ISTANBUL TECHNICAL UNIVERSITY

Faculty of Electric and Electronics

Control and Automation Engineering

KON435E

Industrial Data Communications

FINAL PROJECT

Github

https://github.com/gsokky/automated-temperature-plc-system

Group 8

Yunus Akdal – 040190606

Gökhan Savaş – 040180609

Emirhan Ersöz – 040180607

Yusuf Eren Kılıç – 040190641

Onur Safa Gündoğdu – 040170437

CONTENTS

Aim of the Project 3
Server ESP Code 4
Client ESP Code: 9
Part of Node-red 12
Node-red Dashboard 16
User Interface 17

Aim of the Project

The aim is to create communication networks between esp32s with UDP protocols, the node-red server and an esp32 with MQTT protocol, and finally node-red server and Allen Bradley PLC through Ethernet IP.

Objectives

- Read sensor data(temperature) from esp32 and classify it as Server.
- Send read data to node-red with MQTT protocol.
- Form system parameters on node-red and send them with temperature input to Allen-Bradled PLC through ethernet IP.
- Capture the output of the system formed in PLC, compare it with input temperature and calculate the error, calculate efficiency and send output to the Server esp32.
- Send output data to Client esp32 with UDP protocol.
- Set led 1 high if the system is working and set led 2 high if the output of the system is bigger than the temperature value.

