**CODE**

/\* Irrigation system

Version 1.0

PCF8575N & PCF8574N is 'reverse' logic in as much it SINKS current.

so HIGH is OFF and LOW is ON.

Turn OFF all pins by sending a high byte (1 bit per byte)

// OFF B11111111,B11111111 // OFF for all 16 IO

// ON B00000000,B00000000 // ON for all 16 IO

\*/

#include <Arduino.h>

#include <NodeMCU\_Pinouts.h>

#include <Wire.h>

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <time.h>

#include <EEPROM.h>

const char\* ssid = "MySSID";

const char\* password = "Password";

String host\_name = "PoC";

int timezone = (4 \* 60 \* 60); // For UTC -5.00 : -5 \* 60 \* 60 : -18000

const int EPOCH\_1\_1\_2019 = 1546300800; // For NTP - 1546300800 = 01/01/2019 @ 12:00am (UTC)

const char\* ntpServer = "pool.ntp.org";

time\_t now;

static unsigned long lastGetTimeClock; // Counter for updating UTC time after a delay

static const int ZoneOthers = 0xFF ;

const long zoneTimerInterval = 350000UL; // timer ~ 5min

unsigned long zoneTimerPrevious = 0; // timer will store last time timer was updated

int zoneTimerState = 0; // timer

int EEaddress = 0; // EEprom address

bool flag\_isAutoDay;

bool flag\_isAutoDay\_Executed;

bool flag\_isAutoDay\_notExecute = true; // Reset the flag autoIrrigation to 'Execute' when odd day is coming

bool flag\_isAutoHourReady = false;

String thisTime = "none";

String autoLastrunTime = "";

bool autoPilot = true;

int autoPilotSetZone;

int allZones\_counter;

bool allZones\_flag = false;

bool allZonesTimer\_flag = false;

ESP8266WebServer server(80);

int address = 0x20; //0100000 (7bit) //address is |0100|A0|A1|A2|

static const uint8\_t ku8TWISuccess = 0; //I2C/TWI success (transaction was successful).

static const uint8\_t ku8TWIDeviceNACK = 2; //I2C/TWI device not present (address sent, NACK received).

static const uint8\_t ku8TWIDataNACK = 3; //I2C/TWI data not received (data sent, NACK received).

static const uint8\_t ku8TWIError = 4; //I2C/TWI other error.

static const uint8\_t Zone\_1 = 0b11111110 ;

static const uint8\_t Zone\_2 = 0b11111101 ;

static const uint8\_t Zone\_3 = 0b11111011 ;

static const uint8\_t Zone\_4 = 0b11110111 ;

static const uint8\_t Zone\_5 = 0b11101111 ;

static const uint8\_t Zone\_6 = 0b11011111 ;

static const uint8\_t Zone\_7 = 0b10111111 ;

static const uint8\_t Zone\_8 = 0b01111111 ;

static const uint8\_t Zone\_off = 0b11111111 ;

String htmlpage = ""; // Web page

// ================================= Subroutines

void Relay\_off()

{

Wire.beginTransmission(address);

Wire.write(lowByte(Zone\_off));

Wire.write(lowByte(Zone\_off));

Wire.endTransmission();

zoneTimerState = 0; // disable the 'zoneTimerState - if condition' in the 'loop' function

delay(2500);

}

void Execute\_Zone(uint8\_t zone)

{

Wire.beginTransmission(address); // Set the Relay for the Zone

Wire.write(lowByte(zone));

Wire.write(lowByte(zone));

Wire.endTransmission();

zoneTimerState = 1; // 0 = off, 1 = ON // Enable the 'zoneTimerState' if condition in the 'loop' function

zoneTimerPrevious = millis(); // start the countdown using millis to compare with 'ZoneTimerNow' in 'loop' function

}

void autoMode\_onORoff(bool val)

{

if(val)

{

autoPilot = true;

}

else

{

autoPilot = false;

}

EEaddress = 0; // EEprom address of autoPilotSetting

EEPROM.write(EEaddress, autoPilot); // Write to memory

EEPROM.commit(); // written to flash

}

void webpageMain()

