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

# Automation Lab


## Temperature Control Lab 3 (TCL3)

## Report

Submitted: January 19, 2024

By: Akshay Asokan
Matr No: 243897

Supritha Rajashekaraiah
Matr No: 244941

# Contents

# 1 OPC UA notation and Information model

The OPC UA information model 1.0.2 has been designed according to the described heating system based on figure 1.0.1 and Table 1.0.1.

- The implementation of the Matlab with TCL kit, Red-node (both server and client), and UAExpert is done on single computer.
- The device object establishes a connection with the Folder Type through the "Has Type Definition" association.
- The Nameplate, Parameters, Performance, and Dynamic Variables, categorized as types of folders, are interconnected using the same "Has Type Definition" relationship.
- As the Device encompasses Nameplate, Parameters, Performance, and Dynamic Variables, each of these objects is linked to the Device through the "Has Component" association.
- Each set of variables, identified from the table, is then categorized into the corresponding object set using the "Has Property" connection, inheriting the properties of the respective object.
- Simultaneously, each variable serves as the type definition of the Base Data Variable Type, establishing a connection through the "Has Type Definition" association.
- Lastly, the Folder Type is a subclass of the BaseObject Type, and the Base Data Variable Type is a subclass of the Base Variable Type.

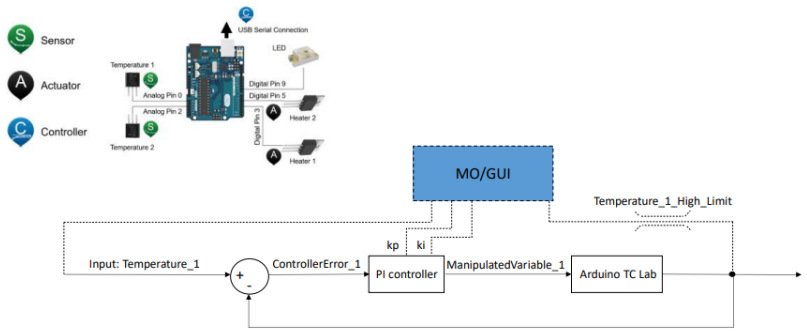


Figure 1.0.1: Block diagram of the control loop and used variables

Table 1.1: parameters of the device which have to be defined in the OPC UA server

| **Browse Name** | **Data Type** | **Range** | **Unit** | **(Initial) Value** | **NodeID** |
|---|---|---|---|---|---|
| Input: Temperature_1 | Double | 30-60 | °C | 40 | ns=1;s=T1 |
| ControllerError_1 | Double | (-60)-60 | °C | NA | ns=1;s=CE1 |
| ManipulatedVariable_1 | Double | 0-80 | % | NA | ns=1;s=MV1 |
| Output: actualTemperature_1 | Double | 30-60 | °C | 0 | ns=1;s=H1 |
| LED | Boolean | true, false | NA | false | false |
| Integral: Ki | Double | NA | NA | 0.0088 | ns=1; s=ki |
| Proportional: Kp | Double | NA | NA | 1.6 | ns=1; s=kp |
| Temperature_1_ High_ Limit | Double | NA | NA | 60 | ns=1; s=T1_limit |
| ManufacturerName | String | NA | NA | Byu Prism | ns=1; s=manufacturer_ name |
| ManufacturerProduct Designation | String | NA | NA | Arduino Temperature Control Lab | ns=1;s= manufacturer_ product_designation |
| ManufacturerProduct Family | String | NA | NA | Educational Equipment | ns=1;s= manufacturer_ product_family |
| SerialNumber | String | NA | NA | F7DH4HJ | ns=1;s= serial_number |
| YearOfConstruction | String | NA | NA | 2020 | ns=1; s=year_ of_construct ion |
| Address | String | NA | NA | Brigham Young University, Provo, UT 84602 | ns=1; s=address |

