSocket = WebSocketsServer(81);

/\*

Include the HTML, STYLE and Script "Pages"

\*/

#include "Page\_Root.h"

#include "Page\_Admin.h"

#include "Page\_Script.js.h"

#include "Page\_Style.css.h"

#include "Page\_NTPsettings.h"

#include "Page\_Information.h"

#include "Page\_General.h"

#include "PAGE\_NetworkConfiguration.h"

#include "example.h"

#include "Page\_cs.h"

#include "Page\_Robo.h"

#include "Page\_Build.h"

#include "Page\_Adc.h"

#include "Page\_Email.h"

#include "Page\_udp.h"

#include "Page\_mem32.h"

#include "Page\_RTC.h"

#include "Timer\_PRGM.h"

#include "PAGE\_MRTC.h"

enum log\_t {LOG\_LEVEL\_DEBUG};

#include <LiquidCrystal\_I2C.h>

//int totalColumns = 16;

//int totalRows = 2;

//LiquidCrystal\_I2C lcd(0x27, totalColumns, totalRows);

//RTC\_DS3231 rtc;

#define ACCESS\_POINT\_NAME "IOT\_SMART\_SOLAR CLEANING"

#define ACCESS\_POINT\_PASSWORD "12345678"

#define AdminTimeOut 180 // Defines the Time in Seconds, when the Admin-Mode will be diabled

#define D0 23//16

//#define D1 18 // 5 Ulsonic-2 tri

//#define D2 5// 4 Ulsonic-2 echo

//#define D3 25// 0 servo

//#define D4 14// 2--buzzer

#define D5 13// 14-RLY1

#define D6 12// 12-RLY2

#define D7 27// 13-RLY 3----AP mode

#define D8 26 // 15-RLY 4 --STA mode

#define D9 15 // 3

//#define D10 2// 1 Ulsoic-1 eco

#define I1 36

#define I2 39

#define I3 34

#define I4 35

#define I5 32//RESET

#define MQTT\_PORT 1883

int sw1=0;

int sw2=0;

int sw3=0;

int sw4=0;

int sw5=0;

int sw6=0;

String webPage = "";

String msg;

String responseJson = "";

MDNSResponder mdns;

WiFiClient wifiClient;

PubSubClient client(config.mqttServer.c\_str(), config.mqttPort.toInt(), wifiClient);

void callback(char\* topic, byte\* payload, unsigned int length);

String addada1;

String addada2;

String addada3;

String addada4;

String addada5;

String addada6;

String addada7;

String addada8;

String addada9;

String addada10;

String addada11;

String addada12;

String addada13;

String addada14

WiFiClient aclient;

//Adafruit\_MQTT\_Client mqtt(&aclient, MQTT\_SERV, MQTT\_PORT, MQTT\_NAME, MQTT\_PASS);

Adafruit\_MQTT\_Client mqtt(&aclient,(char\*)test3, MQTT\_PORT, (char\*)test1,(char\*) test2);

Adafruit\_MQTT\_Subscribe relay1 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic1);

Adafruit\_MQTT\_Publish RELAY1 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic1);

Adafruit\_MQTT\_Subscribe relay2 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic2);

Adafruit\_MQTT\_Publish RELAY2 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic2);

Adafruit\_MQTT\_Subscribe relay3 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic3);

Adafruit\_MQTT\_Publish RELAY3 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic3);

Adafruit\_MQTT\_Subscribe relay4 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic4);

Adafruit\_MQTT\_Publish RELAY4 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic4);

Adafruit\_MQTT\_Subscribe reply = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic5);

Adafruit\_MQTT\_Publish REPLY = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic5);

Adafruit\_MQTT\_Subscribe serial1 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic6);

Adafruit\_MQTT\_Publish SERIAL1 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic6);

Adafruit\_MQTT\_Subscribe ch1 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic7);

Adafruit\_MQTT\_Publish CH1 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic7);

Adafruit\_MQTT\_Subscribe ch2 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic8);

Adafruit\_MQTT\_Publish CH2 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic8);

Adafruit\_MQTT\_Subscribe ch3 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic9);

Adafruit\_MQTT\_Publish CH3 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic9);

Adafruit\_MQTT\_Subscribe ch4 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic10);

Adafruit\_MQTT\_Publish CH4 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic10);

Adafruit\_MQTT\_Subscribe ch5 = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic11);

