Chapter 1
Automated Warehouse Shelf Space Monitoring and Warning System

1.1 INTRODUCTION

In today's fast-paced and competitive industrial environment, efficient warehouse management is crucial for optimising space utilisation, ensuring safety, and responding swiftly to potential hazards. Automated systems are key in enhancing productivity and safety by monitoring and controlling various aspects of warehouse operations. This document outlines the implementation of an Automated Warehouse Shelf Space Monitoring and Warning System using CODESYS, a versatile automation software suite for industrial control systems. The system aims to increase the efficiency of warehouse shelf space and temperature monitoring, reduce reliance on humans for generic monitoring tasks, and provide real-time visualisation of shelf space utilisation and warehouse conditions.

1.2 PROBLEM BACKGROUND

Warehouses face significant challenges related to space utilisation, safety, and environmental monitoring, leading to inefficiencies and increased operational costs. Effective shelf space management requires precise and real-time data on occupancy to ensure correct storage and prevent overloading. Manual tracking of shelf occupancy can be labour-intensive and error-prone, resulting in underutilization or overloading. Additionally, maintaining optimal temperature conditions is critical, especially for temperature-sensitive goods, as high temperatures can cause spoilage, equipment malfunctions, or fire hazards. Continuous temperature monitoring and prompt alerts are necessary to address safety hazards before they escalate. Furthermore, automation reduces manual interventions, minimises human error, and improves response times to changing warehouse conditions, enhancing operational efficiency. Therefore, there is a need for an automated system that can monitor shelf space and

temperature in real-time, reducing the reliance on human intervention and providing accurate and timely data for better decision-making.

1.3 OBJECTIVES

1. To increase the efficiency of warehouse shelf space and temperature monitoring.
2. To reduce reliance on humans to perform generic monitoring tasks.
3. To allow real time visualisation of shelf space utilisation and warehouse conditions.

1.4 METHODOLOGY

The Automated Warehouse Shelf Space Monitoring and Warning System utilises CODESYS to integrate various sensors and indicators for efficient monitoring and control. The system includes proximity sensors for real-time shelf occupancy monitoring, a temperature sensor for continuous environmental monitoring, and alarm mechanisms for safety alerts. The process begins with pressing the ONPB switch to activate the system, enabling all rungs in the ladder logic and making it ready to listen for sensor and temperature inputs. Proximity sensor switches simulate shelf occupancy, and corresponding indicator lamps light up when sensors are triggered. The temperature is adjusted using a potentiometer, and a bar display shows the current reading. When the temperature exceeds 80°C, the alarm lamp lights up, indicating a high-temperature alert. Pressing the STOPPB switch stops all operations, turning off the indicator lamps and halting temperature monitoring, even if the temperature is high. The system's visualisation in CODESYS facilitates understanding and testing by providing real-time feedback and ensuring correct behaviour under various conditions. This automated approach increases efficiency, reduces human dependency, and allows real-time monitoring and visualisation of warehouse conditions.

1.4.1 SYSTEM ARCHITECTURE

Figure 1.4.1 shows the system architecture of this system which consists of proximity sensors, temperature sensor, start & stop push buttons and camera as inputs, CODESYS PLC as controller, Node JS as server, SQL database as local database, website used as SCADA and alarm as output. The proximity sensor is used to detect the availability of shelf space

while the temperature sensor is used to measure the temperature. These inputs will then be displayed on Web SCADA. Besides, start and stop push buttons are used to control the power status of the system while the camera is feeding live video. Codesys PLC is a simulated industrial computer used to monitor inputs, make decisions based on its executed program, and produces outputs to automate processes. Node JS acts as a server to collect output data from PLC and store them in the SQL database. SQL database is functioning as a storage that keeps all data for recording and reporting purposes. Web SCADA displays all shelf space availability, power status, alarm status, and graphs that visualise system data including shelf space occupancy and temperature.

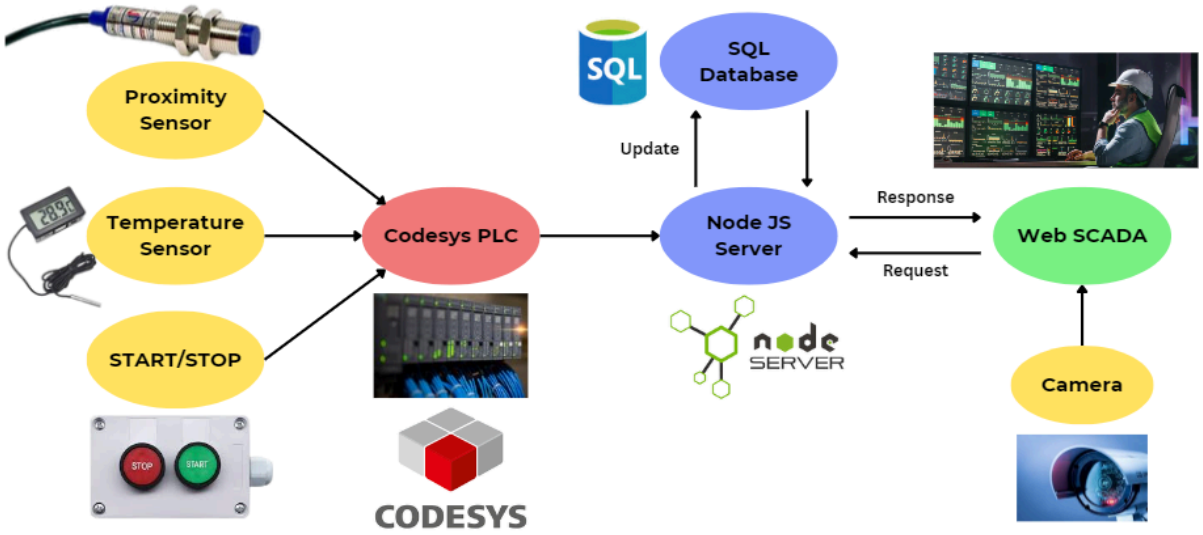


Figure 1.4.1 System Architecture

1.4.2 FLOWCHART

Figure 1.4.2 shows the flow of the system. First, the sensors and timer are initialised when the ON pushbutton is pressed. The current time, Timenow is set to 0. The time period for the timer is 3600 000 milliseconds, which is an hour. When Timenow is less than the timer period, the PLC will do nothing but the update Timenow. When Timenow is greater than period, which means a full timer period of time has passed, the PLC will check all the sensor status and update them to the database. After that, Timenow is resetted to 0 and proceed to the next cycle. The above steps are repeated every hour until the OFF pushbutton is pressed. Then, the PLC will shutdown and close all the sensors.