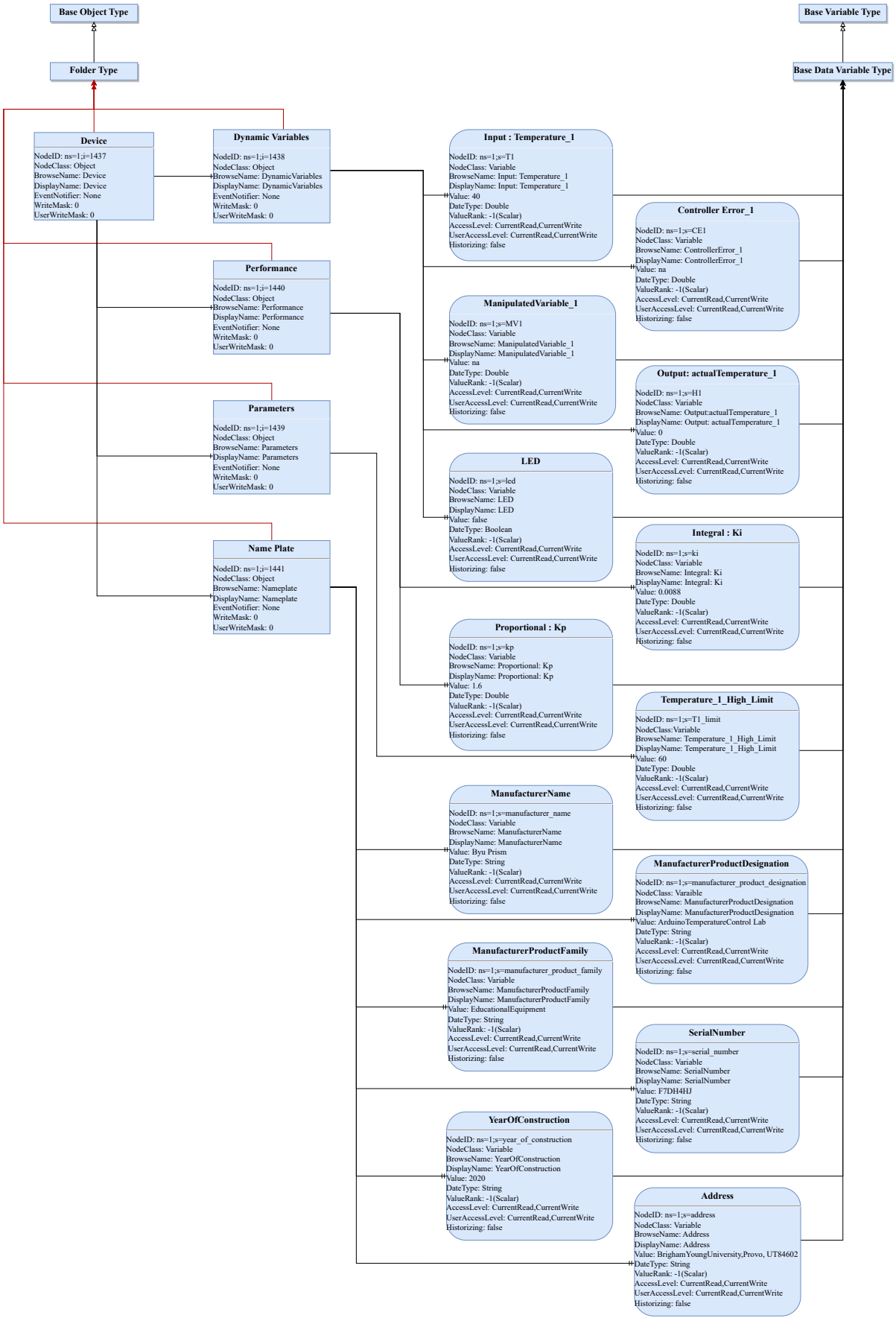


Figure 1.0.2: Information model of Temperature control system

# 2 Explanation of Java Script.

**Explain the configuration of the OPC UA server in terms of the java script code. Explain the meaning of the used attributes.**

–> **Initialization of Global Variables:** The script initializes essential components, including the OPC UA library and captures the server's namespace. Internal sandbox objects for Node-RED contexts and standard functions are established. as shown in fig 2.0.1.

```
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;
```

Figure 2.0.1: Initialization of Global Variables

–> **Structure Declaration:** Post-initialization, a structured folder system (e.g., dynamic variables, parameters) is defined in the OPC UA server's address space, enhancing organization as shown in fig 2.0.2

```
var flexServerInternals = this;

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

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

//Deklaration der Struktur
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" });
```

Figure 2.0.2: Structure Declaration

–> **Variable Declaration:** Diverse OPC UA variables (e.g., temperature input, controller error) are declared under folders like dynVar, conveying various system properties as shown in fig: 2.0.3.

```
const temp1 = namespace.addVariable({
    "organizedBy": dynVar,
    "browseName": "Input: Temperature_1",
    "nodeId": "ns=1;s=T1",
    "dataType": "Double",
    "value": {
        "dataType": DataType.Double,
        "value": 40
    }
});
```

Figure 2.0.3: Java code of a variable node

- **namespace.addVariable:** Function adding a new variable node to the OPC UA server's address space, representing "Temperature_1."