Circuit Scheme

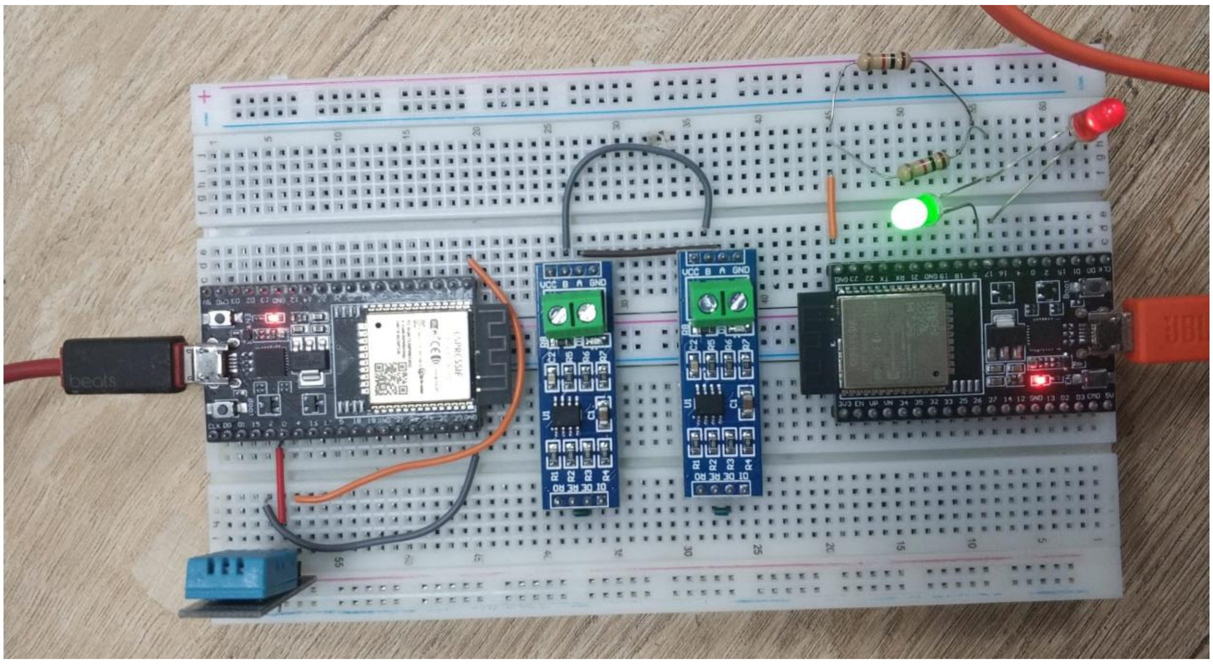


Figure 1- Constructed circuit for final. Photo was taken when the system was on*

*MAX485s are not used for the final project

Server ESP Code

```
 //External and Arduino libraries, local network settings for UDP
communication and node-red server address are set:
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <DHT.h>
#include <ArduinoJson.h>

//Definition of pin and type of the sensor
#define DHTPIN  2
#define DHTTYPE DHT11

//Setting local IP adress, gateway, subnet and DNS servers
IPAddress local_IP(192, 168, 137, 184);
IPAddress gateway(192, 168, 137, 1);

IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);

//WiFi network SSID and password
const char* ssid = "Redmi";
const char* password = "gnrn5080";

//MQTT server adress
const char* mqtt_server = "160.75.154.100";

//UDP adress and port
const char * udpAddress = "192.168.137.185";
const int udpPort = 44444;

//Create UDP instance
WiFiUDP udp;
DHT dht(DHTPIN, DHTTYPE);

WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
String startSystem = "false";
String sysOut;
String tsgo;

bool sendSysTempFlag = 0;
bool tsFlag = 0;

float temperature = 0;
float humidity = 0;

void reconnect() ;
void setup_wifi();
void callback(char* topic, byte* message, unsigned int length);
void send_udp(String header,String msg,int port,char * sendAddress);

// Setup configuration for the server esp
void setup() {
  Serial.begin(115200);

  dht.begin();

  setup_wifi();
  udp.begin(udpPort);

  client.setServer(mqtt_server, 1884);
  client.setCallback(callback);
}

//Reading temperature, forming connections and sending sensor data to the
client
void loop() {
  // Check if the client is connected to the server
  if (!client.connected())
    reconnect();

  //Check for the new messages
  client.loop();

  if(tsFlag == 1)
  {
    tsFlag = 0;
    client.publish("group8/tscome", tsgo.c_str());
  }

  if(startSystem=="true")
  {
    long now = millis();
    // If it has been more than 1 second since the last message was sent
    if (now - lastMsg > 1000)
    {
      lastMsg = now;
      temperature = dht.readTemperature();

      if (isnan(temperature))
        Serial.println("Failed to read from DHT sensor!");
      else
      {
        // Convert the temperature to a string and publish it to the server
        char tempString[8];
        dtostrf(temperature, 2, 2, tempString);