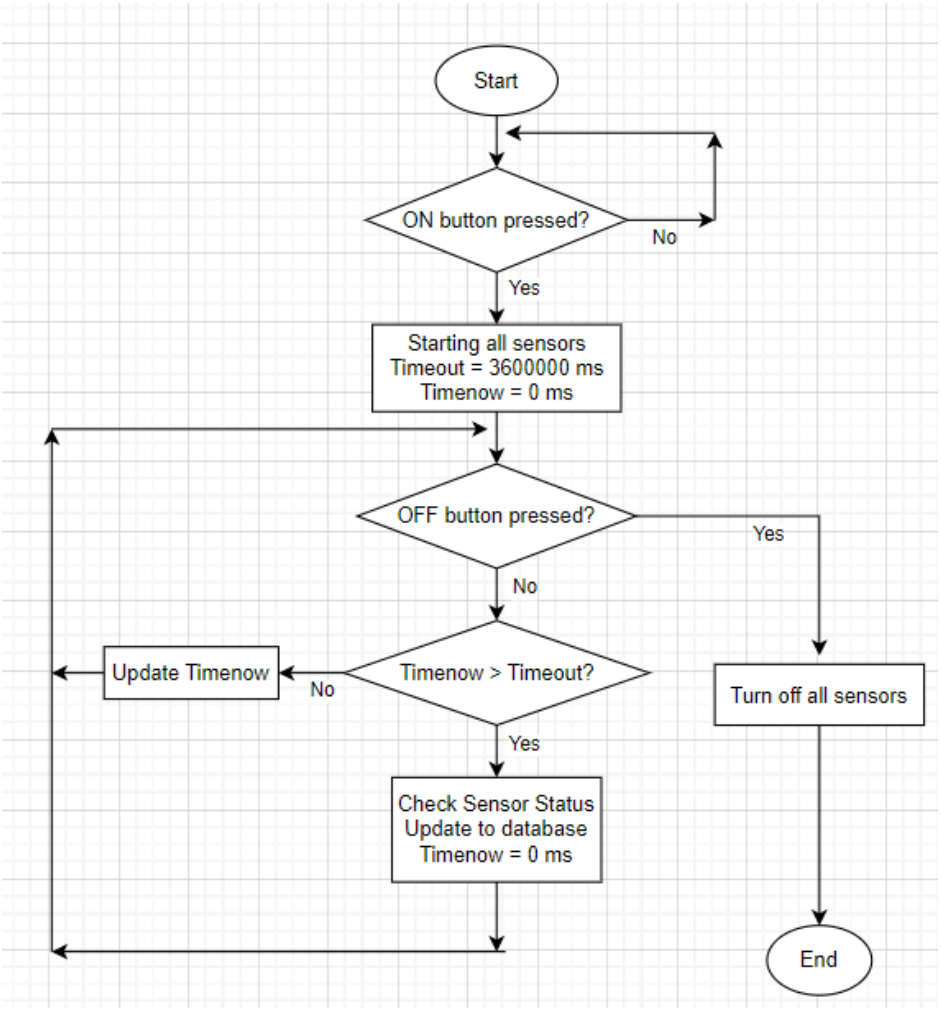


Figure 1.4.2 Flowchart

1.4.3 STATE DIAGRAM

Based on the system architecture and the flowchart, the state diagram was designed as shown in Figure 1.4.3. When the ON pushbutton is not pressed, the system is in OFF state. All the sensors and the backend server are closed in this state. When the ON pushbutton is pressed, the system will advance into ON state. In this state, all sensors are turned on and the timer is initialised. When the OFF pushbutton is pressed, the system will go back to OFF state. After the ON state, the system will be in an idle or waiting state, where the system does nothing until the timer period has passed. When the timer period is satisfied, the system will advance to the active state where the PLC checks all the status of sensors and uploads them into the database. After the active state, it will return to the waiting state until the next timer period is satisfied. The process repeats until the OFF pushbutton is pressed.

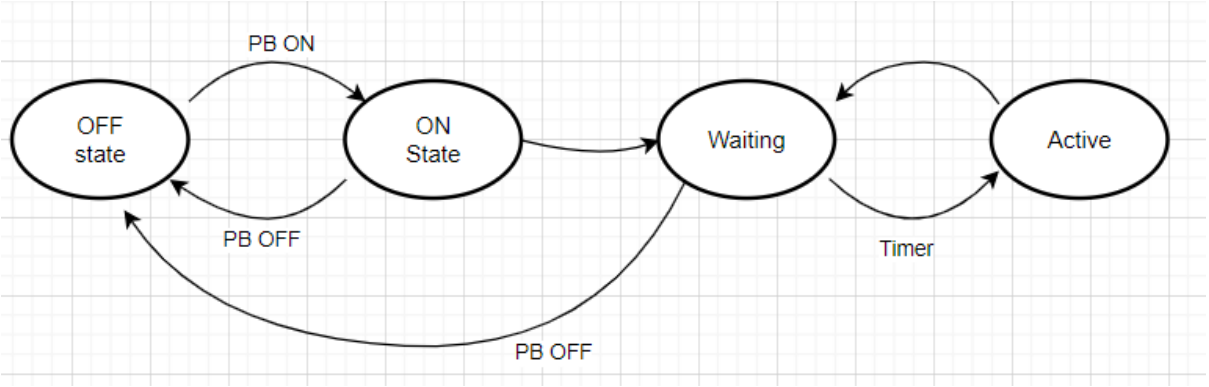


Figure 1.4.3 State diagram

Table 1.4.1 explains the symbols used in the state diagram. The output equations were derived from the state diagram, as shown in Table 1.4.2.

Table 1.4.1 Symbols used in the state diagram

Input	Output
RG = Rain Gauge Sensor	Alert_1 = Level 1 Alert
FR = Ultrasonic Water Flow Meter	Alert_2 = Level 2 Alert
L = Water Level Detector	A1 = Alarm Ring Bell
S = Start Button	B1 = Ball Valve Actuator
Em_Stop = Stop Button	AC_1 = Close Ball Valve Actuator

Table 1.4.2 Output equations

State 0	Em_Stop
State 1	$S{\cdot}\underline{Em\_Stop}$
State 2	$S{\cdot}\underline{Em\_Stop} + L$
State 3	$S{\cdot}\underline{Em\_Stop} + RG$
State 4	$S{\cdot}\underline{Em\_Stop} + FR$

1.4.4 CODESYS

CODESYS is a flexible yet powerful industrial automation platform with its extensive toolkit for control system programming, visualisation, testing, and deployment. It is the best option for a variety of automation applications due to its cross-platform interoperability, support for numerous programming languages, and strong development tools. CODESYS offers the efficiency and flexibility required to improve the performance of automation solutions and expedite the development process, regardless of the size of the project or system being worked on.

Figure 1.4.4 shows CODESYS interface for ladder diagram programming in building automated warehouse shelf space monitoring and warning systems.