{

htmlpage += "<!DOCTYPE html>";

htmlpage += "<head><title>IoT Irrigation</title>";

htmlpage += "<style>";

htmlpage += "#header {background-color:blue; font-family:Tahoma,Verdana,Serif,sans-serif; width:1024px; padding:5px; color:white; text-align:center; }";

htmlpage += "#section {background-color:#C2DEFF; font-family:Tahoma,Verdana,Serif,sans-serif; width:1024px; padding:5px; color:blue; font-size:12px;}";

htmlpage += "#footer {background-color:steelblue; font-family:Tahoma,Verdana,Serif,sans-serif; width:1024px; padding:5px; color:white; font-size:9px; clear:both;}";

htmlpage += "</style></head>";

htmlpage += "<script type=\"text/javascript\"> function reloadPage() {location.reload(true)} </script>";

htmlpage += "<body>";

htmlpage += "<div id=\"header\"><h1>Irrigation System 1.0</h1></div>";

htmlpage += "<div id=\"section\"><h3>";

htmlpage += "<form action=\"/\" method=\"POST\">";

htmlpage += "<center>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone1\"> Zone-1";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone2\"> Zone-2";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone3\"> Zone-3";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone4\"> Zone-4";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone5\"> Zone-5";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone6\"> Zone-6";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone7\"> Zone-7";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zone8\"> Zone-8";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"zoneX\"> Execute all zones";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"stop\"> Stop irrigation";

htmlpage += "<br><br>";

htmlpage += "<input type=\"radio\" name=\"zone\" value=\"auto-OR-man\"> Automatic or manual mode";

htmlpage += "<br><br>";

}

void webpage\_Execute()

{

String strAutoPilotMode = "";

if(autoPilot)

{

strAutoPilotMode = "automatic";

}

else

{

strAutoPilotMode = "manual";

}

htmlpage = ""; // reset webpage to this form - else will have multiple pages...

webpageMain();

htmlpage += "<br>";

htmlpage += "<input type=\"submit\">";

htmlpage += "<br><br>";

htmlpage += "<input type=\"button\" value=\"Refresh\" onclick=\"reloadPage()\" />";

htmlpage += "</form>";

htmlpage += "<br><br>";

htmlpage += "Irrigation mode : " + strAutoPilotMode + "";

htmlpage += "<br><br>";

htmlpage += "Last execution time " + autoLastrunTime + "";

htmlpage += "<br><br>";

htmlpage += "Last check UTC time " + thisTime + "";

htmlpage += "<br>";

htmlpage += "</center>";

htmlpage += "</h5>";

htmlpage += "</div></body></html>";

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

}

void returnFail(String msg)

{

server.sendHeader("Connection", "close");

server.sendHeader("Access-Control-Allow-Origin", "\*");

server.send(500, "text/plain", msg + "\r\n");

}

void handleSubmit()

{

if (!server.hasArg("zone")) return returnFail("BAD ARGS");

String val\_Zone = server.arg("zone");

if (val\_Zone == "zone1") {

Relay\_off();

Execute\_Zone(Zone\_1);

webpage\_Execute();

}

else if (val\_Zone == "zone2") {

Relay\_off();

Execute\_Zone(Zone\_2);

webpage\_Execute();

}

else if (val\_Zone == "zone3") {

Relay\_off();

Execute\_Zone(Zone\_3);

webpage\_Execute();

}

else if (val\_Zone == "zone4") {

Relay\_off();

Execute\_Zone(Zone\_4);

webpage\_Execute();

}

else if (val\_Zone == "zone5") {

Relay\_off();

Execute\_Zone(Zone\_5);

webpage\_Execute();

}

else if (val\_Zone == "zone6") {

Relay\_off();

Execute\_Zone(Zone\_6);

webpage\_Execute();

}

else if (val\_Zone == "zone7") {

Relay\_off();

Execute\_Zone(Zone\_7);

webpage\_Execute();

}

else if (val\_Zone == "zone8") {

Relay\_off();

Execute\_Zone(Zone\_8);

webpage\_Execute();

}

else if (val\_Zone == "zoneX") { // Execute all zones

Relay\_off();

allZones\_flag = true;

allZonesTimer\_flag = true;

allZones\_counter = 1; // Execution for all zones will be set to zone 1

webpage\_Execute();

}

else if (val\_Zone == "stop") {

Relay\_off();

allZones\_flag = false;

allZonesTimer\_flag = false;

webpage\_Execute();

}

else if (val\_Zone == "auto-OR-man") { // Automatic or manual mode

bool SetAutomode;

if(autoPilot)

{SetAutomode = false;}

else

{SetAutomode = true;}

autoMode\_onORoff(SetAutomode);

webpage\_Execute();

}

else {webpage\_Execute(); }

}

void handleRoot()

{

if (server.hasArg("zone"))

{

webpage\_Execute();

handleSubmit();

}

else {

webpage\_Execute();

}

}

void returnOK()

{

server.sendHeader("Connection", "close");

server.sendHeader("Access-Control-Allow-Origin", "\*");

server.send(200, "text/plain", "OK\r\n");

}

void handleNotFound()

{

String message = "File Not Found\n\n";

message += "URI: ";

message += server.uri();

message += "\nMethod: ";

message += (server.method() == HTTP\_GET)?"GET":"POST";

message += "\nArguments: ";

message += server.args();

message += "\n";

for (uint8\_t i=0; i<server.args(); i++){

message += " " + server.argName(i) + ": " + server.arg(i) + "\n";

