**ภาคผนวก ก**

**โปรแกรมควบคุมการทำงาน**

**ก.1 โนดเอ็มคิวทีที**

#include <WiFiClient.h>

#include <PubSubClient.h>

#include <painlessMesh.h>

const char\* ssid = "Proj\_IoT";

const char\* password = "testproj";

const char\* mqttServer = "192.168.2.1";

const int mqttPort = 1883;

const char\* mqttUser = "testmymqtt";

const char\* mqttPassword = "testmymqtt";

#define MESH\_PREFIX "whateverYouLike"

#define MESH\_PASSWORD "somethingSneaky"

#define MESH\_PORT 5555

#define HOSTNAME "MQTT\_Bridge"

const char\* Current1;

const char\* Current2;

const char\* Current3;

const char\* Voltage12;

const char\* Voltage23;

const char\* Voltage31;

const char\* Voltage1;

const char\* Voltage2;

const char\* Voltage3;

const char\* Power1;

const char\* Power2;

const char\* Power3;

const char\* PowerF1;

const char\* PowerF2;

const char\* PowerF3;

String \* str = NULL;

char I1[50],I2[50],I3[50];

char Volt12[50],Volt23[50],Volt31[50];

char Volt1[50],Volt2[50],Volt3[50];

char P1[50],P2[50],P3[50];

char PF1[50],PF2[50],PF3[50];

void receivedCallback( const uint32\_t &from, const String

&msg );

void mqttCallback(char\* topic, byte\* payload, unsigned int

length);

WiFiClient espClient;

PubSubClient client(espClient);

painlessMesh mesh;

void reconnect(){

while(!client.connected()){

Serial.println("Attempting MQTT Connection...");

String clientId = "ESP32Client-";

clientId += String(random(0xffff), HEX);

if(client.connect(clientId.c\_str(), mqttUser,

mqttPassword)){

Serial.println("Connected");

client.publish("outTopic", "Connected");

client.subscribe("inTopic");

}

else{

Serial.print("Failed, rc = ");

Serial.print(client.state());

Serial.println("Try again in 2 seconds");

delay(1000);

}

}

}

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

length) {

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

String msg = "";

int i = 0;

while(i < length){

msg += (char)payload[i++];

}

Serial.print("Message Require : ");

Serial.println(msg);

//////////////////////////////////Current////////////////////////////////////////

if(msg == "2999"){

Serial.print("Message Response : ");

Serial.println(I1);

client.publish("Response/Current", I1);

}

else if(msg == "3001"){

Serial.print("Message Response : ");

Serial.println(I2);

client.publish("Response/Current", I2);

}

else if(msg == "3003"){

Serial.print("Message Response : ");

Serial.println(I3);

client.publish("Response/Current", I3);

}

///////////////////////////////////VoltageToLine///////////////////////////////////////

else if(msg == "3019"){

Serial.print("Message Response : ");

Serial.println(Voltage12);

client.publish("Response/VoltageToLine", Volt12);

}

else if(msg == "3021"){

Serial.print("Message Response : ");

Serial.println(Voltage23);

client.publish("Response/VoltageToLine", Volt23);

}

else if(msg == "3023"){

Serial.print("Message Response : ");

Serial.println(Voltage31);

client.publish("Response/VoltageToLine", Volt31);

}

//////////////////////////////////VoltageToNuetron////////////////////////////////////////

else if(msg == "3027"){

Serial.print("Message Response : ");

Serial.println(Volt1);

client.publish("Response/VoltageToNuetron", Volt1);

}

else if(msg == "3029"){

Serial.print("Message Response : ");

Serial.println(Volt2);

client.publish("Response/VoltageToNuetron", Volt2);

}

else if(msg == "3031"){

Serial.print("Message Response : ");

Serial.println(Volt3);

client.publish("Response/VoltageToNuetron", Volt3);

}

//////////////////////////////////Power///////////////////////////////////////

else if(msg == "3053"){

Serial.print("Message Response : ");

Serial.println(P1);

client.publish("Response/Power", P1);