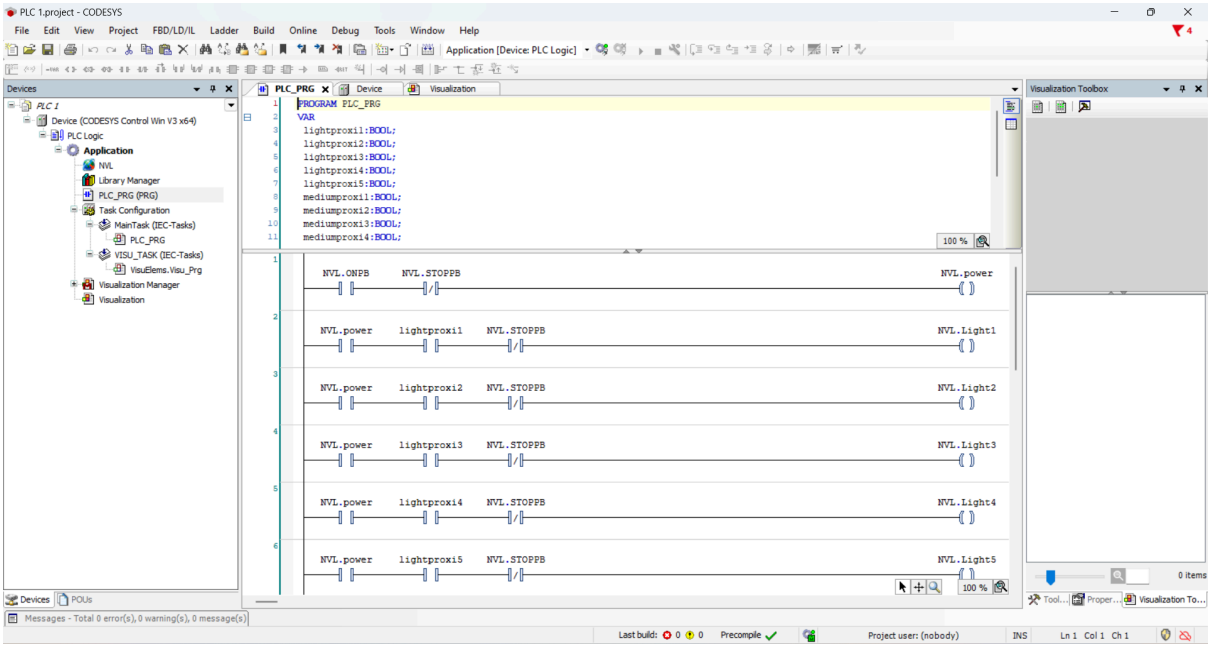


Figure 1.4.4 CODESYS interface

Figure 1.4.5 shows the visualisation interface for the control panel of the automated warehouse shelf space monitoring and warning systems.

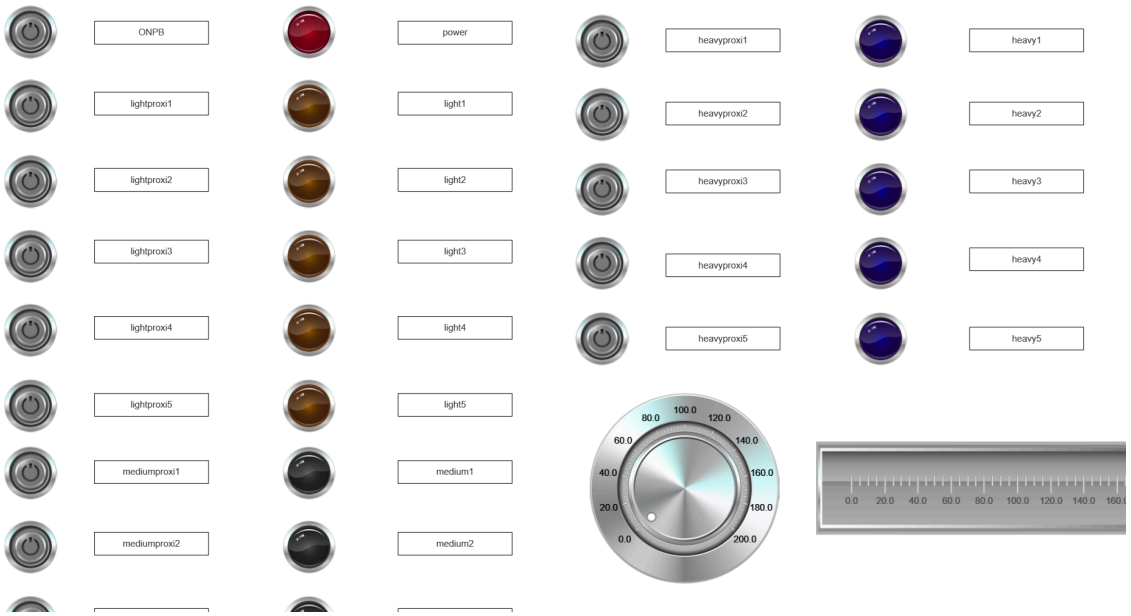


Figure 1.4.5 CODESYS Visualisation

1.4.5 EXPRESS.JS

Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It is widely used for building web applications and APIs due to its simplicity and ease of use. Its robust set of features, middleware support, and flexible routing system make it an excellent choice for developers looking to build high-performance, scalable, and maintainable server-side applications. With its active community and extensive ecosystem, Express.js shown in Figure 1.4.6 is our optimal choice for web development in building Web SCADA for our system.

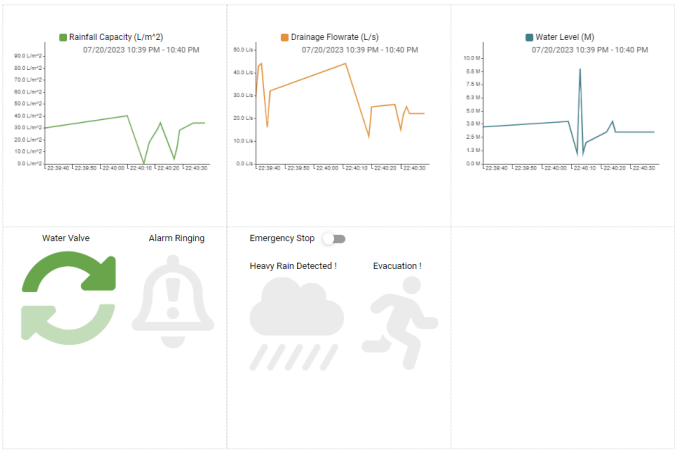


Figure 1.4.6 Express.js

1.4.6 REACT.JS

React.js, commonly referred to as React, is a popular open-source JavaScript library used for building user interfaces, particularly for single-page applications where data changes over time. It was developed by Facebook and is maintained by Facebook and a community of individual developers and companies. Figure 1.4.6 shows an example interface developed using React.js..

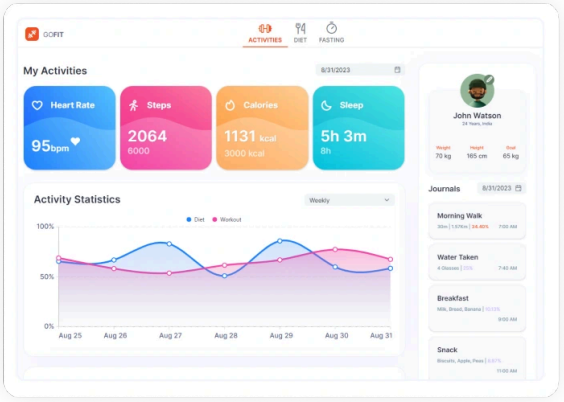


Figure 1.4.6 Web Dashboard created using React.js

1.5 RESULT AND DISCUSSION

The automated warehouse shelf space monitoring and warning system for warehouse management of this project is developed and tested using CODESYS with ladder diagram programming technique. Then, the CODESYS visualization tool is used to demonstrate and simulate the process and operation of the PLC system.