}

server.send(404, "text/plain", message);

}

void updateClock()

{

if ( (millis() - lastGetTimeClock) >= 900000UL) // every 15min = 900000UL - updates clock time every 30min = 1800000UL - update every hour = 3600000UL

{

time\_t now;

struct tm \* timeNow;

time(&now);

timeNow = localtime(&now);

int setDay = timeNow->tm\_mday;

int setHour = timeNow->tm\_hour;

int setMin = timeNow->tm\_min;

char snumDay[5];

char snumHour[5];

char snumMin[5];

String thisDay = itoa(setDay, snumDay, 10); // use only to display in html page

String thisHour = itoa(setHour, snumHour, 10); // use only to display in html page

String thisMin = itoa(setMin, snumMin, 10); // use only to display in html page

thisTime = "DayOfWeek:" + thisDay + " Time:" + thisHour + "-" + thisMin;

if(setDay % 2 == 0) // if day is even (ex.: 2, 4, 8, 10, etc.) disable auto pilot

{

flag\_isAutoDay = false; // autoPilot = false;

flag\_isAutoDay\_notExecute = true; // Reset the flag autoIrrigation to 'Execute' when odd day is coming

flag\_isAutoHourReady = false;

}

else // day is odd (ex.: 5, 11, 15, 17, etc.) Set auto Pilot to 'true'

{

flag\_isAutoDay = true;

if(setHour == 6)

{

flag\_isAutoHourReady = true;

}

else{flag\_isAutoHourReady = false;} // change flag value - before and after the 'authorize set time'

}

lastGetTimeClock = millis(); // Reset lastGetTimeClock counter

}

}

// ================================= End SubRoutines

void setup()

{

Serial.begin(115200); // Serial Window (debugging)

EEPROM.begin(32);

Wire.begin(); // I2C Two Wire initialisation

// PCF8575 - Turn OFF all pins by sending a high byte (1 bit per byte)

Wire.beginTransmission(address);

Wire.write(lowByte(Zone\_off));

Wire.write(highByte(Zone\_off));

Wire.endTransmission();

// ----- wifi

WiFi.hostname(host\_name);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

delay(1200);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

Serial.println("hostname : ");

Serial.println(WiFi.hostname());

// ----- end of wifi

// ----- Set Time with UTP

configTime(timezone, 0, ntpServer);

while (now < EPOCH\_1\_1\_2019)

{

now = time(nullptr);

delay(500);

Serial.print("\*");

}

// Read EEProm memory and set irrigation for automatic or manual mode

EEaddress = 0;

autoPilot = EEPROM.read(EEaddress);

// Enable server mode

server.on("/", handleRoot);

server.onNotFound(handleNotFound);

server.begin(); //Start server

Serial.println("HTTP server started");

}

void loop(void)

{

updateClock();

server.handleClient();

if(autoPilot) // Enable or Disable the 'Automatic irrigation'

{

if(flag\_isAutoDay && flag\_isAutoDay\_notExecute && flag\_isAutoHourReady) // start irrigation system

{

flag\_isAutoDay\_notExecute = false; // Prevent re-executing this 'if statement'

allZones\_flag = true; // Ready to Execute all zones (1 to 8)

allZonesTimer\_flag = true; // Ready - Permit to re-enter the all zones 'if statement' when zone X delay is finish

allZones\_counter = 1; // Reset order (zone 1 to 8) to start at zone 1

autoLastrunTime = thisTime; // Record time when AutoPilot is ready to execute all zones

}

} // end of if condition

if(zoneTimerState == 1) // if irrigation is running - validate time left from timer

{

unsigned long ZoneTimerNow = millis();

if(ZoneTimerNow - zoneTimerPrevious >= zoneTimerInterval )

{

zoneTimerPrevious = ZoneTimerNow;

Relay\_off();

allZonesTimer\_flag = true; // USED only by allZones\_flag 'if condition' in void loop - Will wait until timer is finish

// before executing another zone...

}

}

if(allZones\_flag == true && allZonesTimer\_flag == true)

{

switch (allZones\_counter)

{

case 1:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_1);

break;

case 2:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_2);

break;

case 3:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_3);

break;

case 4:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_4);

break;

case 5:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_5);

break;

case 6:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_6);

break;

case 7:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_7);

break;

case 8:

allZones\_counter++ ; // increment counter for next zone to execute after timer

allZonesTimer\_flag = false; // disable 'if condition' because timer is not finished and not starting another zone

Execute\_Zone(Zone\_8);

allZones\_flag = false; // All zones are executed. Now disable the 'if condition - allZones\_flag' from void loop

break;

}

} // end for executing allzones

}