}

else if(msg == "3055"){

Serial.print("Message Response : ");

Serial.println(P2);

client.publish("Response/Power", P2);

}

else if(msg == "3057"){

Serial.print("Message Response : ");

Serial.println(P3);

client.publish("Response/Power", P3);

}

////////////////////////////////Power Factor///////////////////////////////////////

else if(msg == "3077"){

Serial.print("Message Response : ");

Serial.println(PF1);

client.publish("Response/PowerF", PF1);

}

else if(msg == "3079"){

Serial.print("Message Response : ");

Serial.println(PF2);

client.publish("Response/PowerF", PF2);

}

else if(msg == "3081"){

Serial.print("Message Response : ");

Serial.println(PF3);

client.publish("Response/PowerF", PF3);

}

}

void setup(){

Serial.begin(115200);

mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION );

mesh.init( MESH\_PREFIX, MESH\_PASSWORD, MESH\_PORT,

WIFI\_AP\_STA, 6 );

mesh.onReceive(&receivedCallback);

mesh.stationManual(ssid, password);

mesh.setHostname(HOSTNAME);

mesh.setRoot(true);

mesh.setContainsRoot(true);

WiFi.begin(ssid, password);

Serial.println();

Serial.print("Wait for WiFi...");

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

Serial.print(".");

delay(500);

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

client.setServer(mqttServer, 1883);

client.setCallback(callback);

}

void loop(){

if (!client.connected()) {

reconnect();

}

mesh.update();

client.loop();

//delay(1000);

}

void receivedCallback( const uint32\_t &from, const String

&msg ) {

Serial.printf("bridge: Received from %u msg=%s\n", from,

msg.c\_str());

client.publish("Send", msg.c\_str());

////////////////////////////////////////// https://github.

com/TridentTD/TridentTD\_StringSplit ///////////////////////

String input = msg.c\_str();

//String \* str = NULL;

int counter = input.td\_split("[],",&str);

str[0].toCharArray(I1,50);

str[1].toCharArray(I2,50);

str[2].toCharArray(I3,50);

str[3].toCharArray(Volt12,50);

str[4].toCharArray(Volt23,50);

str[5].toCharArray(Volt31,50);

str[6].toCharArray(Volt1,50);

str[7].toCharArray(Volt2,50);

str[8].toCharArray(Volt3,50);

str[9].toCharArray(P1,50);

str[10].toCharArray(P2,50);

str[11].toCharArray(P3,50);

str[12].toCharArray(PF1,50);

str[13].toCharArray(PF2,50);

str[14].toCharArray(PF3,50);

}

**ก.2 โนดม็อดบัส**

#include <WiFi.h>

#include <ModbusMaster.h>

#include <SoftwareSerial.h>

#include <painlessMesh.h>

#define MESH\_PREFIX "whateverYouLike"

#define MESH\_PASSWORD "somethingSneaky"

#define MESH\_PORT 5555

#define Address 1

#define Total\_Regs 15

//////////////////Current//////////////////////////

#define regCurrent\_A 2999

#define regCurrent\_B 3001

#define regCurrent\_C 3003

///////////////////Voltage L-L////////////////////

#define regVoltage\_AB 3019

#define regVoltage\_BC 3021

#define regVoltage\_CA 3023

///////////////////Voltage L-N////////////////////

#define regVoltage\_AN 3027

#define regVoltage\_BN 3029

#define regVoltage\_CN 3031

/////////////////Power////////////////////////////

#define regPower\_A 3053

#define regPower\_B 3055

#define regPower\_C 3057

/////////////////PF///////////////////////////////

#define regPF\_A 3077

#define regPF\_B 3079

#define regPF\_C 3081

uint16\_t regs3phase[5][3] = {

{regCurrent\_A,

regCurrent\_B, regCurrent\_C},

{regVoltage\_AB,

regVoltage\_BC, regVoltage\_CA},

{regVoltage\_AN,

regVoltage\_BN, regVoltage\_CN},

{regPower\_A,

regPower\_B, regPower\_C},

{regPF\_A,

regPF\_B, regPF\_C}

};