1.5.1 LADDER DIAGRAM PROGRAMMING AND DATA VISUALIZATION

The functions in automated warehouse shelf space monitoring and warning systems are achieved with different layers of PLC rungs. The global network variables are in Figure 1.5.1. The ladder diagrams are shown in Figure 1.5.2.

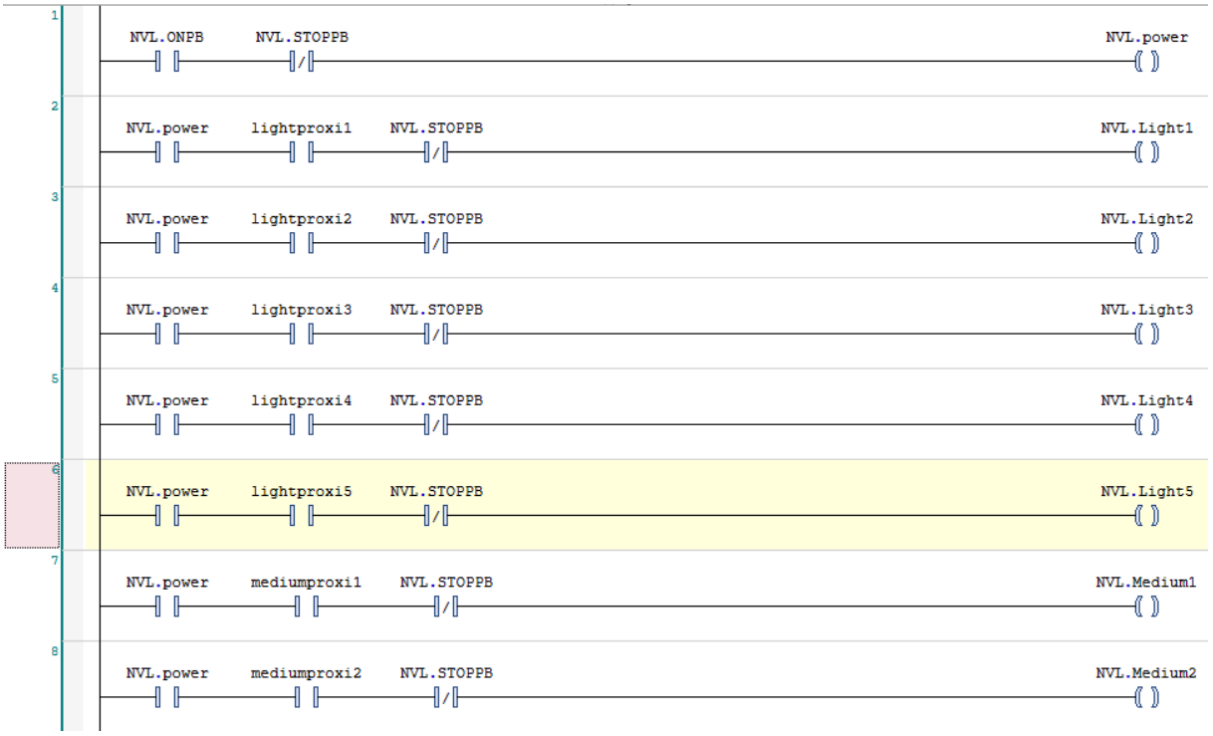
```
VAR_GLOBAL
  ONPB: BOOL;
  STOPPB: BOOL;
  Light1: BOOL;
  Light2: BOOL;
  Light3: BOOL;
  Light4: BOOL;
  Light5: BOOL;
  Medium1: BOOL;
  Medium2: BOOL;
  Medium3: BOOL;
  Medium4: BOOL;
  Medium5: BOOL;
  Heavy1: BOOL;
  Heavy2: BOOL;
  Heavy3: BOOL;
  Heavy4: BOOL;
  Heavy5: BOOL;
  temp: INT;
  alarm: BOOL;
  power: BOOL;

END_VAR
```

```
PROGRAM PLC_PRG
VAR
 lightproxi1:BOOL;
 lightproxi2:BOOL;
 lightproxi3:BOOL;
 lightproxi4:BOOL;
 lightproxi5:BOOL;
 mediumproxi1:BOOL;
 mediumproxi2:BOOL;
 mediumproxi3:BOOL;
 mediumproxi4:BOOL;
 mediumproxi5:BOOL;
 heavyproxi1:BOOL;
 heavyproxi2:BOOL;
 heavyproxi3:BOOL;
 heavyproxi4:BOOL;
 heavyproxi5:BOOL;
END_VAR
```

Figure 1.5.1 Global Variables in CODESYS

A total of 20 network variables and 15 local variables are defined. 19 of the network variables are boolean variables and only one is an integer variable. On the other hand, the 15 local variables are all boolean. **ONPB** and **STOPPB** are used to turn on and turn off the PLC. The local variables **lightproxi1, lightproxi2, lightproxi3, lightproxi4** and **lightproxi5** represent the proximity sensor of the shelves. The status of the occupancy will be shown in

the variables **Light1, Light2, Light3, Light4** and **Light5**. These variables control the sensor indicators for the shelves of lightweight objects. The local variables **mediumproxi1, mediumproxi2, mediumproxi3, mediumproxi4** and **mediumproxi5** represent the proximity sensor of the shelves. The status of the occupancy will be shown in the variables **Medium1, Medium2, Medium3, Meidum4** and **Medium5**. These variables control the sensor indicators for the shelves of medium weight objects. The local variables **heavyproxi1, heavyproxi2, heavyproxi3, heavyproxi4** and **heavyproxi5** represent the proximity sensor of the shelves. The status of the occupancy will be shown in the variables **Heavy1, Heavy2, Heavy3, Heavy4** and **Heavy5**. These variables control the sensor indicators for the shelves of heavyweight objects. **Temp** represents the temperature of the warehouse, **alarm** is the alert mechanism when the temperature inside the warehouse is too high and **power** is the indicator that lights up after **ONPB** is pressed.



