Otto-von-Guericke-University Magdeburg
Faculty of Electrical Engineering and Information Technology
Institute for Automation Engineering

Automation Lab

Monitoring the Process using OPC UA

submitted: February 6, 2022

by: Md Yasin Arafat

Contents

1 The OPC UA Information Model Notation

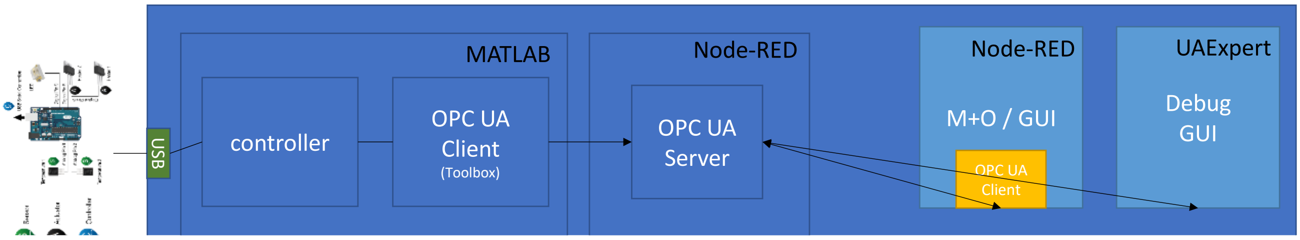


Figure 1.1: Architecture of the practicum task [1]

In Fig. 1.2 is shown the diagram of the designed OPC UA node set in the OPC UA notation.

Table 1.1 content is mapped into OPC UA notation as variable attributes.

Name	Data Type	Range	Unit	Initial Value
Input: Temperature_1	Double	30 − 60	°C	40
ControllerError_1	Double	(−60) − 60	°C	na
ManipulatedVariable_1	Double	0 − 80	%	na
Output: actualTemperature_1	Double	30 − 60	°C	na
LED	Boolean	true, false	na	false
Integral: Ki	Double	na	na	0.0088
Proportional: Kp	Double	na	na	1.6
Temperature_1_High_Limit	Double	na	na	60
ManufacturerName	String	na	na	Byu Prism
ManufacturerProductDesignation	String	na	na	Arduino Temperature Control Lab
ManufacturerProductFamily	String	na	na	Educational Equipment
SerialNumber	String	na	na	F7DH4HJ
YearOfConstruction	String	na	na	2020
Address	String	na	na	Brigham Young University, Provo, UT 84602

Table 1.1: Parameters of the device which are defined in the OPC UA server [1]