- **"organizedBy"- dynVar:** Specifies that the new variable node is organised under the dynamic variable (dynVar) node, indicating dynamic variables in the OPC UA server.
- **"browseName" - "Input:Temperature_1":** Sets the human-readable browse name for the variable node as "Input: Temperature_1."
- **"nodeId"- "ns=1;s=T1":** Establishes the NodeId for the variable node, with the namespace index (ns=1) and identifier (s=T1), ensuring a unique identifier within the OPC UA server.
- **"dataType": "Double":** indicates the data type of the variable as Double, representing storage for a double-precision floating-point value.
- **"value":** Object containing details about the initial value of the variable.
- **"dataType": DataType.Double:** Specifies the data type of the initial value, confirming it as a double.
- **"value": 40:** Represents the initial value of the variable, set to 40.

–> **UAExpert Server:** Following the declaration of variables in the OPC UA Server's address space, the configuration is deployed and subsequently tested using UA Expert. Upon establishing a successful server connection, users can access and view the declared variables along with their respective folders in the address space through UA Expert. To inspect the status of each variable, users can utilize the Data Access view, facilitating a seamless process of dragging and dropping variables for detailed examination as shown in fig 2.0.4

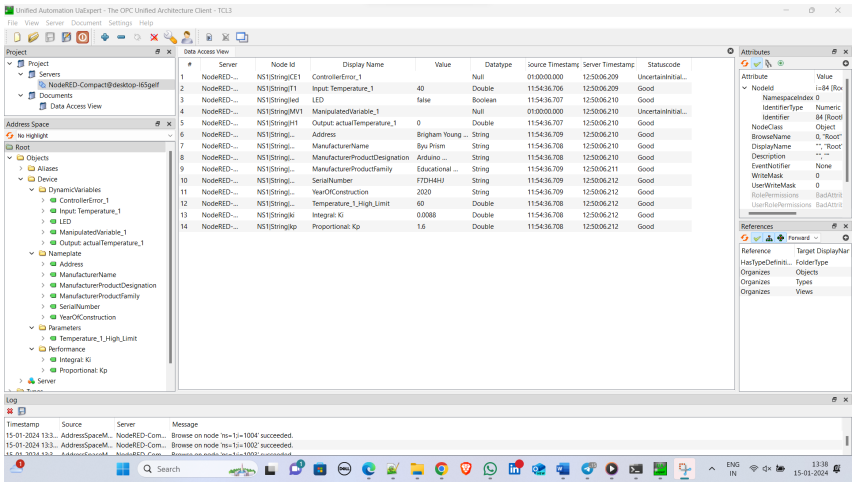

Figure 2.0.4: Overview of UAExpert Server

# 3 Practical Part - Results

## 3.1 Variable description in the “Compact Context Server”

The below picture shows 3.1.1 Java Script of Controller error, manipulated variable and actual temperature in address space of OPC UA Server node.