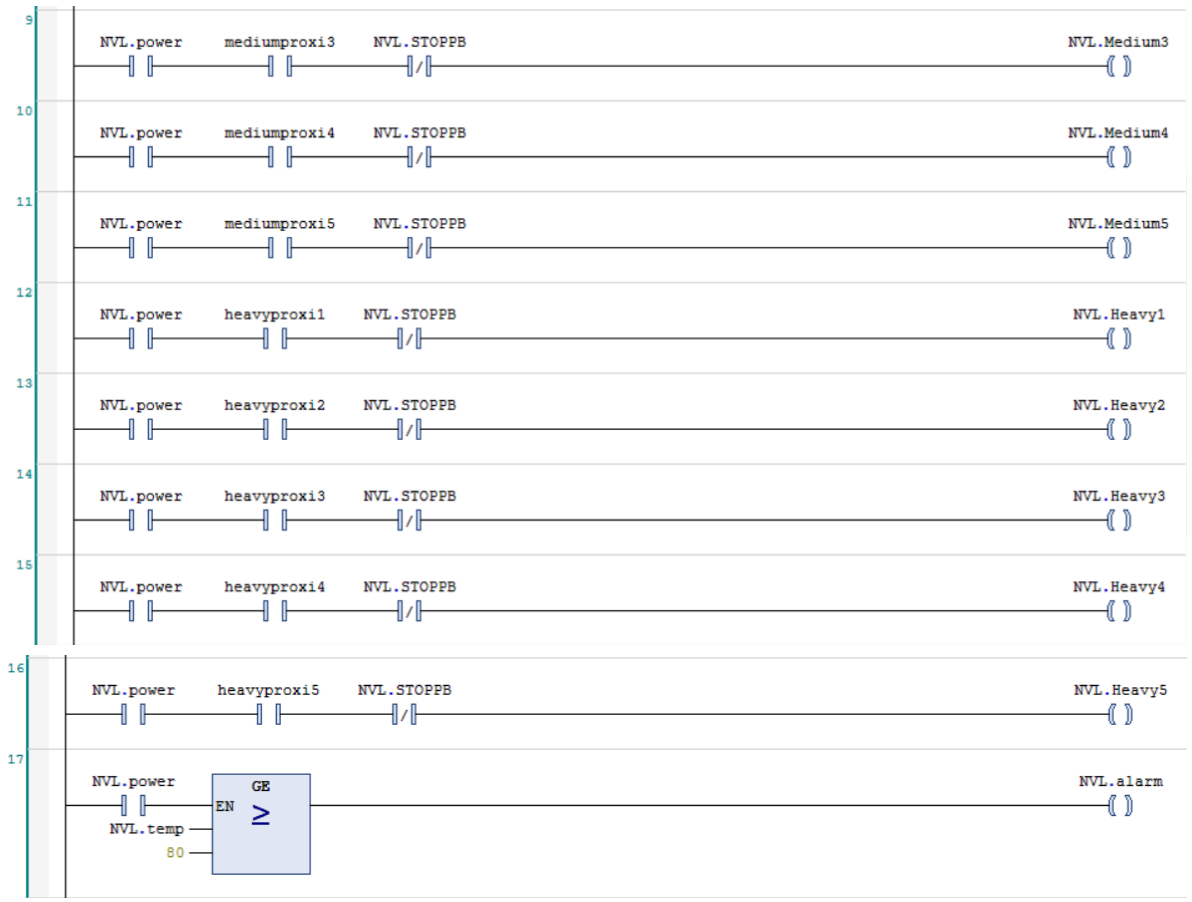
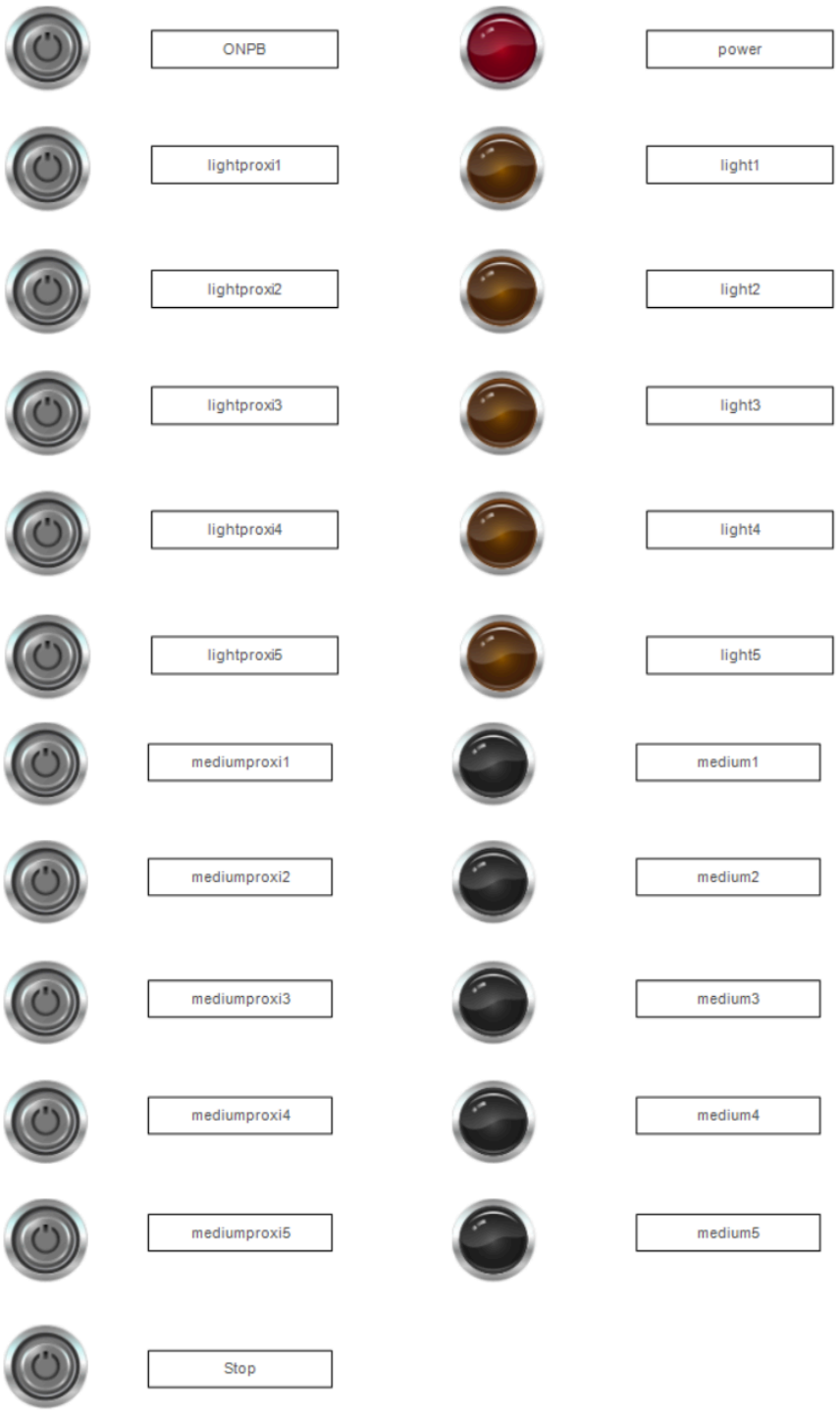


Figure 1.5.2 Ladder Diagram in CODESYS

Figure 1.5.2 shows how the PLC works. First and foremost, when ONPB is triggered, the power indicator will light up. After the power indicator is lit up, all the rungs are activated and actively listening for proximity sensor input and temperature input. When any of the proximity sensors are triggered, the corresponding sensor indicator lights up. For example, when lightproxi1 is triggered, Light1 will light up. When the temperature input received is less than 80°C, there will be no change. However, when the temperature exceeds 80℃, the alarm is triggered.