Adafruit\_MQTT\_Publish CH5 = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic11);

Adafruit\_MQTT\_Subscribe adc = Adafruit\_MQTT\_Subscribe(&mqtt,(char\*)topic12);

Adafruit\_MQTT\_Publish ADC = Adafruit\_MQTT\_Publish(&mqtt,(char\*)topic12);

unsigned long LCD\_previousMillis = 0; // will store last time LED was updated

const long LCD\_interval = 5000; // interval at which to blink (milliseconds)

unsigned long PING\_previousMillis = 0; // adafruit mqtt ping to alive status every 60 sec

const long PING\_interval = 120000; //(2 MIN KEEP ALIVE)

int PING\_FLAG = 0;

String addDEVICE1;

String addDEVICE2;

String addDEVICE3;

String addDEVICE4;

String addDEVICE5;

String addDEVICE6;

String addREPLY;

String addCH1;

String addCH2;

String addCH3;

int RS1 = 0;

int RS2 = 0;

int RS3 = 0;

int RS4 = 0;

int RS5 = 0;

int RS6 = 0;

#define SIGNAL\_PIN 33

float adc\_voltage = 0.0;

float R1 = 57000.0;

float R2 = 7500.0;

float ref\_voltage = 3;

float adc\_value = 0.0;

float in\_voltage;

float in\_current;

void setup () {

EEPROM.begin(700);

Serial.begin(115200);

Serial.println("Starting IOT HARDWARE BRIDGE -CLOUD- WIFI DEVELOPMENT BOARD-1.8.5\_STABLE");

// Serial.println("M:");

// Serial.println(config.mqttServer);

pinMode(D0, OUTPUT);

digitalWrite(D0, HIGH);

//pinMode(D4, OUTPUT);

// digitalWrite(D4, HIGH);

pinMode(D5, OUTPUT);

// digitalWrite(D5, HIGH);

pinMode(D6, OUTPUT);

// digitalWrite(D6, HIGH);

pinMode(D7, OUTPUT);

// digitalWrite(D7, HIGH);

pinMode(D8, OUTPUT);

// digitalWrite(D8, HIGH);

pinMode(D9, OUTPUT);

pinMode(I1, INPUT);

pinMode(I2, INPUT);

pinMode(I3, INPUT);

pinMode(I4, INPUT);

pinMode(I5, INPUT);//RESET

lcd.init();

lcd.backlight(); // use to turn on and turn off LCD back light

lcd.setCursor(0, 0);

lcd.print(" IOT-SMART");

lcd.setCursor(0, 1);

lcd.print("SOLAR CLEANING");

if (!ReadConfig())

{

// DEFAULT CONFIG

config.ssid = "MYSSID";

config.password = "MYPASSWORD";

config.dhcp = true;

config.IP[0] = 192;config.IP[1] = 168;config.IP[2] = 1;config.IP[3] = 100;

config.Netmask[0] = 255;config.Netmask[1] = 255;config.Netmask[2] = 255;config.Netmask[3] = 0;

config.Gateway[0] = 192;config.Gateway[1] = 168;config.Gateway[2] = 1;config.Gateway[3] = 1;

config.ntpServerName = "0.de.pool.ntp.org";

config.Update\_Time\_Via\_NTP\_Every = 0;

config.timezone = -10;

config.daylight = true;

config.DeviceName = "Label1";

config.AutoTurnOff = false;

config.AutoTurnOn = false;

config.TurnOffHour = 0;

config.TurnOffMinute = 5;

config.TurnOnHour = 0;

config.TurnOnMinute = 0;

config.cloud = 0;

config.ada = false;

config.mqttServer = "cloudmqtt.com";

config.mqttUser = "user name";

config.mqttPassword = "password";

config.mqttPort ="12345";

// config.mqttTopic ="Topic to subscribe"; cloud ynable to connect due to overload of memory

config.Email = "Lagos,NG";

config.Code = "LABEL";

WriteConfig();

Serial.println("General config applied");