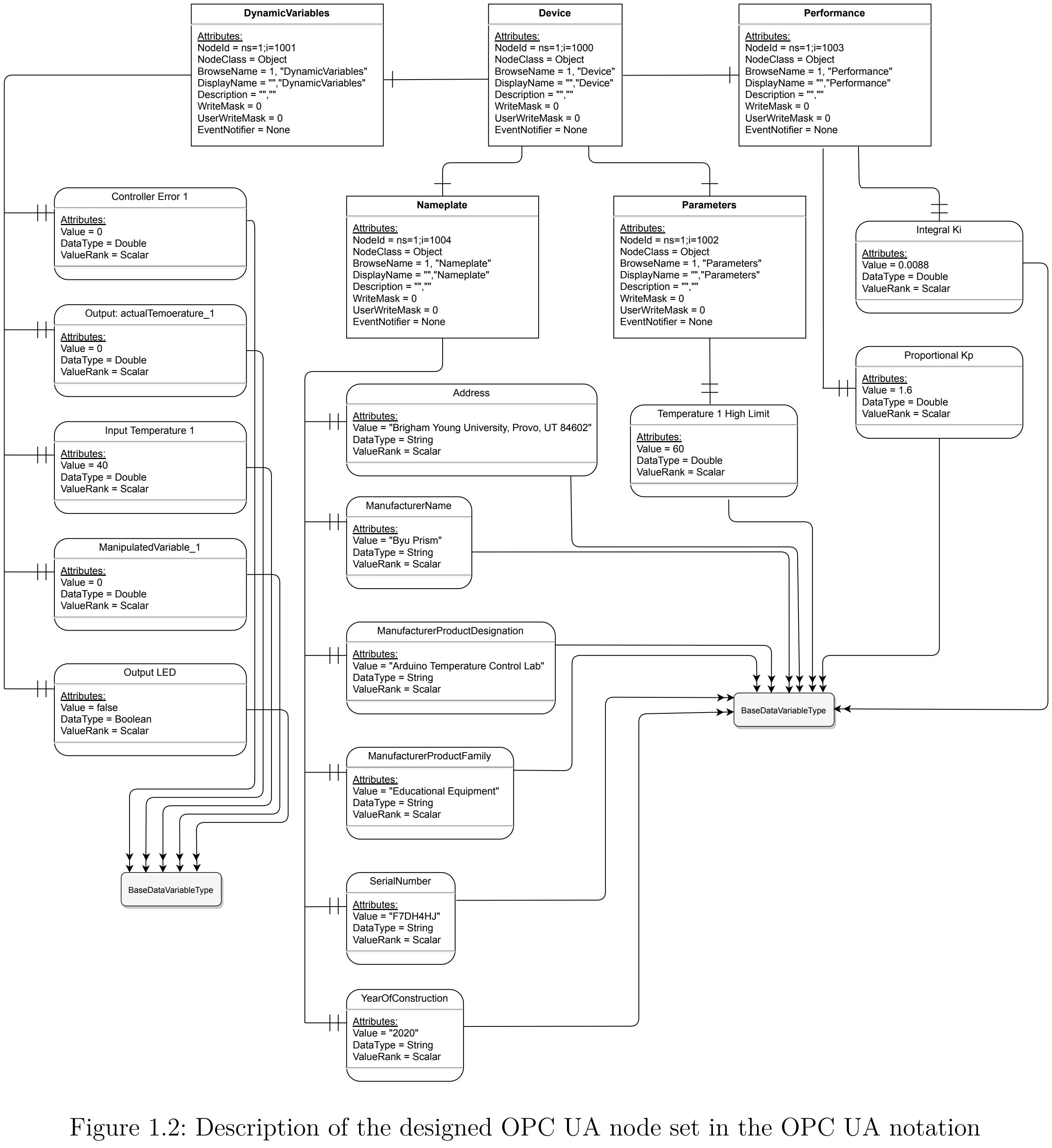


Figure 1.2: Description of the designed OPC UA node set in the OPC UA notation

2 Configuration of the OPC UA server

The java script code of the OPC UA server configuration is given in Appendix A and is based on the handout which was provided [1]. The variables that are received from the client to the server-side are: temperature of sensor 1, controller output, proportional controller gain, integral controller coefficient, error signal. These variables are updated on each iteration of the loop. The variables which are sent from the server-side to the client-side are command signal, controller gain, and integral controller coefficient.

Detailed description of the variables functionality:

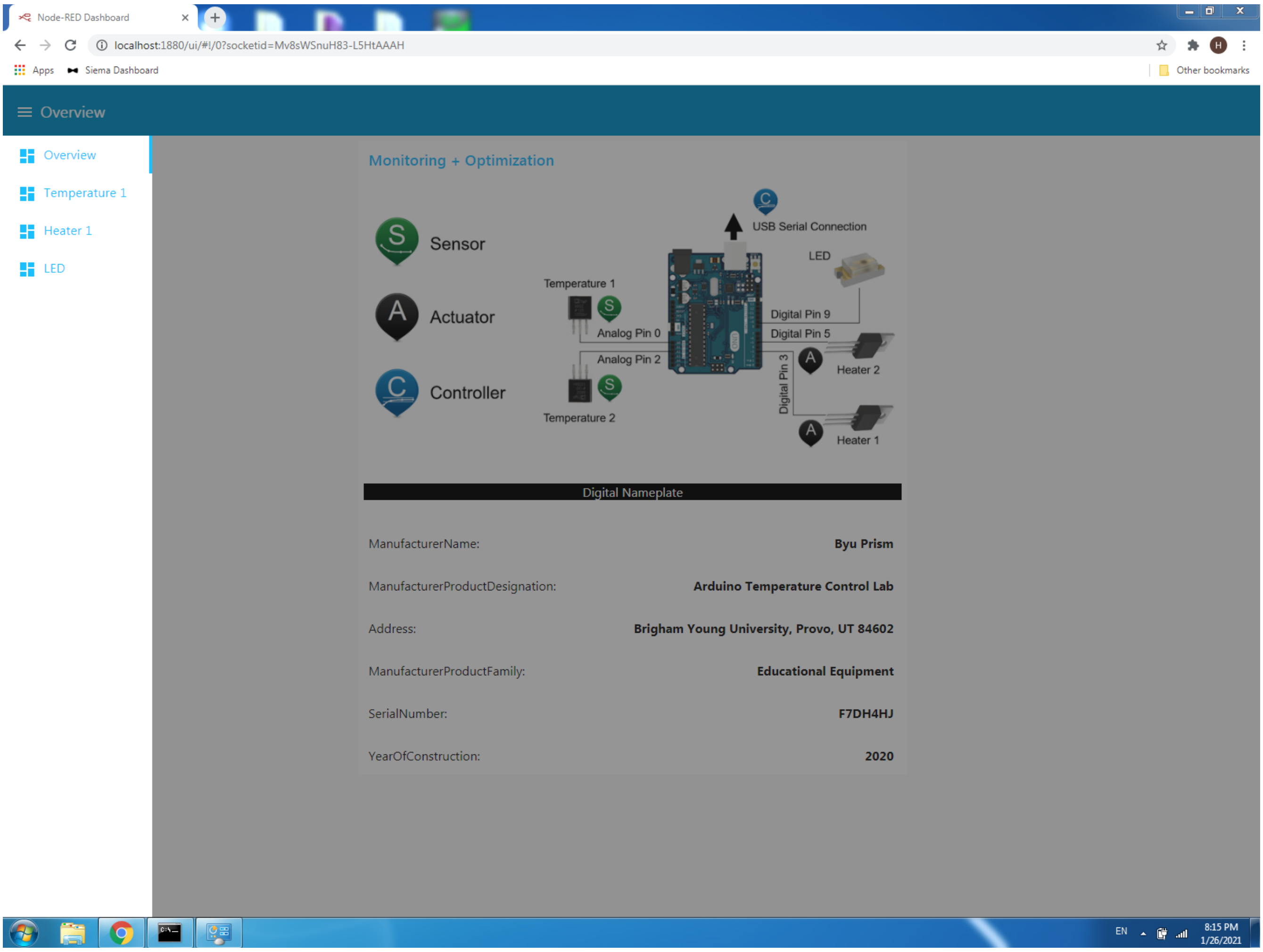
- temp1 with "browseName": "Output: actualTemperature_1" is used to store the temperature value of sensor 1 and it is updated on every iteration of the control loop.
- tempinput with "browseName": "Input: Temperature_1" is used to store the value of the command signal 1 and is fed to the control loop on each iteration.
- heater1 with "browseName": "ManipulatedVariable_1" is used to store the value of the controller output and it is updated on every iteration of the control loop.
- controller_error1 with "browseName": "Controller Error 1" is used to store the value of the error signal and it is updated on every iteration of the control loop.
- led with "browseName": "Output: LED" is used to store the current value of the LED.
- temp1_limit with "browseName": "Temperature_1_High_Limit" is used to store the value of the temperature 1 upper boundary, and it is displayed in the graphical user interface.
- kp with "browseName": "Proportional: Kp" is used to store the value of the proportional controller gain and is fed to the control loop on each iteration.
- ki with "browseName": "Integral: Ki" is used to store the value of the integral controller gain and is fed to the control loop on each iteration.
- manufacturer_name, manufacturer_product_designation, address, serial_number, manufacturer_product_family, year_of_construction have string datatype and are displayed in the graphical user interface.

3 Screenshots

3.1 The GUI root

In Fig. 3.1 is shown the root of the graphical user interface. The user interface contains 3 pages:

- Temperature 1, which displays the plot of the output of sensor 1 and the command signal in one graph.
- Heater 1, which displays the output of the PI controller.
- LED, which displays the output of the LED.