ONPB
power
lightproxi1
light1
lightproxi2
light2
lightproxi3
light3
lightproxi4
light4
lightproxi5
light5
mediumproxi1
medium1
mediumproxi2
medium2
mediumproxi3
medium3
mediumproxi4
medium4
mediumproxi5
medium5
Stop

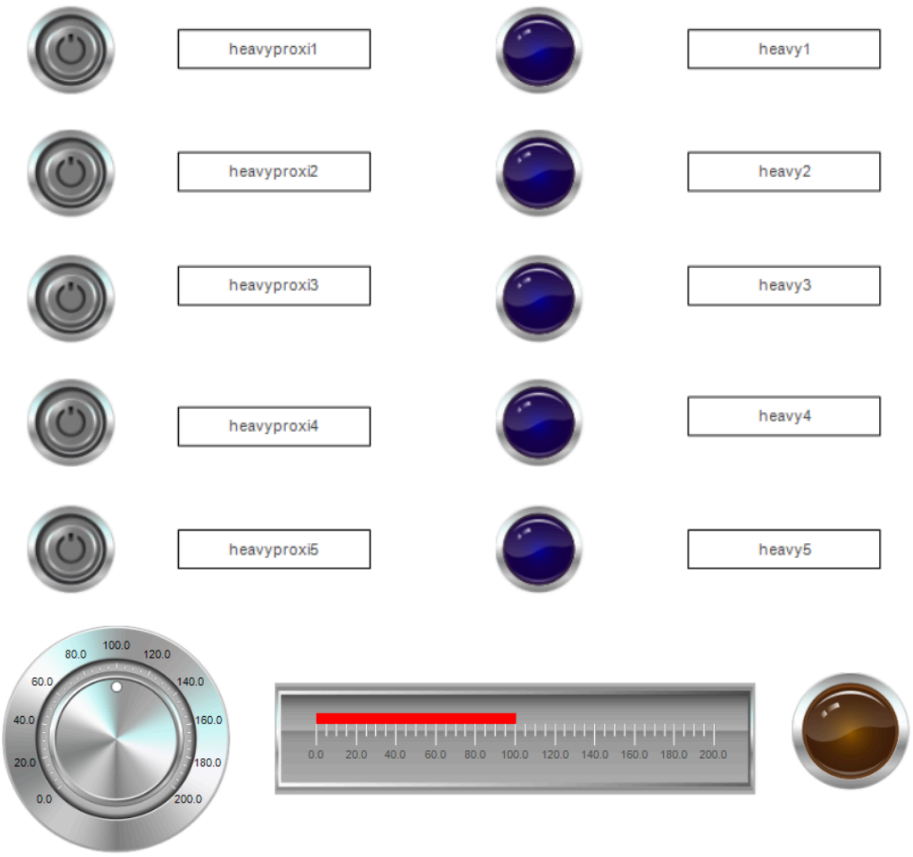


Figure 1.5.3 Visualization in CODESYS

The visualization function in CODESYS helps us better understand how the system will work. The ONPB and STOPPB switches are used to trigger the start and stop of the PLC respectively. The other switches are used to represent the proximity sensors for each shelf. Using CODESYS visualization, we will manually input using the switches. The indicators are represented using lamps. The temperature sensor input is represented using a potentiometer. The bar display shows the temperature input and a lamp is used to show that the alarm is triggered.

1.5.2 SIMULATION AND RESULT

For the simulation, first, the system is started by pressing the ONPB switch. Then, all rungs are activated and actively waiting for sensor and temperature input.