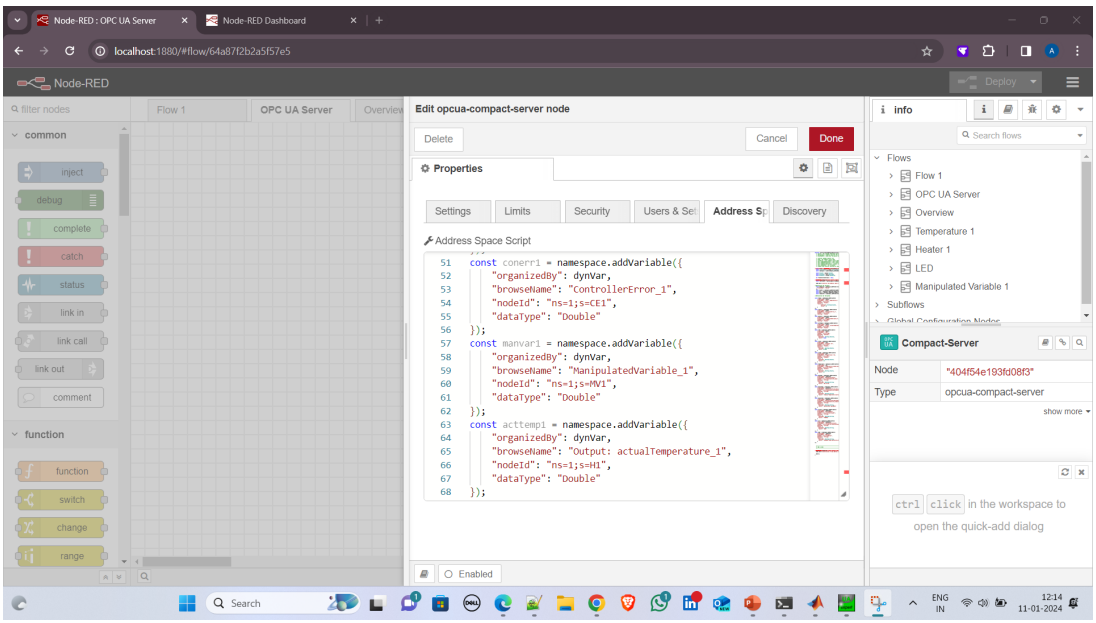


Figure 3.1.1: Variable Description

## 3.2 The GUI root.

The depicted figure 3.2.1 illustrates the GUI root, serving as the interface for the Monitoring and Optimization Application client. This dashboard prominently displays the variables outlined in the server. In addition to the mentioned variables, a new dashboard group has been incorporated specifically for controller errors, complementing the information already detailed in the server. Furthermore, the dashboard features an image of the TCL, integrated using a URL.

## 3.3 The signal curve of Input: Temperature_1 and Output: actualTemperature_1.

The initial temperature value assigned to Temperature 1 was 40°C, as depicted in the figure 3.3.1. The graphical representation of the actual temperature for the set point of 40°C is illustrated as shown in the fig 3.3.2