Digital Nameplate

ManufacturerName: Byu Prism

ManufacturerProductDesignation: Arduino Temperature Control Lab

Address: Brigham Young University, Provo, UT 84602

ManufacturerProductFamily: Educational Equipment

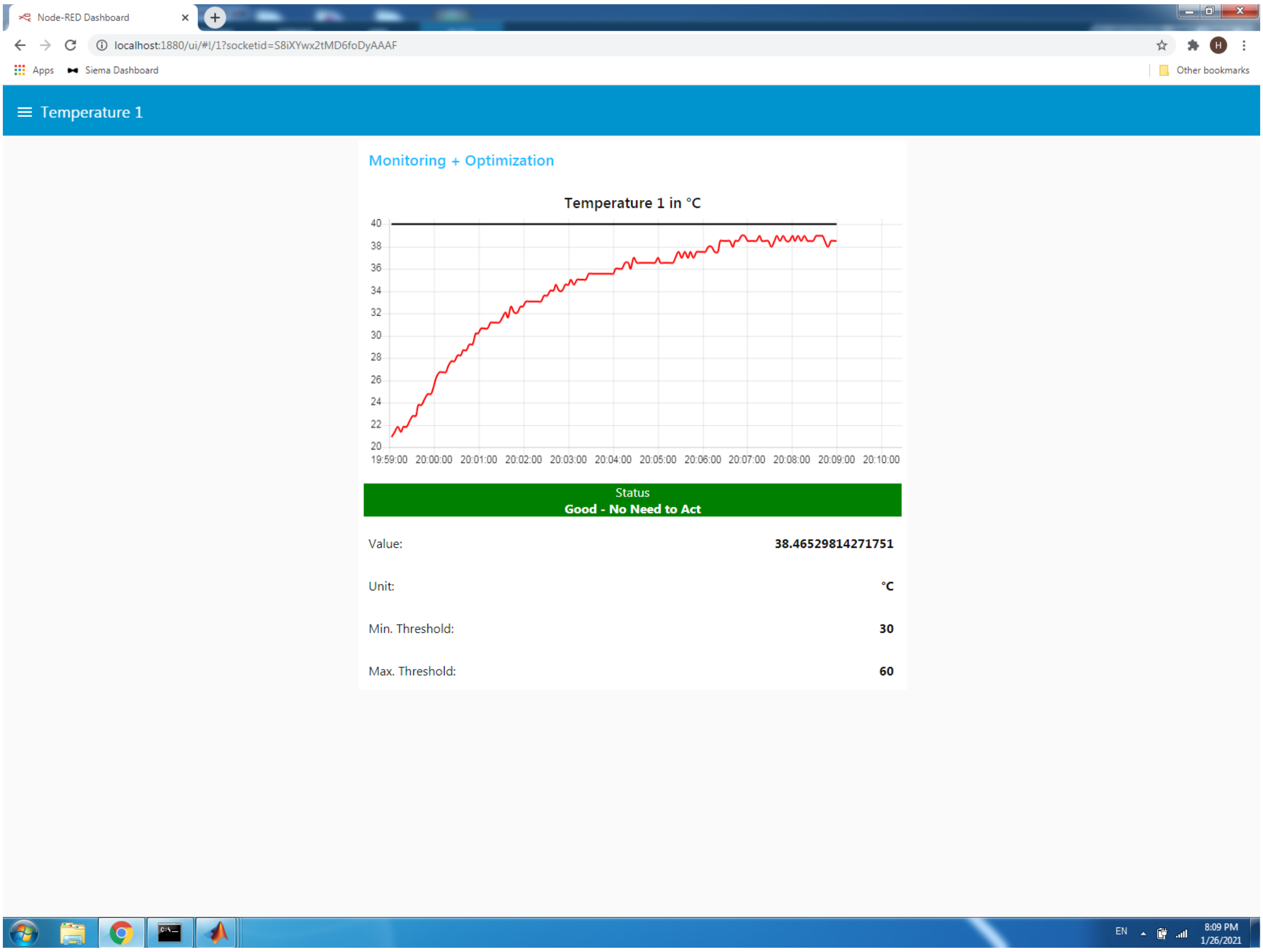
SerialNumber: F7DH4HJ

YearOfConstruction: 2020

Figure 3.1: Root of the GUI

3.2 The signal curve

Fig. 3.2 displays the signal curve of the sensor 1 output as well as the command signal in the same graph.