        client.publish("group8/temperature", tempString);
        String str(tempString);
        send_udp("DHTtemp:",str,udpPort,(char *)udpAddress);
      }

      // Reading humidity from sensor
      humidity = dht.readHumidity();
      if (isnan(humidity))
        Serial.println("Failed to read from DHT sensor!");
      else
      {
        char humString[8];
        dtostrf(humidity, 2, 2, humString);
        client.publish("group8/humidity", humString);
      }
      if(sendSysTempFlag == 1)
      {
        sendSysTempFlag = 0;
        send_udp("symsout:",sysOut,udpPort,(char *)udpAddress);
      }
    }

  }
}

// Send data over UDP
void send_udp(String header,String msg,int port,char * sendAddress)
{
  //Start the packet with the specified address and port
  udp.beginPacket(sendAddress, port);
  char buffer[50] = "";

  // Concatenate the header and message to the buffer
  strcat(buffer,header.c_str());
  strcat(buffer,msg.c_str());

  Serial.println(buffer);

  udp.write((uint8_t *)buffer,16);
  // Close and send the packet
  udp.endPacket();
}
```

```
// Creates WiFi connection with the client for UDP communication
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// When a message is received from MQTT server, callback function is called
void callback(char* topic, byte* message, unsigned int length) {
  String messageTemp = "";

  // Convert incoming byte message to a string
  for (int i = 0; i < length; i++) {
    messageTemp += (char)message[i];
  }
  Serial.println();

  // Check the topic of the incoming message
  if (String(topic) == "group8/start") {
    startSystem = messageTemp;
  }
  else if (String(topic) == "group8/SystemOut") {
    sendSysTempFlag = 1;
    sysOut = messageTemp;
  }
  else if (String(topic) == "group8/tsgo") {
    tsFlag = 1;
    tsgo = messageTemp;

  }
}
```

```
// Forming reconnection with the client if the connection is lost
void reconnect()
{
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32Client","iturockwell","963258741")) {
      Serial.println("connected");
      client.subscribe("group8/start");
      client.subscribe("group8/SystemOut");
      client.subscribe("group8/tsgo");

    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Client ESP Code:

```
 #include <WiFi.h>

// WiFi network SSID and password
const char* ssid     = "Redmi";
const char* password = "gnrn5080";

char * DHTtemp;
char * sysout;

double DHTtempNumb;
double sysoutNumb;

uint8_t buffer[50];