String a[5][3] = {{"I1","I2","I3"},

{"V12","V23","V31"},

{"V1","V2","V3"},

{"P1","P2","P3"},

{"PF1","PF2","PF3"}

};

String DATA\_METER[5][3];

int ii=0,jj=0;

//String msg;

union IntToFloat

{

uint32\_t asInt[2];

float asFloat = 0;

};

IntToFloat dataConverter;

void sendMessage();

void receivedCallback(uint32\_t from, String & msg);

void newConnectionCallback(uint32\_t nodeId);

void changedConnectionCallback();

void nodeTimeAdjustedCallback(int32\_t offset);

void delayReceivedCallback(uint32\_t from, int32\_t delay);

ModbusMaster node;

SoftwareSerial swSerial(22, 23);

Scheduler userScheduler;

painlessMesh mesh;

void sendMessage() ; // Prototype so PlatformIO doesn't

complain

Task taskSendMessage( TASK\_SECOND \* 1 , TASK\_FOREVER,

&sendMessage );

void setup() {

Serial.begin(38400);

swSerial.begin(38400);

mesh.setDebugMsgTypes(ERROR | DEBUG);

mesh.init( MESH\_PREFIX, MESH\_PASSWORD, &userScheduler,

MESH\_PORT );

mesh.onReceive(&receivedCallback);

mesh.onNewConnection(&newConnectionCallback);

mesh.onChangedConnections(&changedConnectionCallback);

mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);

mesh.onNodeDelayReceived(&delayReceivedCallback);

userScheduler.addTask( taskSendMessage );

taskSendMessage.enable();

}

void loop() {

userScheduler.execute();// it will run the user scheduler

as well

GET\_METER();

mesh.update();

}

float Read\_Meter\_Float(char addr , uint16\_t REG){

float i=0;

uint8\_t j,result;

node.begin(addr, swSerial);

result = node.readHoldingRegisters(REG,2);

delay(150);

if (result == node.ku8MBSuccess) {

dataConverter.asInt[0] = node.getResponseBuffer(0);

dataConverter.asInt[1] = node.getResponseBuffer(1);

dataConverter.asInt[0] = dataConverter.asInt[0] << 16;

dataConverter.asInt[0] = dataConverter.asInt[0] +

dataConverter.asInt[1];

i = dataConverter.asFloat;

//Serial.println("OK");

return i;

}

else {

Serial.print("Failed REG --->");

Serial.println(REG , HEX);

delay(500);

return 0;

}

}

void GET\_METER() {

delay(500);

DATA\_METER[ii][0] = Read\_Meter\_Float(Address,

regs3phase[ii][0]);

DATA\_METER[ii][1] = Read\_Meter\_Float(Address,

regs3phase[ii][1]);

DATA\_METER[ii][2] = Read\_Meter\_Float(Address,

regs3phase[ii][2]);

if(ii++ == 4)

ii=0;

}

void sendMessage(){

String msg = "";

msg += DATA\_METER[0][0]; msg += ","; msg +=

DATA\_METER[0][1]; msg += ","; msg += DATA\_METER[0][2];

msg += ",";

msg += DATA\_METER[1][0]; msg += ","; msg +=

DATA\_METER[1][1]; msg += ","; msg += DATA\_METER[1][2];

msg += ",";

msg += DATA\_METER[2][0]; msg += ","; msg +=

DATA\_METER[2][1]; msg += ","; msg += DATA\_METER[2][2];

msg += ",";

msg += DATA\_METER[3][0]; msg += ","; msg +=

DATA\_METER[3][1]; msg += ","; msg += DATA\_METER[3][2];

msg += ",";

msg += DATA\_METER[4][0]; msg += ","; msg +=

DATA\_METER[4][1]; msg += ","; msg += DATA\_METER[4][2];

mesh.sendBroadcast(msg);

Serial.

println("---------------------------------------------------

------------------------------------------------------------

----");

Serial.print("Send message: ");

Serial.println(msg);