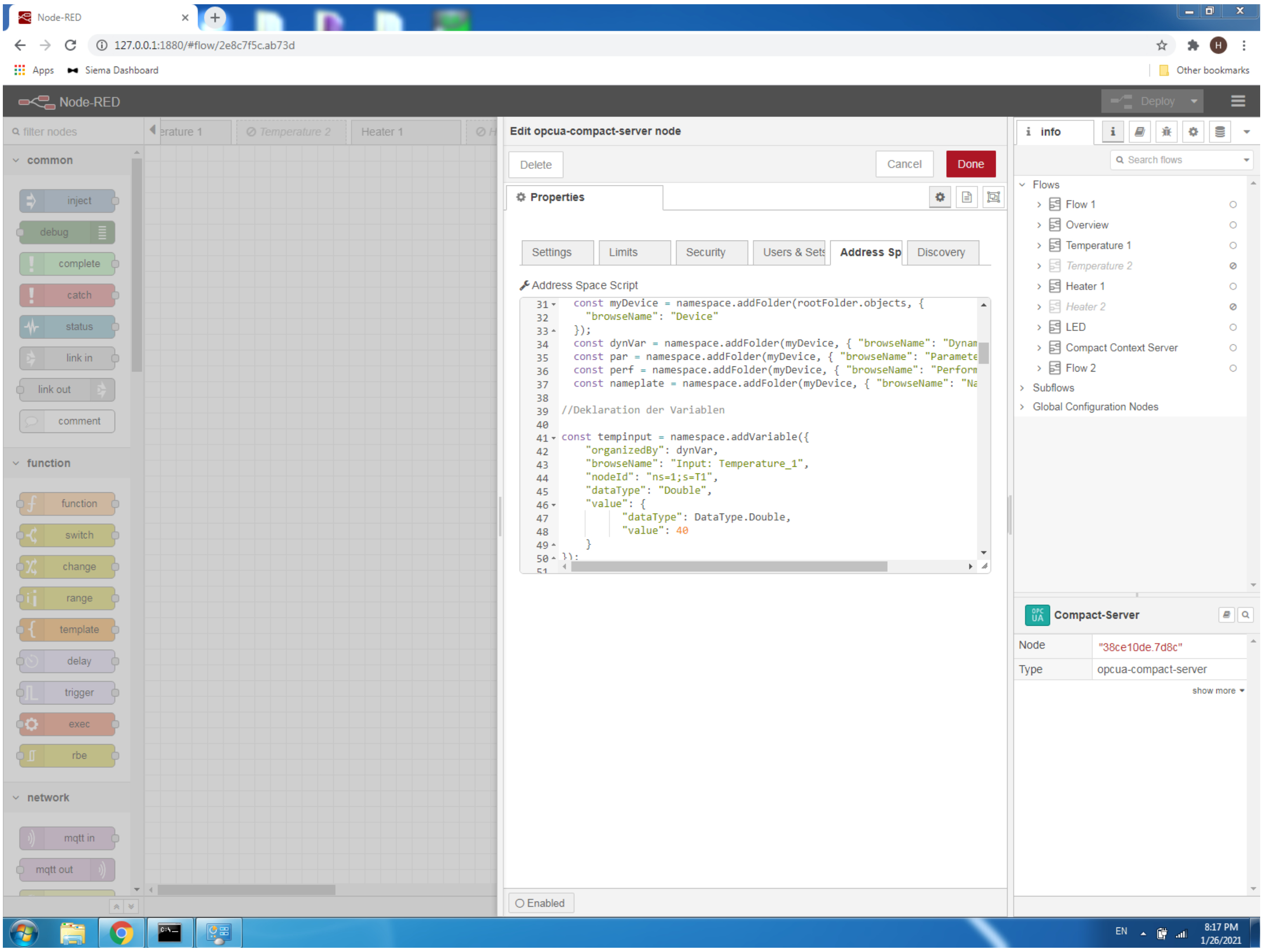
Status
Good - No Need to Act

Value:	38.46529814271751
Unit:	°C
Min. Threshold:	30
Max. Threshold:	60

Figure 3.2: Signal curve of temperature 1

3.3 Variable description

Fig. 3.3 displays the “Compact Context Server” address space field. Formatting the address space is performed through java script code given in Appendix A.



```
  const myDevice = namespace.addFolder(rootFolder.objects, {
    "browseName": "Device"
  });
  const dynVar = namespace.addFolder(myDevice, { "browseName": "Dynam
  const par = namespace.addFolder(myDevice, { "browseName": "Paramete
  const perf = namespace.addFolder(myDevice, { "browseName": "Perform
  const nameplate = namespace.addFolder(myDevice, { "browseName": "Na

//Deklaration der Variablen

const tempinput = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Input: Temperature_1",
    "nodeId": "ns=1;s=T1",
    "dataType": "Double",
    "value": {
        "dataType": DataType.Double,
        "value": 40
    }
});
```