// Pins for LEDS
const int ledPin1 = 16;
const int ledPin2 = 17;

// Indicates whether system is working or not
bool isWorking = 0;

long timer = 0;

// Setting local IP adress, gateway, subnet and DNS servers
IPAddress local_IP(192, 168, 137, 185);
IPAddress gateway(192, 168, 137, 1);

IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);

const char * udpAddress = "192.168.137.184";
const int udpPort = 44444;

// Create UDP instance
WiFiUDP udp;

//Settling the connetion with Server esp through local network
void setup() {

  // Start serial communication
  Serial.begin(115200);

  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);

  // Configures static IP address
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("STA Failed to configure");
  }

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  udp.begin(udpPort);

}

void loop(){
  // Clear buffer for new data
  memset(buffer, 0, 50);

  // Check incoming data on UDP
  udp.parsePacket();
  if(udp.read(buffer, 16) > 0)
  {
    isWorking = 1;
    timer = millis();

    char * header;
    char * msg;

   // Extract the header and message from the received data
    header = strtok((char*)buffer, ":");
    msg = strtok(NULL, ":");

    Serial.print((char *)header);
    Serial.print(" ");
    Serial.println((char *)msg);

   // Check the header
    if(!strcmp("DHTtemp",header))
    {
      DHTtemp = strdup(msg);
      DHTtempNumb = strtod(DHTtemp,NULL);
    }
    else if(!strcmp("symsout",header))
    {
      sysout = strdup(msg);
      sysoutNumb = strtod(sysout,NULL);
    }
  }

  if(isWorking)
  {
    // Check if the time since the last message is greater than 5 seconds
    if(abs(millis() - timer) > 5000)
      isWorking = 0;
    else
    {
      //Check if system output temperature is greater than DHT sensor
temperature
      if(sysoutNumb > DHTtempNumb)
        digitalWrite(ledPin2, HIGH);
      else
        digitalWrite(ledPin2, LOW);

      digitalWrite(ledPin1, HIGH);
    }
  }
  else
  // Indicates that the system is not working
  {
     digitalWrite(ledPin1, LOW);
     digitalWrite(ledPin2, LOW);
  }
}
```

Part of Node-red

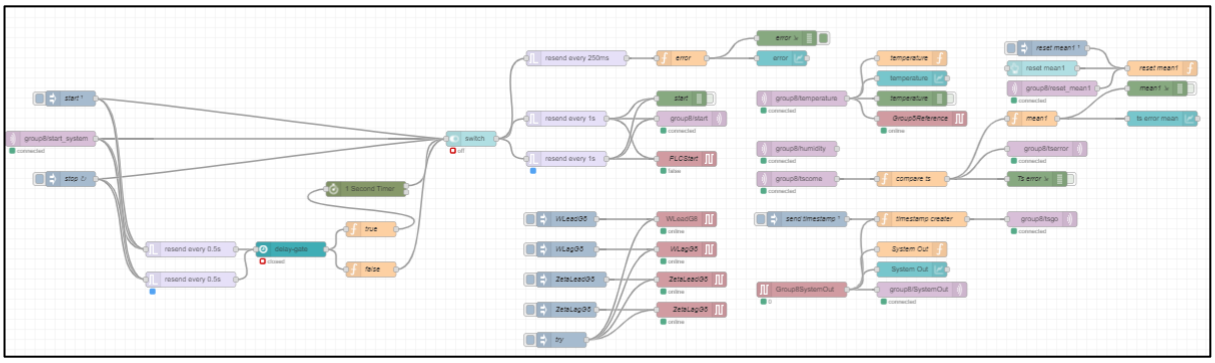


Figure 2- Whole chart of node-red for group 8

As can be seen in figure 3, the system can be started from node-red through the injection button manually, the User Interface, or automatically every 15 minutes in 1 hour with the"delay-gate" node. The start button activates the system if the switch node was turned on from the node-red dashboard in figure 8 and the group8/start_system sends a boolean signal periodically with the trigger node.

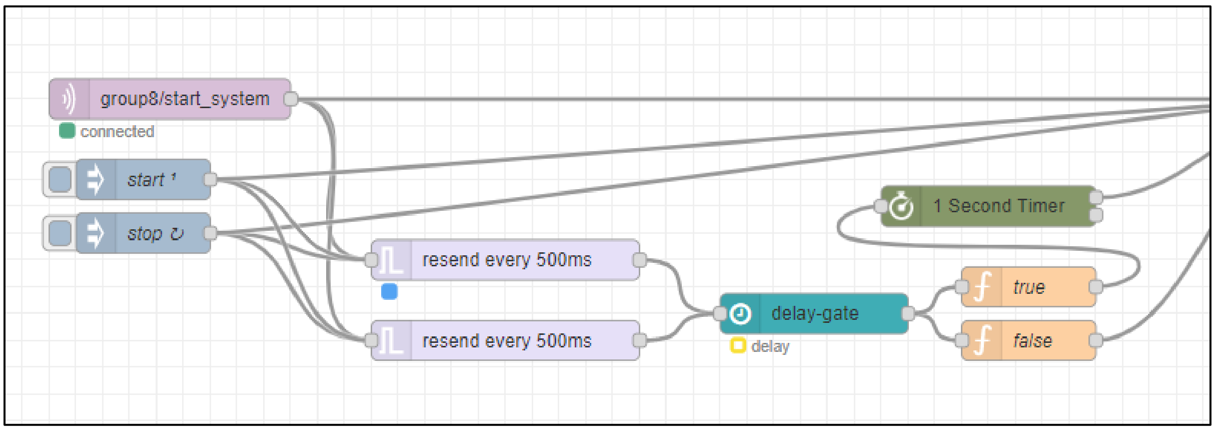


Figure 3- Starting the system.

With the switch turns on, the start or stop command is sent to the system periodically to prevent interruptions that stop the system to operate. Also, the error is calculated every 250ms and registered to the chart at the node-red dashboard.

Content of error function:

```
group8Temp = flow.get("group8/temperature");
group8Sysout = flow.get("group8/sysout");
msg.payload = group8Temp - group8Sysout
return msg;
```

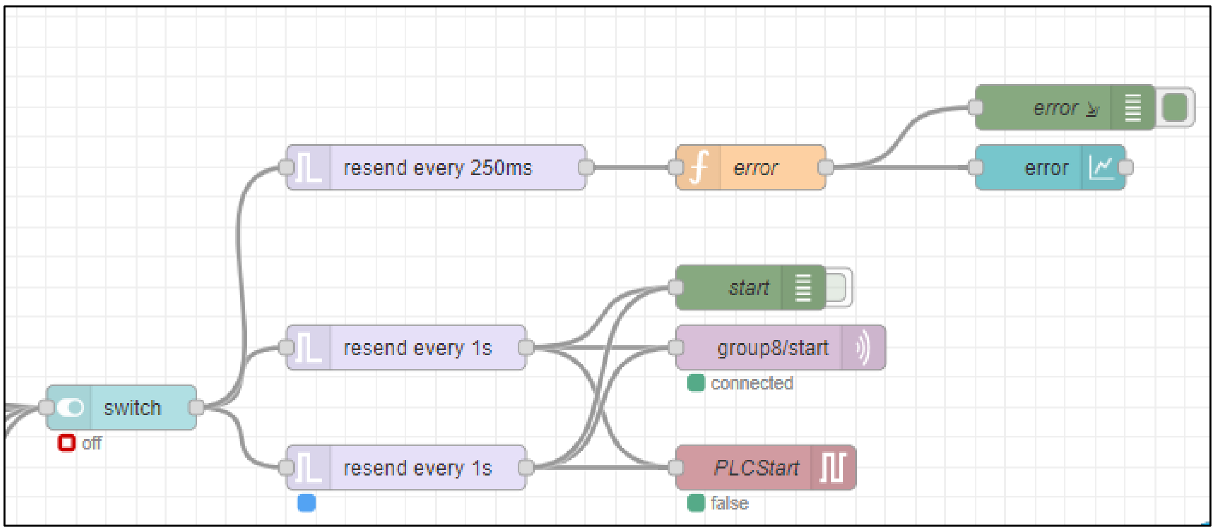


Figure 4- Starting PLC system and error calculation.

Sending the temperature data from esp32 with MQTT to the PLC for the reference signal of the formed system in the PLC, an ui_chart node was added to the scheme to observe the temperature changes through the designed dashboard.

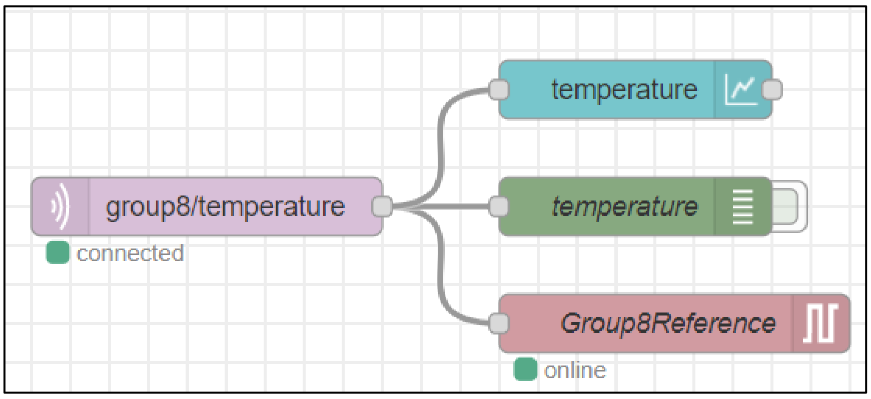


Figure 5- Sending the temperature data to the PLC.

With the part that is visualized in figure 6, the system parameters that attended to group 8 are sent to the PLC.

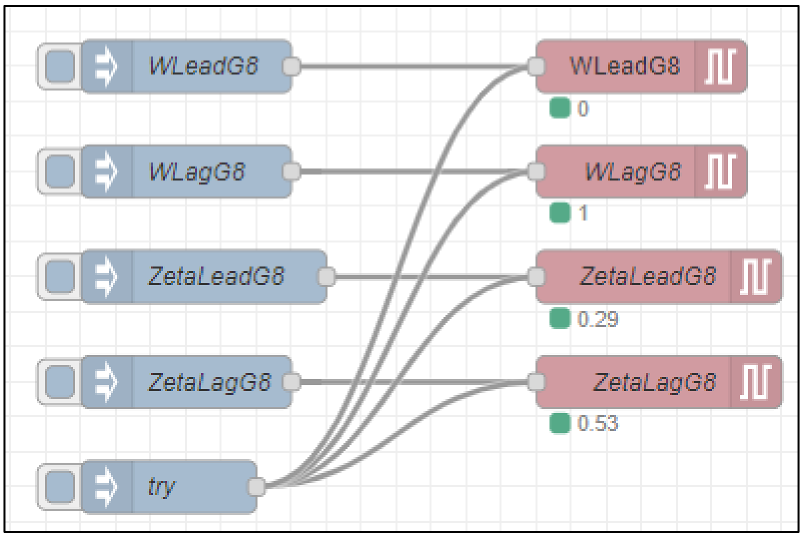


Figure 6- Pushing system parameters to the PLC.

Here with this part in figure 7, the output of the system in PLC is captured with the eth_ip_in node and starts a time stamp when the output comes in for the communication delay. The function in the “timestamp creator” is as follows:

```
msg.payload = new Date();
msg.payload = msg.payload.getTime();
return msg;
```

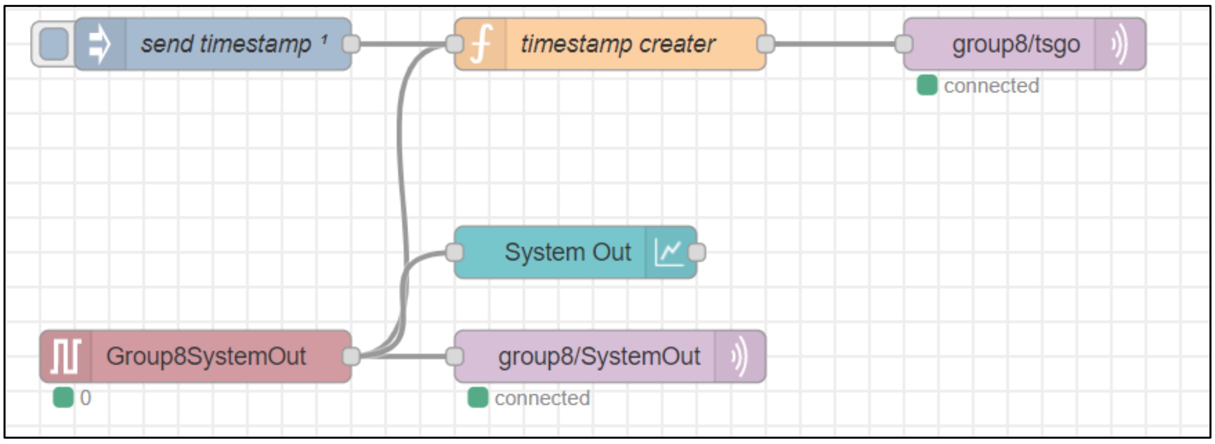


Figure 7- Pushing system parameters to the PLC.

Communication delay is calculated as follows:

“timestamp creater” (figure 7)function creates timestamp of current time and send it to esp32. After esp32 take the timestamp it send it back to node-red server via “group8/tscome”. When message comes from “group8/tscome” it directly sent to “compare ts” (figure 8)function. “compare ts” function calculate error between current timestamp and timestamp of coming message. This is shows to communication delay. After that, the mean is calculated with last hundred points of communication delay in” mean1” function and sent to the created node-red dashboard (figure-8). “reset mean1” node is controlled on the same page and it adjusts the parameters used in the “mean1” function.

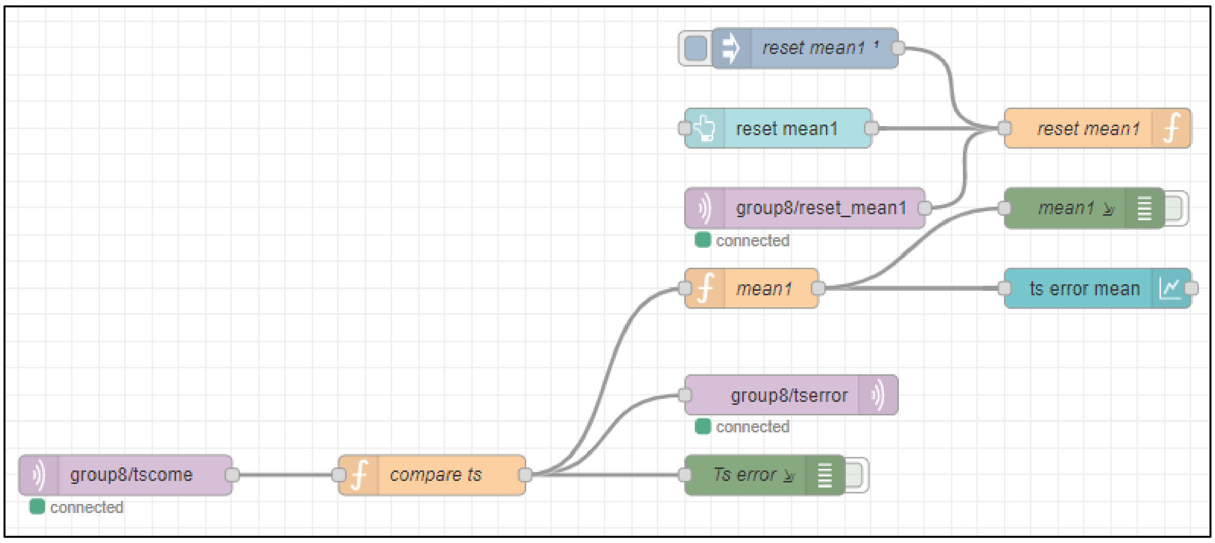


Figure 8

Here are the contents of the functions in figure 8:

compare ts:

```
now = new Date();
msg.payload = (now.getTime() - msg.payload);
return msg;
```

mean1:

```
group8Count1 = flow.get("group8/count1");
group8Mean1 = flow.get("group8/mean1");
group8Recalculate1 = flow.get("group8/recalculate1");
group8Queue1 = flow.get("group8/queue1");
var len = 100
if(group8Recalculate1)
{
    group8Queue1 = [];
    group8Count1 = 0;
    group8Mean1 = 0;
    group8Recalculate1 = 0;
}
group8Queue1.push(msg.payload);
```

```
if(group8Count1 < len)
{
    group8Mean1 = (group8Mean1*group8Count1+msg.payload)/(group8Count1+1);
    group8Count1 = group8Count1 + 1;
    msg.payload = group8Mean1;
}
else
{
    var removed_element = group8Queue1.shift();
    group8Mean1 = (group8Mean1*len + msg.payload - removed_element)/len
    msg.payload = group8Mean1;
}
flow.set("group8/count1",group8Count1);
flow.set("group8/mean1",group8Mean1);
flow.set("group8/recalculate1",group8Recalculate1);
flow.set("group8/queue1",group8Queue1);
return msg;
```

reset mean1:

```
flow.set("group8/count1",0);
flow.set("group8/mean1",0);
flow.set("group8/recalculate1",0);
flow.set("group8/queue1",[]);
return msg;
```

Node-red Dashboard

In this section, the mean of the timestamp, input temperature and system output are visualized as described in the earlier parts of the node-red section. When the “reset mean” button is pressed, the “reset_mean1” UI button node in figure 7 gets activated. Also, switch turns on the switch node in figure 3.

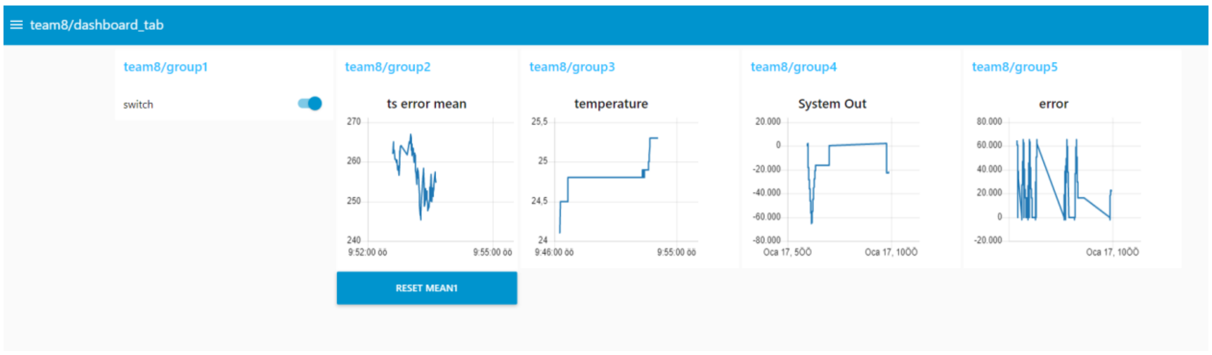


Figure 9- Node-red dashboard with graphs and a button.

User Interface