}

ReadConfig();

addada1=config.mqttUser+"/f/DEVICE1";

addada2=config.mqttUser+"/f/DEVICE2";

addada3=config.mqttUser+"/f/DEVICE3";

addada4=config.mqttUser+"/f/DEVICE4";

addada5=config.mqttUser+"/f/REPLY";

addada6=config.mqttUser+"/f/SERIAL1";

addada7=config.mqttUser+"/f/CH1";

addada8=config.mqttUser+"/f/CH2";

addada9=config.mqttUser+"/f/CH3";

addada10=config.mqttUser+"/f/CH4";

addada11=config.mqttUser+"/f/CH5";

addada12=config.mqttUser+"/f/ADC";

strcpy(test1, config.mqttUser.c\_str());

strcpy(test2, config.mqttPassword.c\_str());

strcpy(test3, config.mqttServer.c\_str());

strcpy(topic1, addada1.c\_str());

strcpy(topic2, addada2.c\_str());

strcpy(topic3, addada3.c\_str());

strcpy(topic4, addada4.c\_str());

strcpy(topic5, addada5.c\_str());

strcpy(topic6, addada6.c\_str());

strcpy(topic7, addada7.c\_str());

strcpy(topic8, addada8.c\_str());

strcpy(topic9, addada9.c\_str());

strcpy(topic10, addada10.c\_str());

strcpy(topic11, addada11.c\_str());

strcpy(topic12, addada12.c\_str());

WiFi.mode(WIFI\_STA);

// }

ConfigureWifi();

//server.on ( "/cs.html", handleConsole);

server.on ( "/build.html", processExample );

server.on ( "/admin/filldynamicdata", filldynamicdata );

server.on ( "/favicon.ico", []() {

Serial.println("favicon.ico");

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

} );

server.on ( "/admin.html", []() {

Serial.println("admin.html");

server.send ( 200, "text/html", PAGE\_AdminMainPage );

} );

server.on ( "/console.html", []() {

Serial.println("console.html");

server.send ( 200, "text/html", PAGE\_CONSOLE );

} );

server.on ( "/robo.html", []() {

Serial.println("robo.html");

server.send ( 200, "text/html", PAGE\_ROBO );

// email\_flag=1;

} );

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

Serial.println("build.html");

server.send ( 200, "text/html", PAGE\_BUILD );

//adc\_flag=0;

// email\_flag=0;

} );

server.on ( "/adc.html", []() {

Serial.println("adc.html");

server.send ( 200, "text/html", PAGE\_ADC );

// adc\_flag=1;

} );

server.on ( "/mem32.html", []() {

Serial.println("mem32.html");

server.send ( 200, "text/html", PAGE\_Mem32 );

// adc\_flag=1;

} );

/\* server.on ( "/email.html", []() {

Serial.println("email.html");

server.send ( 200, "text/html", PAGE\_EMAIL );

} );\*///NOT USED IF GET OR POST A PAGE

// server.on ( "/adc", send\_information\_tag\_values\_html );

server.on ( "/admin/tagvalues", send\_information\_tag\_values\_html );

server.on ( "/udp.html", send\_udp\_html);

server.on ( "/admin/udpvalues", send\_udp\_configuration\_values\_html);

server.on ( "/admin/udp", send\_udp\_value\_html);

server.on ( "/weather.html", send\_email\_html);

server.on ( "/admin/emailvalues", send\_email\_configuration\_values\_html);

server.on ( "/admin/Email", send\_email\_value\_html);

server.on ( "/config.html", send\_network\_configuration\_html );

server.on ( "/info.html", []() {

Serial.println("info.html");

server.send ( 200, "text/html", PAGE\_Information );

} );

server.on ( "/ntp.html", send\_NTP\_configuration\_html );

server.on ( "/RTC.html", []() {

// Serial.println("info.html");

server.send ( 200, "text/html", PAGE\_RTC );

} );

server.on ( "/admin/RTC", send\_RTC\_values\_html);

server.on ( "/general.html", send\_general\_html );

server.on ( "/MRTC.html", send\_MRTC\_html );

server.on ( "/example.html", []() { server.send ( 200, "text/html", PAGE\_example ); } );

server.on ( "/style.css", []() {

Serial.println("style.css");

server.send ( 200, "text/plain", PAGE\_Style\_css );

} );

server.on ( "/microajax.js", []() {

Serial.println("microajax.js");

server.send ( 200, "text/plain", PAGE\_microajax\_js );

} );

server.on ( "/admin/values", send\_network\_configuration\_values\_html );

server.on ( "/admin/connectionstate", send\_connection\_state\_values\_html );