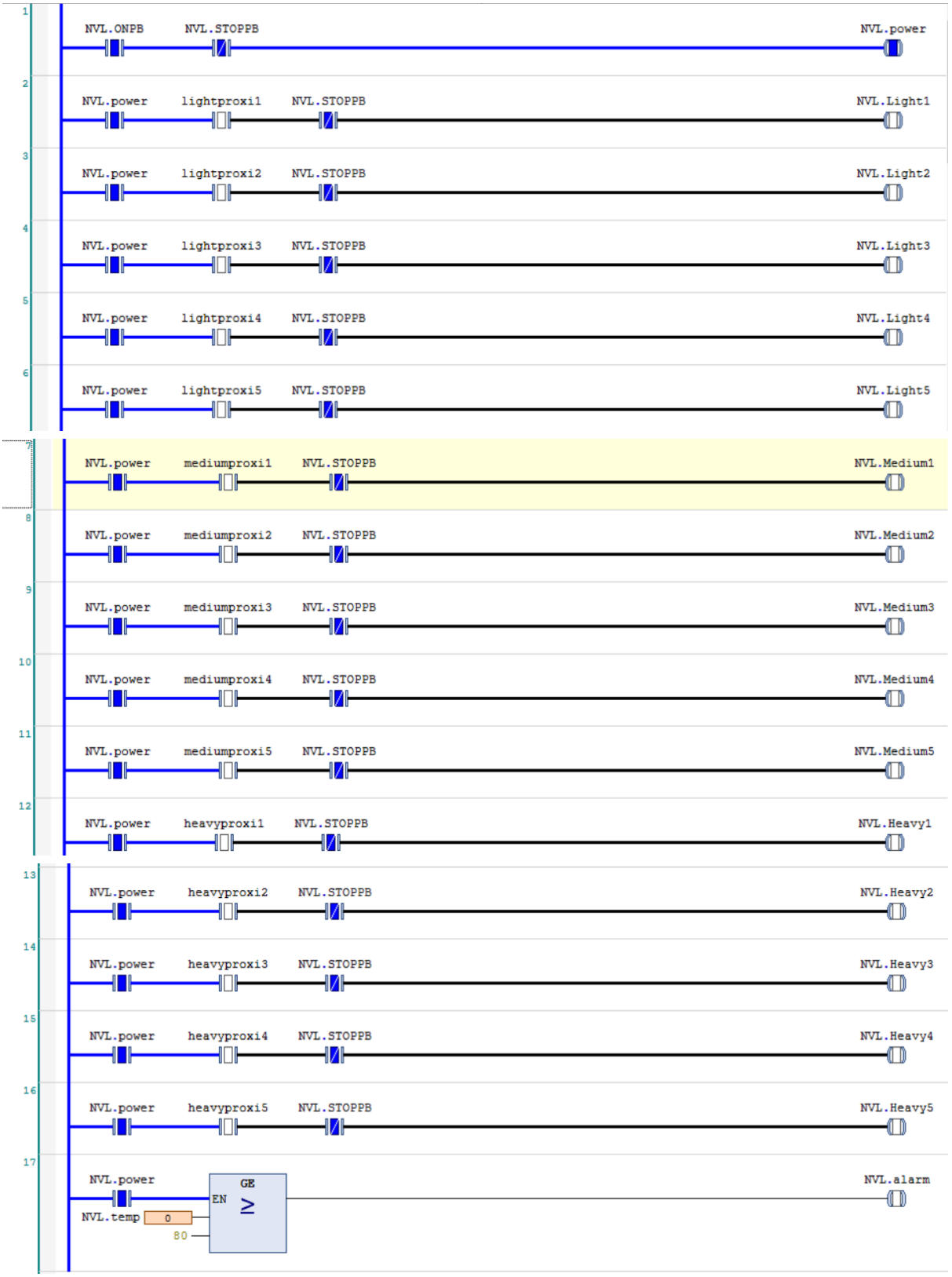


Figure 1.5.4 The ladder diagram after system is on

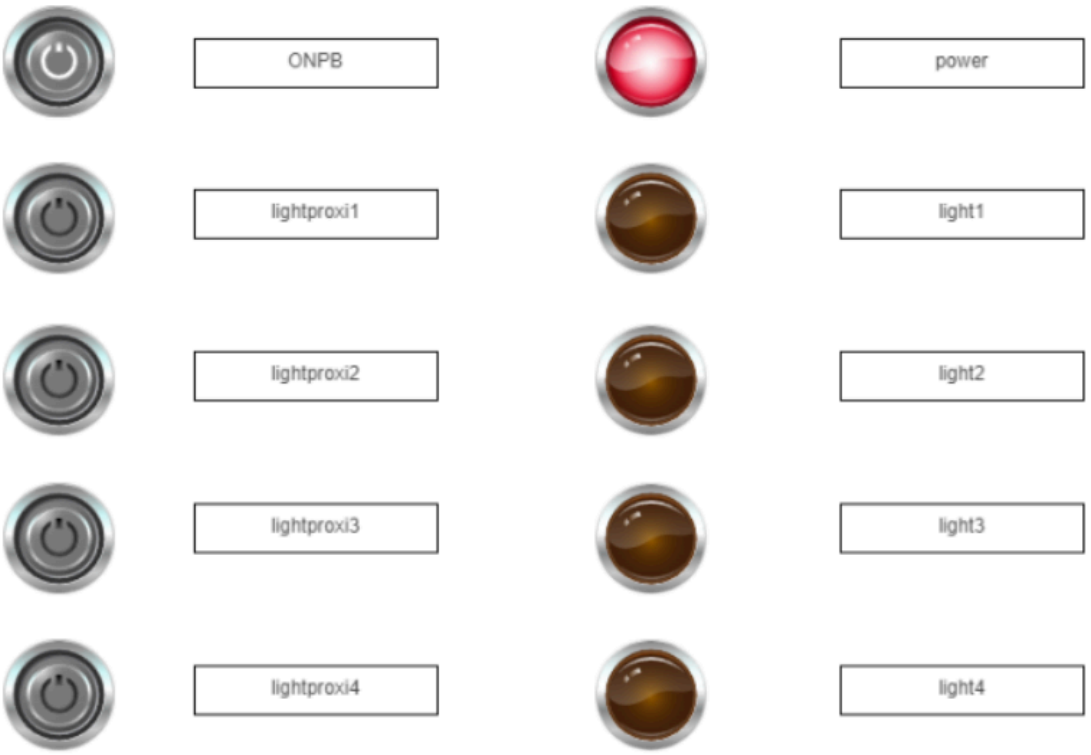
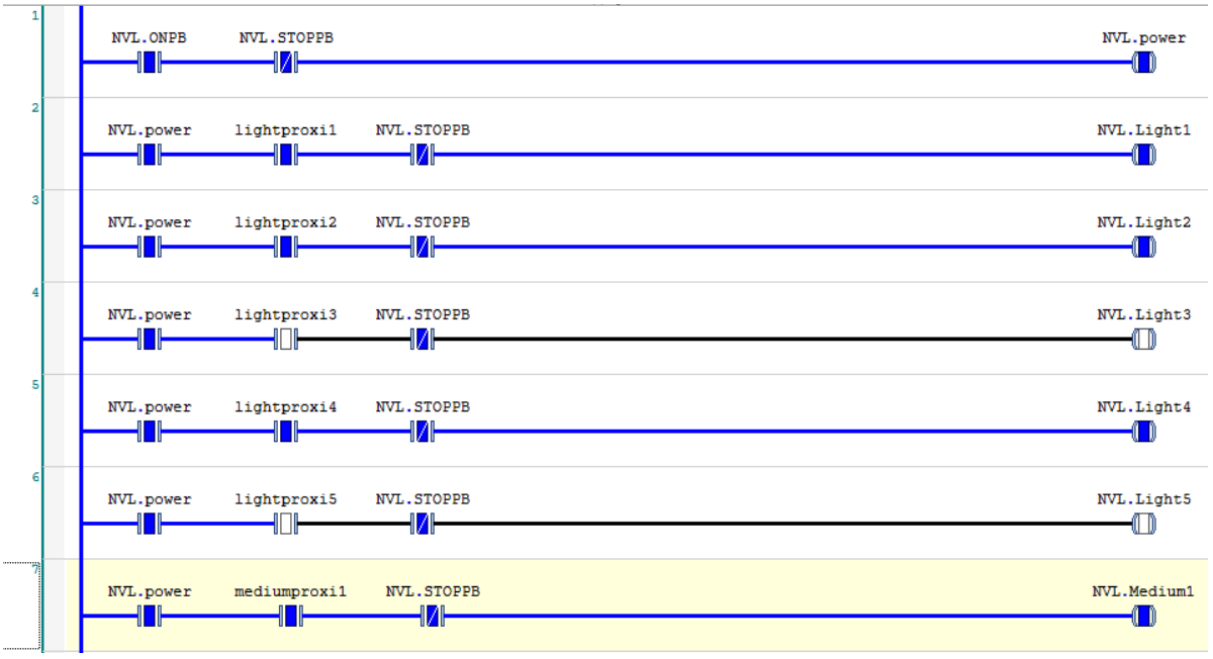


Figure 1.5.5 The visualization with on button triggered

When the proximity sensors are triggered, the corresponding sensor indicators will also light up. In the case in Figure 1.5.6, lightproxi1, lightproxi2, lightproxi4, mediumproxi1, mediumproxi2, mediumproxi3, heavyproxi4, heavyproxi5 were triggered. And it is observed that the corresponding indicators, which are Light1, Light2, Light4, Medium1, Medium2, Medium3, Heavy4 and Heavy5 light up.