The user interface has been created to command the node-red server group 8 flow. The abilities of the user interface can be listed as:

- Connection and disconnection with node-red server and read the values specified in node-red.
- Start/stop the system with the start button.
- Send message to the topic with the publish button

- This code written in python

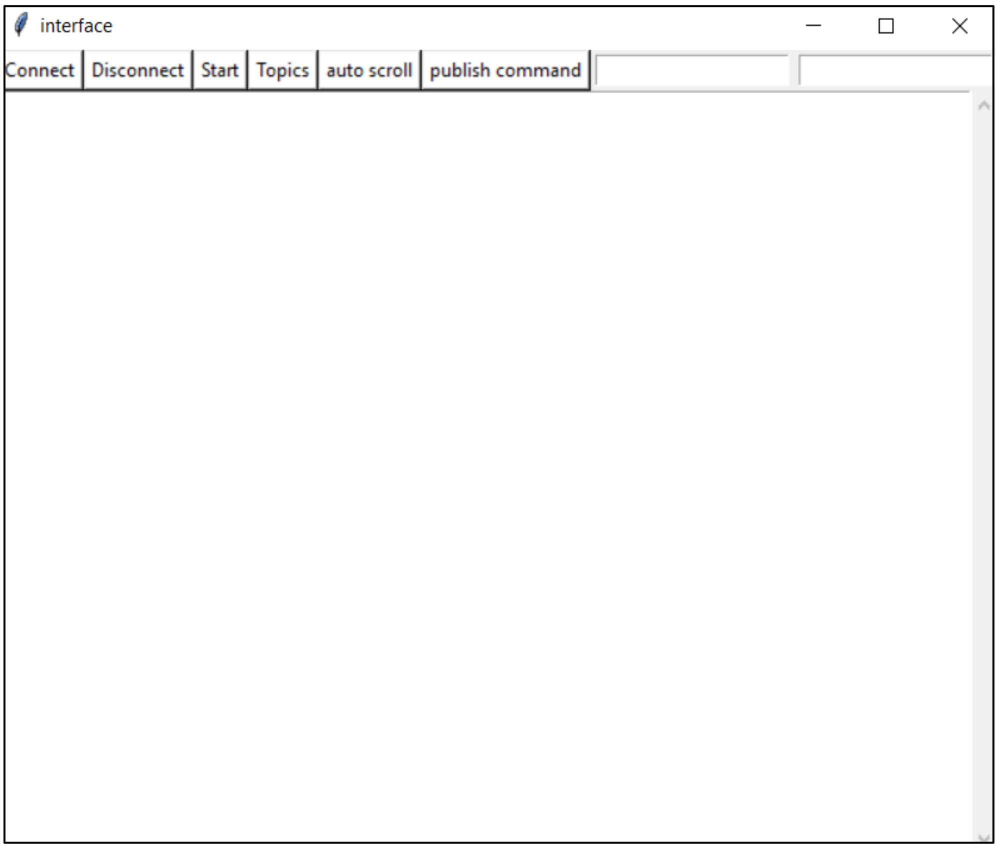


Figure 10. The user interface created to interfere with the system.

In Topics screen (figure 11), topics that want to listen can add or delete.

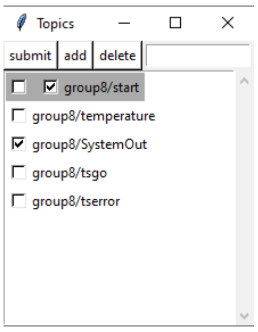
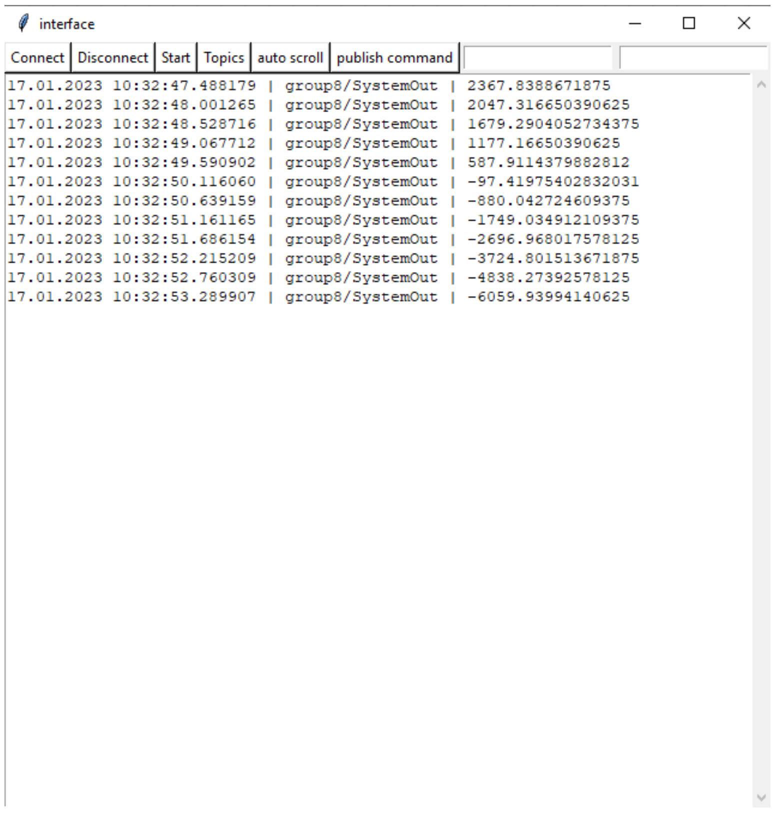


Figure 11. The user interface topic screen.

Topics can be logged with this interface.(figure 12)



```
17.01.2023 10:32:47.488179 | group8/SystemOut | 2367.8388671875
17.01.2023 10:32:48.001265 | group8/SystemOut | 2047.316650390625
17.01.2023 10:32:48.528716 | group8/SystemOut | 1679.2904052734375
17.01.2023 10:32:49.067712 | group8/SystemOut | 1177.16650390625
17.01.2023 10:32:49.590902 | group8/SystemOut | 587.9114379882812
17.01.2023 10:32:50.116060 | group8/SystemOut | -97.41975402832031
17.01.2023 10:32:50.639159 | group8/SystemOut | -880.042724609375
17.01.2023 10:32:51.161165 | group8/SystemOut | -1749.034912109375
17.01.2023 10:32:51.686154 | group8/SystemOut | -2696.968017578125
17.01.2023 10:32:52.215209 | group8/SystemOut | -3724.801513671875
17.01.2023 10:32:52.760309 | group8/SystemOut | -4838.27392578125
17.01.2023 10:32:53.289907 | group8/SystemOut | -6059.93994140625
```

Figure 12. The user interface when logging.