//server.on ( "/admin/Cpuinfos", send\_information\_values\_html );

server.on ( "/admin/infovalues", send\_information\_values\_html );

server.on ( "/admin/ntpvalues", send\_NTP\_configuration\_values\_html );

server.on ( "/admin/generalvalues", send\_general\_configuration\_values\_html);

server.on ( "/admin/devicename", send\_devicename\_value\_html);

server.on ( "/admin/MRTCvalues", send\_MRTC\_configuration\_values\_html);

server.on ( "/admin/relayvalues", send\_information\_relayvalues\_html );

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

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

});

Serial.println("WIFI\_FLAG=1");

StaticJsonBuffer<300> JSONbuffer;

JsonObject& JSONencoder = JSONbuffer.createObject();

ReadConfig();

JsonArray& IPaddress = JSONencoder.createNestedArray("IPaddress");

IPaddress.add(WiFi.localIP()[0]);

IPaddress.add(WiFi.localIP()[1]);

IPaddress.add(WiFi.localIP()[2]);

IPaddress.add(WiFi.localIP()[3]);

int lenghtSimple = JSONencoder.measureLength();

Serial.print("Less overhead JSON message size: ");

Serial.println(lenghtSimple);

Serial.println("Connecting to WiFi..");

char JSONmessageBuffer[300];

JSONencoder.prettyPrintTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));

Serial.println(JSONmessageBuffer);

//Udp.write(JSONmessageBuffer);

JSONencoder.printTo(Udp);

}

Serial.println("Connected to the WiFi network");

Serial.println("");

//wifi\_flag=1;

//Serial.println("WIFI\_FLAG=1");

Serial.println("WiFi connected");

Serial.println("IP address: ");

//Serial.println(WiFi.localIP());

//ada

//Subscribe to the onoff topic

mqtt.subscribe(&relay1);

mqtt.subscribe(&relay2);

mqtt.subscribe(&relay3);

mqtt.subscribe(&relay4);

mqtt.subscribe(&reply);

mqtt.subscribe(&serial1);

mqtt.subscribe(&ch2);

// mqtt.subscribe(&eetopic);

//CLIENT ID ADDED

//PubSubClient (server, port, [callback], client, [stream])

if(config.cloud == 1)

{

client.setServer(config.mqttServer.c\_str(),config.mqttPort.toInt());

client.setCallback(callback);

}

Udp.begin(localUdpPort);

delay(9000);

Serial.printf("Now listening at IP %s, UDP port %d\n", WiFi.localIP().toString().c\_str(), localUdpPort);

// digitalWrite(D7, LOW);

// digitalWrite(D8, HIGH);

lcd.clear();

lcd.setCursor(0,1);

lcd.print(WiFi.localIP().toString());

//auto connect

String wchk=config.ssid;

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

{

if(wchk != "MYSSID")

{

if(digitalRead(I5) == LOW)//wifi auto connect

{

//Serial.println("initwifi activated");

delay(10000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" DEVICE");

lcd.setCursor(0, 1);

lcd.print("Booting..........");//delay for to protect connection going to ap mode

Serial.println("Booting..");

delay(20000);

initWiFi();

}else{

WiFi.mode(WIFI\_AP);

WiFi.softAP( ACCESS\_POINT\_NAME , ACCESS\_POINT\_PASSWORD);

Serial.println("AP MODE-RSW");}}}

//auto connect

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

//delay(500);

WiFi.mode(WIFI\_AP);

WiFi.softAP( ACCESS\_POINT\_NAME , ACCESS\_POINT\_PASSWORD);

Serial.println("AP IP:");

Serial.println(WiFi.softAPIP());

// digitalWrite(D7, HIGH);

// digitalWrite(D8, LOW);

wifi\_flag=0;

Serial.println("WIFI\_FLAG=0");

lcd.clear();

lcd.setCursor(0,1);

lcd.print(WiFi.softAPIP());

// modify TTL associated with the domain name (in seconds)

// default is 60 seconds

dnsServer.setTTL(300);

// set which return code will be used for all other domains (e.g. sending

// ServerFailure instead of NonExistentDomain will reduce number of queries

// sent by clients)

// default is DNSReplyCode::NonExistentDomain

dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure);

// start DNS server for a specific domain name

dnsServer.start(DNS\_PORT, "gkwave", WiFi.softAPIP());

webServer.begin(); //dns

}

//webSocket.onEvent(webSocketEvent);

// broadcast udp awesome

/\*IPAddress broadcastIp = WiFi.localIP();

broadcastIp[3] = 255;

Serial.println(broadcastIp);

Udp.beginPacket(broadcastIp, localUdpPort);