Serial.

println("---------------------------------------------------

------------------------------------------------------------

----");

//taskSendMessage.setInterval(TASK\_SECOND \* 5);

//taskSendMessage.setInterval( random(TASK\_SECOND \* 1,

TASK\_SECOND \* 5));

}

void receivedCallback( uint32\_t from, String &msg ) {

Serial.

println("###################################################

############################################################

####");

Serial.printf("ModbusNode: Received from %u msg=%s\n",

from, msg.c\_str());

Serial.

println("###################################################

############################################################

####");

}

void newConnectionCallback(uint32\_t nodeId) {

Serial.printf("--> ModbusNode: New Connection, nodeId =

%u\n", nodeId);

}

void changedConnectionCallback() {

Serial.printf("Connection list:");

}

void nodeTimeAdjustedCallback(int32\_t offset) {

Serial.printf("Adjusted time %u. Offset = %d\n", mesh.

getNodeTime(), offset);

}

void delayReceivedCallback(uint32\_t from, int32\_t delay) {

Serial.printf("Delay to node %u is %d us\n", from, delay);

}

**ก.3 โนดเมชเน็ตเวิร์ค**

#include <painlessMesh.h>

#define LED 2

#define BLINK\_PERIOD 3000

#define BLINK\_DURATION 100

#define MESH\_SSID "MeshNetwork\_for\_level\_plant"

#define MESH\_PASSWORD "somethingSneaky"

#define MESH\_PORT 5555

#define nodecheck D6

#define changePID D7

void sendMessage();

void receivedCallback(uint32\_t from, String & msg);

void newConnectionCallback(uint32\_t nodeId);

void changedConnectionCallback();

void nodeTimeAdjustedCallback(int32\_t offset);

void delayReceivedCallback(uint32\_t from, int32\_t delay);

Scheduler userScheduler;

painlessMesh mesh;

bool calc\_delay = false;

SimpleList<uint32\_t> nodes;

void sendMessage() ;

Task taskSendMessage( TASK\_SECOND \* 1, TASK\_FOREVER, &sendMessage ); // start with a one second interval

Task blinkNoNodes;

bool onFlag = false;

float sp,pv,kp,ki;

int timenode;

float Error,MV;

float Ierror=0.0;

String MV\_SHOW;

int PWM=0;

int last;

int photocheck;

int LDR;

int x=2;

int F=1;

void setup()

{

Serial.begin(9600);

pinMode(LED, OUTPUT);

pinMode(changePID, INPUT);

pinMode(nodecheck, OUTPUT);

pinMode(D8, INPUT);

mesh.setDebugMsgTypes(ERROR | DEBUG | CONNECTION); // set before init() so that you can see startup messages

mesh.init(MESH\_SSID, MESH\_PASSWORD, &userScheduler, MESH\_PORT);

mesh.onReceive(&receivedCallback);

mesh.onNewConnection(&newConnectionCallback);

mesh.onChangedConnections(&changedConnectionCallback);

mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);

mesh.onNodeDelayReceived(&delayReceivedCallback);

userScheduler.addTask( taskSendMessage );

taskSendMessage.enable();

blinkNoNodes.set(BLINK\_PERIOD, (mesh.getNodeList().size() + 1) \* 2, []()

{

if (onFlag)

onFlag = false;

else

onFlag = true;

blinkNoNodes.delay(BLINK\_DURATION);

if (blinkNoNodes.isLastIteration())

{

blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) \* 2);

blinkNoNodes.enableDelayed(BLINK\_PERIOD -

(mesh.getNodeTime() % (BLINK\_PERIOD\*1000))/1000);

}

});

userScheduler.addTask(blinkNoNodes);

blinkNoNodes.enable();

}

void loop()

{

userScheduler.execute();

mesh.update();

digitalWrite(LED, !onFlag);

digitalWrite(nodecheck,HIGH);

}

void sendMessage()

{

String msg = "MeshRadarON";

photocheck = analogRead(A0);

if (calc\_delay)