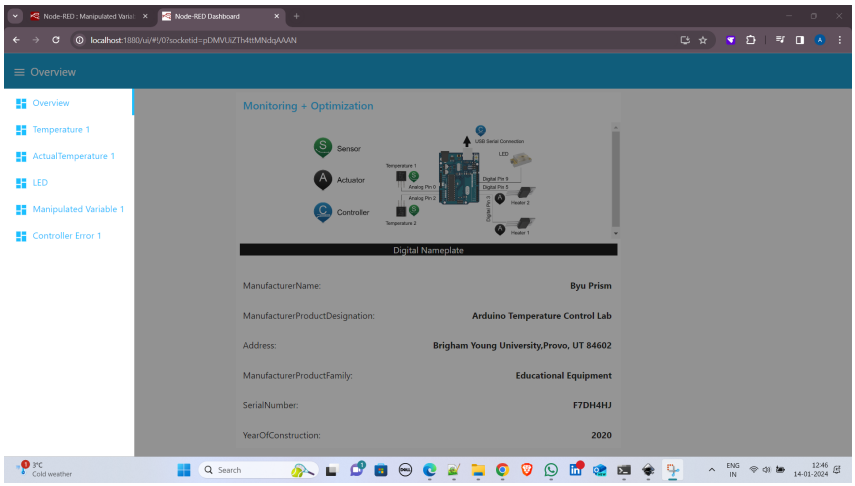


Figure 3.2.1: GUI root

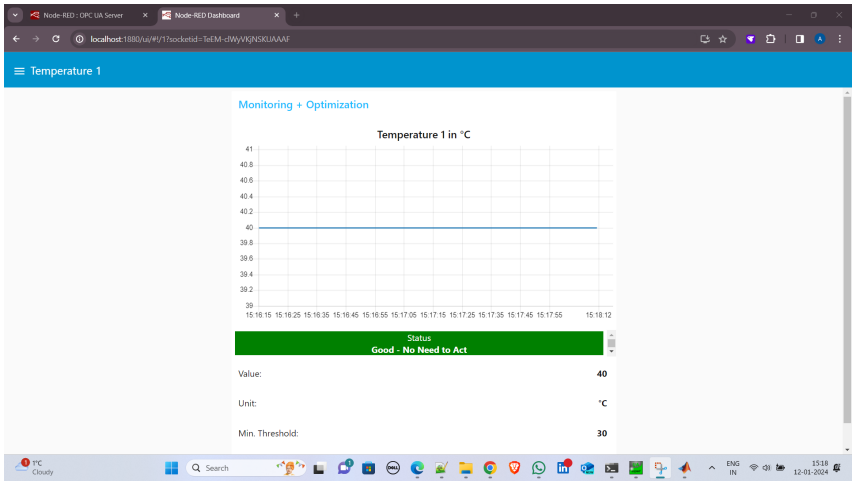


Figure 3.3.1: Input: Temperature_1

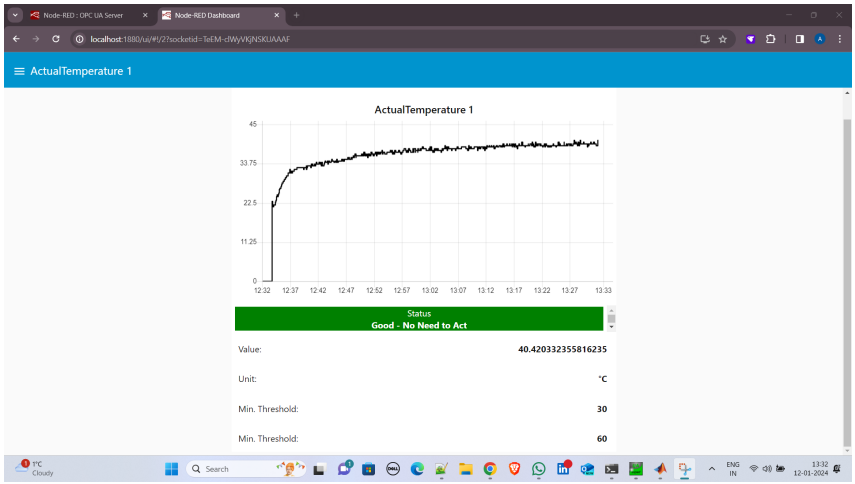


Figure 3.3.2: Output: actualTemperature_1

## 3.4 Change of Threshold

The changed threshold values are a minimum of 30°C and a maximum of 40°C instead of the specified minimum of 30°C and a maximum of 60°C. Input initial temperature also changed to 50°C to observe both ends of the thresholds. If the temperature falls below or exceeds these thresholds as shown in the figure 3.4.1 and 3.4.3, the GUI will display the status as "Error-Check Signal Chain." However, if the temperature is within this range, the status will be shown as "Good - No need to act." 3.4.2