Udp.print("!AIVDM,1,1,,A,14eG;o@034o8sd<L9i:a;WF>062D,0\*7D");

Udp.endPacket();

delay(2000);\*/

//SYS();

}

void loop () {

PING(); // ADA FRUIT

lcd.setCursor(0, 0);

lcd.print("V=");

lcd.setCursor(2, 0);

lcd.print(in\_voltage);

lcd.setCursor(8, 0);

lcd.print("I=");

lcd.setCursor(10, 0);

lcd.print(in\_current);

//solar\_v();

unsigned long LCD\_currentMillis = millis();

if (LCD\_currentMillis - LCD\_previousMillis >= LCD\_interval) {

LCD\_previousMillis = LCD\_currentMillis;

VOLT();

}

if(digitalRead(I5) == HIGH) { //RESET

// Serial.println("Reset total mem 10\n");

// mem.write(10,0);

Serial.printf("Reset Button Pressed!\n");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(WiFi.localIP());

lcd.setCursor(0, 1);

lcd.print("AP>3 & RST>10sec");

// digitalWrite(D4, HIGH);

// Key debounce handling

delay(100);

int startTime = millis();

while(digitalRead(I5) == HIGH) delay(50);

int endTime = millis();

if ((endTime - startTime) > 10000) {

// If key pressed for more than 10secs, reset all

Serial.printf("Reset to factory.\n");

frestore\_2();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" DEVICE");

lcd.setCursor(0, 1);

lcd.print("FRS-Completed");

delay(2000);

ESP.restart();

// RMakerFactoryReset(2);

} else if ((endTime - startTime) > 3000) {

Serial.printf("AP-MODE\n");

// If key pressed for more than 3secs, but less than 10, reset Wi-Fi

//RMakerWiFiReset(2);

AP();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" DEVICE");

lcd.setCursor(0, 1);

lcd.print("Swtiched to AP ");//delay for to protect connection going to ap mode

delay(2000);

}

}

if ((WiFi.status() != WL\_CONNECTED)&&(wifi\_flag==1)) {

lcd.setCursor(0,0);

lcd.print("Check your WiFi..");

lcd.setCursor(15,1);

lcd.print("N");

WiFi.mode(WIFI\_STA);

ConfigureWifi();

delay(2000);

//alarm3();

}

if(config.cloud == 1)

{

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

reconnect();

}

client.loop();

}

if((config.cloud == 2) && (WiFi.status() == WL\_CONNECTED))

{

//Connect/Reconnect to MQTT

MQTT\_connect();

//Read from our subscription queue until we run out, or

//wait up to 5 seconds for subscription to update

Adafruit\_MQTT\_Subscribe \* subscription;

while ((subscription = mqtt.readSubscription(5000)))

{

//If we're in here, a subscription updated...

if (subscription == &relay1)

{

//Print the new value to the serial monitor

Serial.print("relay1: ");

Serial.println((char\*) relay1.lastread);

//If the new value is "ON", turn the light on.

//Otherwise, turn it off.

if (!strcmp((char\*) relay1.lastread, "L5")) //-----------------SOLAR CLEANING

{

REPLY.publish("CLEANING STARTED");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("CLEANING STARTED");

digitalWrite(D5, HIGH);//MOTOR ACTUATOR

digitalWrite(D6, LOW);

digitalWrite(D7, HIGH); //PUMP

delay(4500);

digitalWrite(D5, LOW);

digitalWrite(D6, HIGH);

digitalWrite(D7, LOW);

delay(4300);

digitalWrite(D5, LOW);

digitalWrite(D6, LOW);

// Addstr=config.relay1+"ON"; //adding two strings

// REPLY.publish(Addstr.c\_str());

REPLY.publish("DONE");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("DONE");

}

else if (!strcmp((char\*) relay1.lastread, "D5"))

{

// digitalWrite(D1, LOW);

// LightsStatus.publish("OFF");

digitalWrite(D5, LOW);

sureshkumar = '0';

WriteConfig();

int Relay1 = 0;

config.Relay1 = Relay1;

EEPROM.write(400, config.Relay1);

EEPROM.commit();

// Addstr=config.relay1+"OFF"; //adding two strings

// REPLY.publish(Addstr.c\_str());

REPLY.publish("Relay1 OFF");

}

else

{

// REPLY.publish("ERROR");

}

}

if (subscription == &relay2)

{

//Print the new value to the serial monitor

Serial.print("relay2: ");

Serial.println((char\*) relay2.lastread);

//If the new value is "ON", turn the light on.