Node	"38ce10de.7d8c"
Type	opcua-compact-server

Figure 3.3: Variable description in the “Compact Context Server”

4 Changing thresholds

Initially, the threshold was set 30 - 60 $^{\circ}C$ and was later updated to 30 - 50 $^{\circ}C$. This was done by changing the condition which prints a message in the GUI from "(msg.payload>=30 msg.payload<=60)" to "(msg.payload>=30 msg.payload<=50)". In the presented setup an optimization algorithm was not used, so by changing the thresholds the operator in that case would only get an error message through the graphical user interface which indicates that the value of the temperature is not within the specified boundaries. An optimization algorithm can be developed in order to automate the process and to define new controller parameters automatically in order to keep the output value between the specified boundaries or to perform other control actions. However, as this was not part of the task it was not implemented in our project.

References

[1] Handout-Temperature Control Lab 4.

A Appendix

```
function constructAlarmAddressSpace(server, addressSpace, eventObjects, done) {


  const opcua = coreServer.choreCompact.opcua;
  const LocalizedText = opcua.LocalizedText;
  const namespace = addressSpace.getOwnNamespace();

  const Variant = opcua.Variant;
  const DataType = opcua.DataType;
  const DataValue = opcua.DataValue;

  var flexServerInternals = this;

  coreServer.debugLog("init dynamic address space");
  const rootFolder = addressSpace.findNode("RootFolder");

  node.warn("construct new address space for OPC UA");


  const myDevice = namespace.addFolder(rootFolder.objects, {
    "browseName": "Device"
  });
  const dynVar = namespace.addFolder(myDevice, { "browseName": "DynamicVariables" });
  const par = namespace.addFolder(myDevice, { "browseName": "Parameters" });
  const perf = namespace.addFolder(myDevice, { "browseName": "Performance" });
  const nameplate = namespace.addFolder(myDevice, { "browseName": "Nameplate" });



const tempinput = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Input: Temperature_1",
    "nodeId": "ns=1;s=T1",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 40
    }
});

const temp1 = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Output: actualTemperature_1",
    "nodeId": "ns=1;s=T3",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": undefined
    }
});







const heater1 = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "ManipulatedVariable_1",
    "nodeId": "ns=1;s=H1",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 0
    }
});








const controller_error1 = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Controller Error 1",
    "nodeId": "ns=1;s=CE1",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 0
    }
```

```
});



const led = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Output: LED",
    "nodeId": "ns=1;s=led",
    "dataType": "Boolean",
    "value": {
          "dataType": DataType.Boolean,
          "value": false
    }
});

const temp1_limit = namespace.addVariable({
    "organizedBy": par,
    "browseName": "Temperature_1_High_Limit",
    "nodeId": "ns=1;s=T1_limit",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 60
    }
});



const kp = namespace.addVariable({
    "organizedBy": perf,
    "browseName": "Proportional: Kp",
    "nodeId": "ns=1;s=kp",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 1.6
    }
});

const ki = namespace.addVariable({
    "organizedBy": perf,
    "browseName": "Integral: Ki",
    "nodeId": "ns=1;s=ki",
    "dataType": "Double",
    "value": {
          "dataType": DataType.Double,
          "value": 0.0088
    }
});

const manufacturer_name = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "ManufacturerName",
    "nodeId": "ns=1;s=manufacturer_name",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "Byu Prism"
          }
});

const manufacturer_product_designation = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "ManufacturerProductDesignation",
    "nodeId": "ns=1;s=manufacturer_product_designation",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "Arduino Temperature Control Lab"
          }
});

const address = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "Address",
    "nodeId": "ns=1;s=address",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "Brigham Young University, Provo, UT 84602"
    }
});

const manufacturer_product_family = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "ManufacturerProductFamily",
    "nodeId": "ns=1;s=manufacturer_product_family",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "Educational Equipment"
    }
});

```

```
const serial_number = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "SerialNumber",
    "nodeId": "ns=1;s=serial_number",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "F7DH4HJ"
    }
});

const year_of_construction = namespace.addVariable({
    "organizedBy": nameplate,
    "browseName": "YearOfConstruction",
    "nodeId": "ns=1;s=year_of_construction",
    "dataType": "String",
    "value": {
          "dataType": DataType.String,
          "value": "2020"
    }
});


  coreServer.debugLog("create dynamic address space done");
  node.warn("construction of new address space for OPC UA done");

  done();
}
```