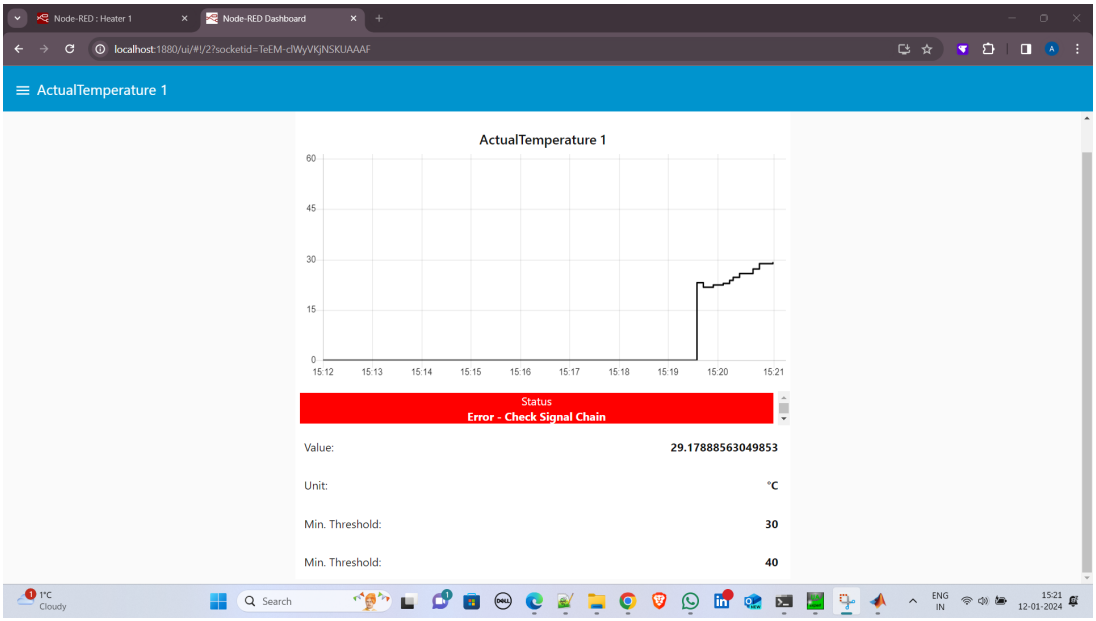


Figure 3.4.1: Below Threshold

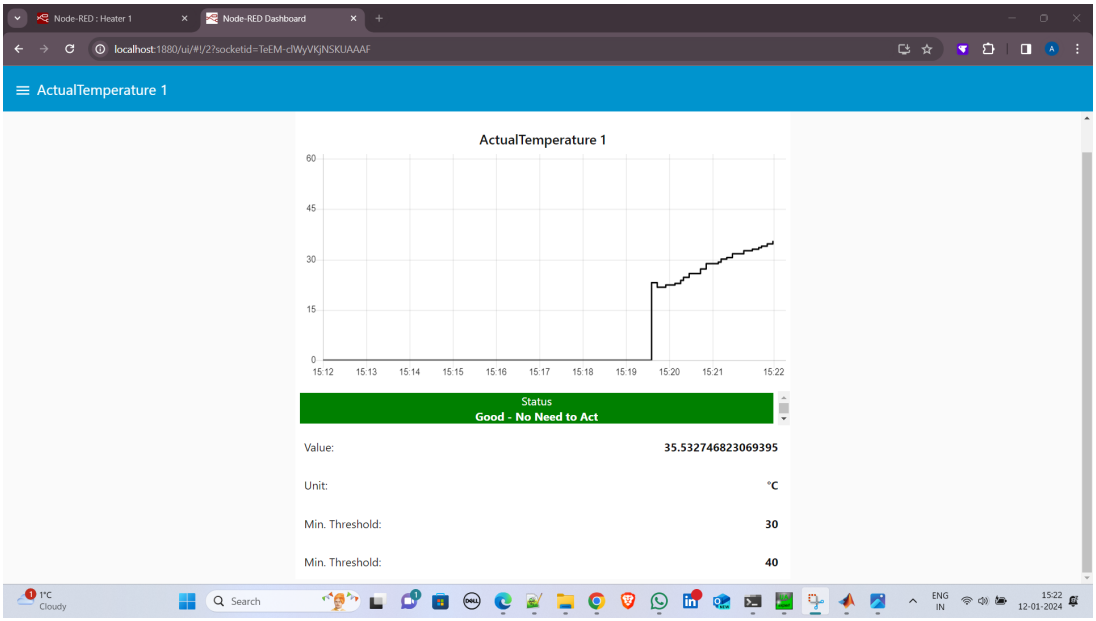


Figure 3.4.2: Within Threshold

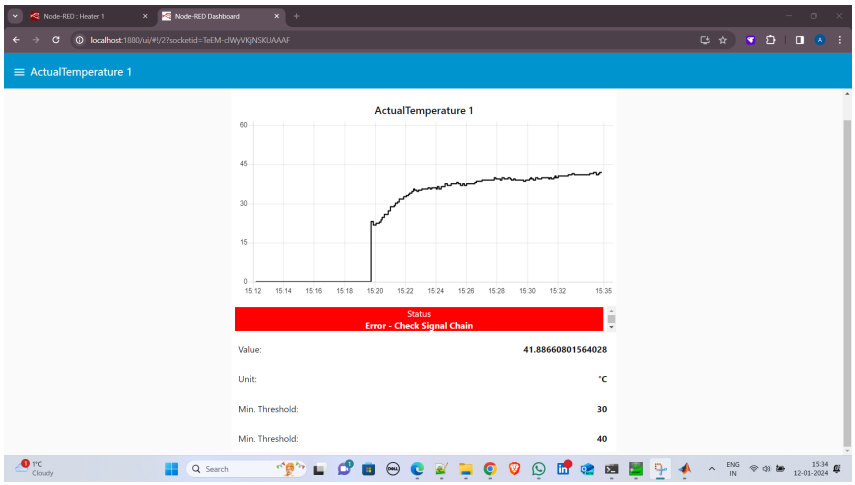


Figure 3.4.3: Above Threshold

# 4 Unit conversion from °C to °F

An additional function block has been incorporated to convert the temperature from degrees Celsius to Fahrenheit as shown in fig 4.0.1. Consequently, adjustments have been made to the graph, units, and threshold values.The dashboard displays the actual temperature output in degrees Fahrenheit, incorporating all graph, unit, and threshold modifications, as depicted below 4.0.2. The Conversion Code is written as below:

**var celsius= msg.payload;**
**msg.payload=(celsius*9/5)+32;**
**return msg;**

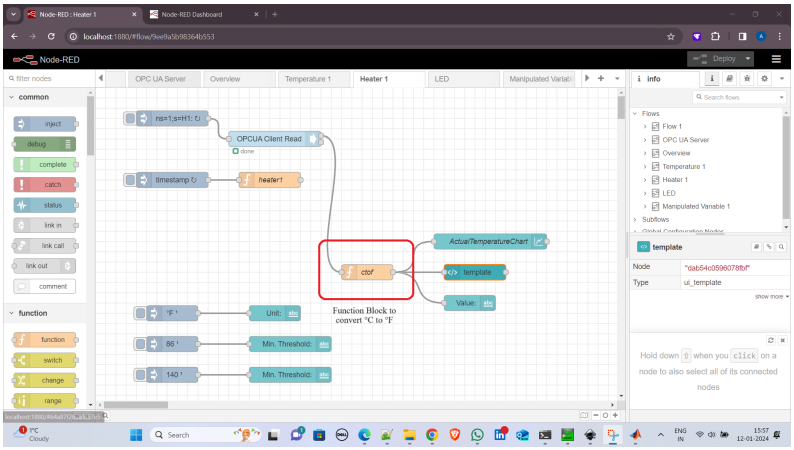


Figure 4.0.1: Function Block to convert unit from °C to °F

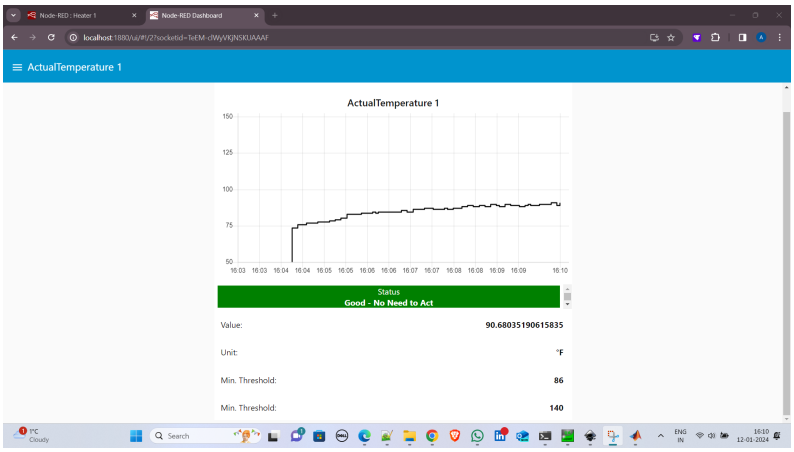


Figure 4.0.2: Output: actualTemperature_1 of °F

# 5 Conclusion

The experiment yielded hands-on experience in the design of an information model for a Temperature Control System and the implementation of OPC UA using Node-RED.

The three-step process began with the successful establishment of a connection between the server-client system, displaying input set temperature and actual output temperature on the client's dashboard. Subsequently, adjusting threshold values for the actual temperature allowed a detailed system status analysis. Beyond the thresholds, the system signaled "Error: Check Signal Chain," showing real-world scenarios, while within the thresholds, a consistent "Good: No Action Needed" status was observed across various variable nodes. Lastly, the integration of an additional function block in Node-RED facilitated the conversion of output temperature from degrees Celsius to Fahrenheit.

This experiment underscored the real-time monitoring and optimization capabilities of the actual system.

# Bibliography


- Kudzai Manditereza, Blog *OPC UA Information Model – How an OPC UA Information Model Works*, `https://www.industry40.tv/opc-ua-information-model-how-an-opc-ua-` July 9, 2021.
- Handout TCL 3.