//Otherwise, turn it off.

if (!strcmp((char\*) relay2.lastread, "L6"))

{

digitalWrite(D6, HIGH);

sureshkumar = '0';

WriteConfig();

int Relay2 = 1;

config.Relay2 = Relay2;

EEPROM.write(401, config.Relay2);

EEPROM.commit();

// Addstr=config.relay2+"ON"; //adding two strings

// REPLY.publish(Addstr.c\_str());

REPLY.publish("Relay2 ON");

// RTC\_LOOP();//------------------------------RTCLOOP TEST

}

else if (!strcmp((char\*) relay2.lastread, "D6"))

{

digitalWrite(D6, LOW);

sureshkumar = '0';

WriteConfig();

int Relay2 = 0;

config.Relay2 = Relay2;

EEPROM.write(401, config.Relay2);

EEPROM.commit();

// Addstr=config.relay2+"OFF"; //adding two strings

// REPLY.publish(Addstr.c\_str());

REPLY.publish("Relay2 OFF");

}

else

{

// REPLY.publish("ERROR");

}

}

if (subscription == &relay3)

{

//Print the new value to the serial monitor

Serial.print("relay3: ");

Serial.println((char\*) relay3.lastread);

//If the new value is "ON", turn the light on.

//Otherwise, turn it off.

if (!strcmp((char\*) relay3.lastread, "L7"))

{

digitalWrite(D7, HIGH);

REPLY.publish("PUMP-ON");

}

else if (!strcmp((char\*) relay3.lastread, "D7"))

{

digitalWrite(D7, LOW);

REPLY.publish("PUMP-OFF");

}

else

{

// REPLY.publish("ERROR");

}

}

if (subscription == &relay4)

{

//Print the new value to the serial monitor

Serial.print("relay4: ");

Serial.println((char\*) relay4.lastread);

//If the new value is "ON", turn the light on.

//Otherwise, turn it off.

if (!strcmp((char\*) relay4.lastread, "L8"))

{

digitalWrite(D8, HIGH);

sureshkumar = '0';

WriteConfig();

int Relay4 = 1;

config.Relay4 = Relay4;

EEPROM.write(403, config.Relay4);

EEPROM.commit();

// Addstr=config.relay4+"ON"; //adding two strings

// REPLY.publish(Addstr.c\_str());

REPLY.publish("Relay4 ON");

}

else if (!strcmp((char\*) relay4.lastread, "D8"))

{

digitalWrite(D8, LOW);

sureshkumar = '0';

WriteConfig();

int Relay4 = 0;

config.Relay4 = Relay4;

EEPROM.write(403, config.Relay4);

EEPROM.commit();

// Addstr=config.relay4+"OFF"; //adding two strings

//REPLY.publish(Addstr.c\_str());

REPLY.publish("Relay4 OFF");

}

else

{

// REPLY.publish("ERROR");

}

}

if (subscription == &ch2)

{

//Print the new value to the serial monitor

Serial.print("ch2: ");

Serial.println((char\*) ch2.lastread);

// config.DeviceName = ((char\*) ch2.lastread);

// WriteStringToEEPROM(306, config.DeviceName); //---------------------------------------------------------------------ADA fruuit ch2 data store in eeprom

// EEPROM.commit();

}

if (subscription == &reply)

{

//Print the new value to the serial monitor

Serial.print("reply: ");

Serial.println((char\*) reply.lastread);

//If the new value is "ON", turn the light on.Status()getip()Cpu()

//Otherwise, turn it off.

if (!strcmp((char\*) reply.lastread, "STATUS"))

{

Status();

}

else if (!strcmp((char\*) reply.lastread, "GETIP"))

{

getip();

}

void MQTT\_connect()

{

// // Stop if already connected

//if (mqtt.connected() && mqtt.ping())//ping is important for adafruit to recoonect when internet is back(note:add some time intervel for ping)

if(mqtt.connected())

//if (mqtt.connected()==1)

//if (mqtt.ping())

{

// mqtt.disconnect();

// Serial.print(mqtt.connected());

// Serial.print(mqtt.ping());

return;

}

/\*else{

Serial.print("mqtt.connected()");

Serial.print(mqtt.connected());

}\*/

int8\_t ret;

mqtt.disconnect();