{

SimpleList<uint32\_t>::iterator node = nodes.begin();

while (node != nodes.end())

{

mesh.startDelayMeas(\*node);

node++;

}

calc\_delay = false;

}

}

void receivedCallback(uint32\_t from, String & msg)

{

Serial.printf("Received from %u msg=%s\n", from, msg.c\_str());

float k,j,o,y,h,SPP,Kpp,Kii,PVV;

float TIM;

if((msg.length())<8)

{

for (y = 0; y < msg.length(); y++)

{

if (msg.substring(y, y + 1) == ":")

{

PVV = msg.substring(1, y).toFloat();

pv=PVV;

break;

}

}

}

if((msg.length())>8)

{

for (j = 0; j < msg.length(); j++)

{

if (msg.substring(j, j + 1) == ",")

{

SPP = msg.substring(1, j).toFloat();

k = j+1;

sp=SPP\*10.23;

break;

}

}

for (k ; k < msg.length(); k++)

{

if (msg.substring(k, k + 1) == ",")

{

Kpp = msg.substring(j+1,k).toFloat();

o = k+1;

kp=Kpp;

break;

}

}

for (o ; o < msg.length(); o++)

{

if (msg.substring(o, o + 1) == ",")

{

Kii = msg.substring(k+1,o).toFloat();

h = o+1;

ki=Kii;

break;

}

}

for (h ; h < msg.length(); h++)

{

if (msg.substring(h, h + 1) == "]")

{

TIM = msg.substring(o+1,h).toFloat();

timenode = TIM;

break;

}

}

}

Error = sp-pv;

Ierror = Ierror + (Error/50);

if(Ierror>1000) Ierror = 1000;

if(Ierror<-1000) Ierror = -1000;

MV = (Error\*kp)+(Ierror\*ki);

if(MV >= 1023) MV = 1023;

if(MV <= 0) MV = 0;

Serial.printf("pv:%.2f ",pv);

Serial.printf("sp:%.2f ",sp);

Serial.printf("kp:%.2f ",kp);

Serial.printf("ki:%.2f\n",ki);

Serial.printf("Error:%.2f",Error);

Serial.printf(" Ierror:%.2f",Ierror);

Serial.printf(" MV:%.2f",MV);

Serial.printf(" LDR = %d\n",photocheck);

int duty = map(MV,0,1023,0,100);

Serial.printf(" duty:%d",duty);

Serial.printf(" timenode:%d\n",timenode);

x=timenode;

if (x==F){Serial.printf(" er\n");}

else{F=x; Serial.printf(" ok\n");}

PWM=0;

if(digitalRead(changePID)==LOW || (photocheck<950) )

{

PWM = MV;

Serial.printf("-------------------MV out!!!\n");

//Serial.printf("---PWM: %d\n",PWM);

}

analogWrite(D3,PWM);

}

void newConnectionCallback(uint32\_t nodeId)

{

onFlag = false;

blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) \* 2);

blinkNoNodes.enableDelayed(BLINK\_PERIOD - (mesh.getNodeTime() % (BLINK\_PERIOD\*1000))/1000);

Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);

}

void changedConnectionCallback()

{

Serial.printf("Changed connections %s\n", mesh.subConnectionJson().c\_str());

onFlag = false;

blinkNoNodes.setIterations((mesh.getNodeList().size() + 1) \* 2);

blinkNoNodes.enableDelayed(BLINK\_PERIOD - (mesh.getNodeTime() % (BLINK\_PERIOD\*1000))/1000);

nodes = mesh.getNodeList();

Serial.printf("Num nodes: %d\n", nodes.size());

Serial.printf("Connection list:");

SimpleList<uint32\_t>::iterator node = nodes.begin();

while (node != nodes.end())

{

Serial.printf(" %u", \*node);

node++;

}

Serial.println();

calc\_delay = true;

}

void nodeTimeAdjustedCallback(int32\_t offset)

{

Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);

}

void delayReceivedCallback(uint32\_t from, int32\_t delay)

{

Serial.printf("Delay to node %u is %d us\n", from, delay);

}