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0&&(WiFi.status() == WL\_CONNECTED)) // connect will return 0 for connected//ada wifi not found

{

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

// alarm3();

delay(5000); // wait 5 seconds

// retries--;

// if (retries == 0)

// {

// ESP.reset();

// }

// digitalWrite(D8, LOW);

lcd.clear();

lcd.setCursor(0, 1);

lcd.print("NETWORK NOTFOUND");

delay(500);

// digitalWrite(D8, HIGH);

return;//ada wifi-NETWORK not found

}

Serial.println("Adafruit MQTT Connected!");

lcd.clear();

lcd.setCursor(0, 1);

lcd.print("DEVICE CONNECTED");

//REPLY.publish("DEVICE CONNECTED");

// digitalWrite(D8, HIGH);

CH1.publish("0");

}

//AUTO WIFI-ROUTER CONNECT

void initWiFi() {

WiFi.mode(WIFI\_STA);

ConfigureWifi();

Serial.print("Connecting to WiFi ..");

lcd.setCursor(0, 1);

lcd.print("Searching Router");

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

//Serial.print('.');

// alarm3();

delay(3000);

WiFi.mode(WIFI\_STA);

ConfigureWifi();

}

// Serial.println(WiFi.localIP());

//The ESP8266 tries to reconnect automatically when the connection is lost

// WiFi.setAutoReconnect(true);

// WiFi.persistent(true);

}

void frestore\_2(){

// DEFAULT CONFIG

config.ssid = "MYSSID";

config.password = "MYPASSWORD";

config.dhcp = true;

config.IP[0] = 192;config.IP[1] = 168;config.IP[2] = 1;config.IP[3] = 100;

config.Netmask[0] = 255;config.Netmask[1] = 255;config.Netmask[2] = 255;config.Netmask[3] = 0;

config.Gateway[0] = 192;config.Gateway[1] = 168;config.Gateway[2] = 1;config.Gateway[3] = 1;

config.cloud = 0;

config.mqttServer = "cloudmqtt.com";

config.mqttUser = "user name";

config.mqttPassword = "password";

config.mqttPort ="12345";

// config.mqttTopic ="Topic to subscribe"; cloud ynable to connect due to overload of memory

WriteConfig();

}

//---------------------------DUAL CORE--DUC

void Task1code( void \* pvParameters ){

// Serial.print("Task1 running on core ");

// Serial.println(xPortGetCoreID());

delay(5000);

for(;;){

server.handleClient(); //webserver in core2

solar\_v();

if(digitalRead(I4) == HIGH) {

/\* REPLY.publish("CLEANING STARTED");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("CLEANING STARTED");\*/

digitalWrite(D5, HIGH);

digitalWrite(D6, LOW);

delay(4500);

digitalWrite(D5, LOW);

digitalWrite(D6, HIGH);

delay(4300);

digitalWrite(D5, LOW);

digitalWrite(D6, LOW);

// Addstr=config.relay1+"ON"; //adding two strings

// REPLY.publish(Addstr.c\_str());

/\* REPLY.publish("DONE");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("DONE");\*/

}

if(digitalRead(I3) == HIGH) {

digitalWrite(D7, HIGH);

delay(10000);

digitalWrite(D7, LOW);

}

//interrupt

/\* if (lastButtonState == 1) {

mem.write(10,buttonState);

Serial.println("READING AT22C32---");

Serial.println(mem.read(10));

lastButtonState = 0;

digitalWrite(D4, HIGH);

delay(500);

digitalWrite(D4, LOW);

} \*/

//----------STATE CHANGE

/\* buttonState = digitalRead(I1);

if (buttonState != lastButtonState) {

if (buttonState == HIGH) {

buttonPushCounter++;

Serial.println("on");

Serial.print("number of button pushes: ");

Serial.println(buttonPushCounter);

mem.write(10,buttonPushCounter);

Serial.println("READING AT22C32---");

Serial.println(mem.read(10));

} else {

Serial.println("off");

}

digitalWrite(D4, HIGH);

delay(50);

digitalWrite(D4, LOW);

}

lastButtonState = buttonState;\*/

//----------STATE CHANGE

/\* if( digitalRead(I1)== HIGH){Serial.println("I1, HIGH");digitalWrite(D4, HIGH);}

if( digitalRead(I1)== LOW){Serial.println("I1, LOW");digitalWrite(D4, LOW);}\*/

/\* UDP();

if(digitalRead(potPin) == HIGH)

{

U\_SEN\_1();//create function sperately and call to interface multi sensor

}

U\_SEN\_2();

// delay(50);

if(servo\_flag == 0)

{

servo();

}

EMRG();\*/

/\* digitalWrite(D4, HIGH);

delay(1000);

digitalWrite(D4, LOW);

if( digitalRead(I1)== HIGH)

{

Serial.println("I1, HIGH");

}else{

Serial.println("I1, LOW");

}

if( digitalRead(I2)== HIGH) //Panic button

{

digitalWrite(D4, HIGH); //buzzer

delay(1000);

digitalWrite(D4, LOW);

}\*/

/\*if (Serial2.available() > 0)

{

// Serial.write(Serial2.read());

if(loc==1)

{

if (GPS\_RL.encode(Serial2.read())){

displayInfo();

loc=0;

}

}

}\*/

if (Serial.available() > 0){

serialstr = Serial.readString();

if (serialstr.startsWith("$RTCDUC$"))

{

// RTC\_LOOP();

}

if (serialstr.startsWith("$FRTC$"))

{

// future\_rtc();

}

if (serialstr.startsWith("$FCHK$"))

{

// Fchk();

}

if (serialstr.startsWith("$CLEAR$"))

{

lcd.clear();

delay(2000);

}

if (serialstr.startsWith("$NO$"))

{

lcd.noDisplay();

}

if (serialstr.startsWith("$YES$"))

{

lcd.display();

}

if (serialstr.startsWith("$NBL$"))

{

lcd.noBacklight();

}

if (serialstr.startsWith("$BL$"))

{

lcd.backlight();

}

if (serialstr.startsWith("$CTC$"))

{

//int m=0;

mem.write(10,000);

delay(10);

mem.write(11,000);

delay(10);

mem.write(12,000);

delay(10);

mem.write(13,000);

delay(10);

WriteConfig0();

Serial.println("READING AT22C32---");

Serial.print(mem.read(10));

Serial.print(mem.read(11));

Serial.print(mem.read(12));

Serial.println(mem.read(13));

// mem.write(14,0);

}

}

delay(1);

// Serial.print("Task1 running on core ");

// Serial.println(xPortGetCoreID());

}

}

//---------------------------DUC

//---------------ADA MQTT PING

void PING()

{

unsigned long PING\_currentMillis = millis();

if( PING\_FLAG ==1){

PING\_previousMillis = PING\_currentMillis;

PING\_FLAG =0;

}

if (PING\_currentMillis - PING\_previousMillis>= PING\_interval) {

lcd.clear();

// PING\_previousMillis = PING\_currentMillis;

/\* if (mqtt.ping())

{

Serial.println("ping to ada\_mqttt");

}\*/

mqtt.disconnect();

PING\_FLAG = 1;

Serial.println("ping to mqttt");

lcd.clear();

}

}

//---------------------GPS

void solar\_v()

{

//voltage sensor

adc\_value = analogRead(SIGNAL\_PIN);

// Read the Analog Input

   // Determine voltage at ADC input

   adc\_voltage  = (adc\_value \* ref\_voltage) / 1024.0;

   // Calculate voltage at divider input

   in\_voltage = adc\_voltage / (R2/(R1+R2));

   // Print results to Serial Monitor to 2 decimal places

  Serial.print("Input Voltage = ");

  Serial.println(in\_voltage, 2);

  // Short delay

  delay(500);

}

void solar\_I()

{

/ Vout is read 1000 Times for precision

  for(int i = 0; i < 1000; i++) {

    Vout = (Vout + (resADC \* analogRead(A0)));

    delay(1);

  }

  // Get Vout in mv

  Vout = Vout /1000;

  // Convert Vout into Current using Scale Factor

  Current = (Vout - zeroPoint)/ scale\_factor;

  // Print Vout and Current to two Current = ");

  Serial.print("Vout = ");

  Serial.print(Vout,2);

  Serial.print(" Volts");

  Serial.print("\t Current = ");

  Serial.print(Current,2);

  Serial.println(" Amps");

  delay(1000);

}

void VOLT()

{

StaticJsonBuffer<300> JSONbuffer;

JsonObject& JSONencoder = JSONbuffer.createObject();

JSONencoder["volt"] = in\_voltage;

JSONencoder["current"] = in\_current;

char JSONmessageBuffer[300];

JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));

CH2.publish(JSONmessageBuffer);

}