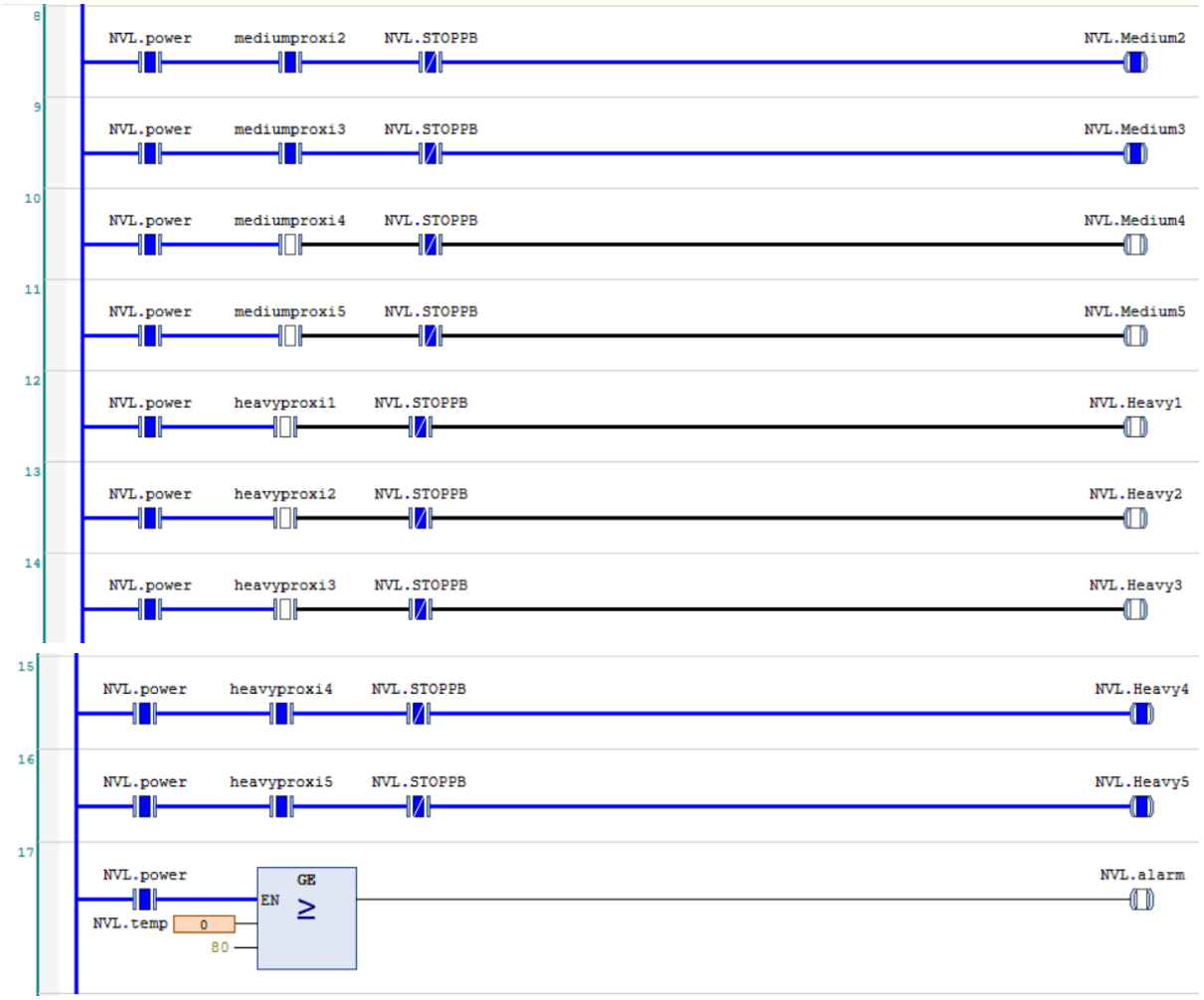


Figure 1.5.6 Ladder Diagram when sensors are triggered


ONPB
power
lightproxi1
light1
lightproxi2
light2
lightproxi3
light3
lightproxi4
light4
lightproxi5
light5
mediumproxi1
medium1
mediumproxi2
medium2
mediumproxi3
medium3
mediumproxi4
medium4
mediumproxi5
medium5

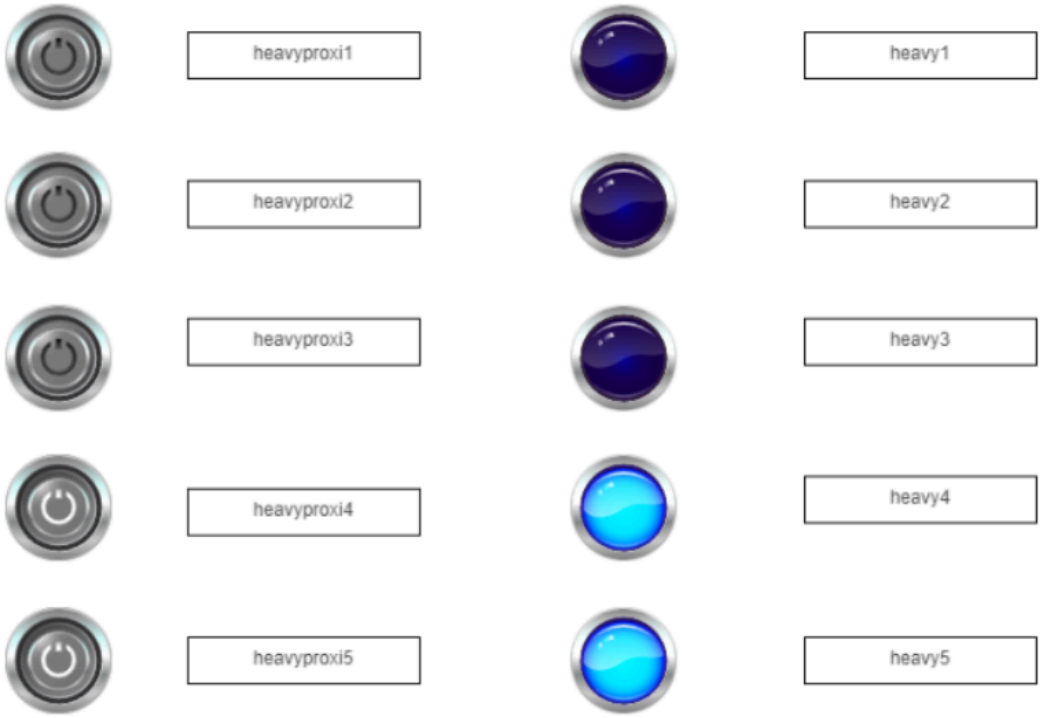


Figure 1.5.7 Visualization when proximity sensors triggered

The input temperature is now 109 ℃, well above the 80 ℃ level, the alarm is triggered. As you can see in the visualization, the lamp lights up when the temperature is higher than 80 ℃.

Figure 1.5.8 Ladder Diagram when temperature is higher than 80

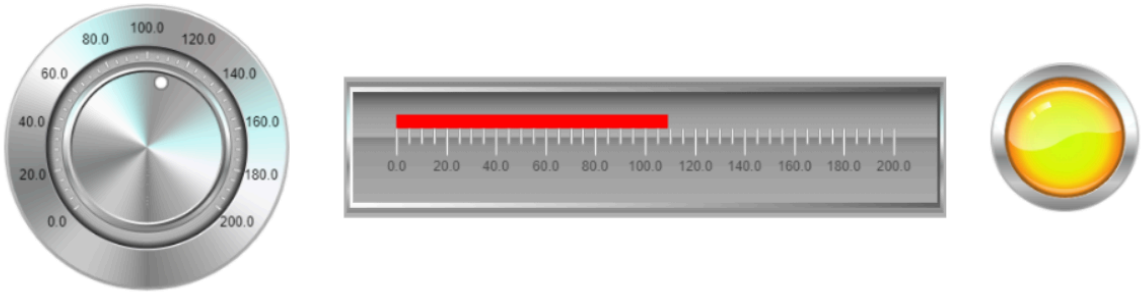


Figure 1.5.9 Visualization when temperature is higher than 80

All the operations are instantly stopped when the stop button is triggered. As you can see, despite the proximity sensors are triggered, the lamps do not light up. The temperature sensor function is also stopped when the stop button is pressed, although the temperature reading is high, the alarm lamp does not light up.

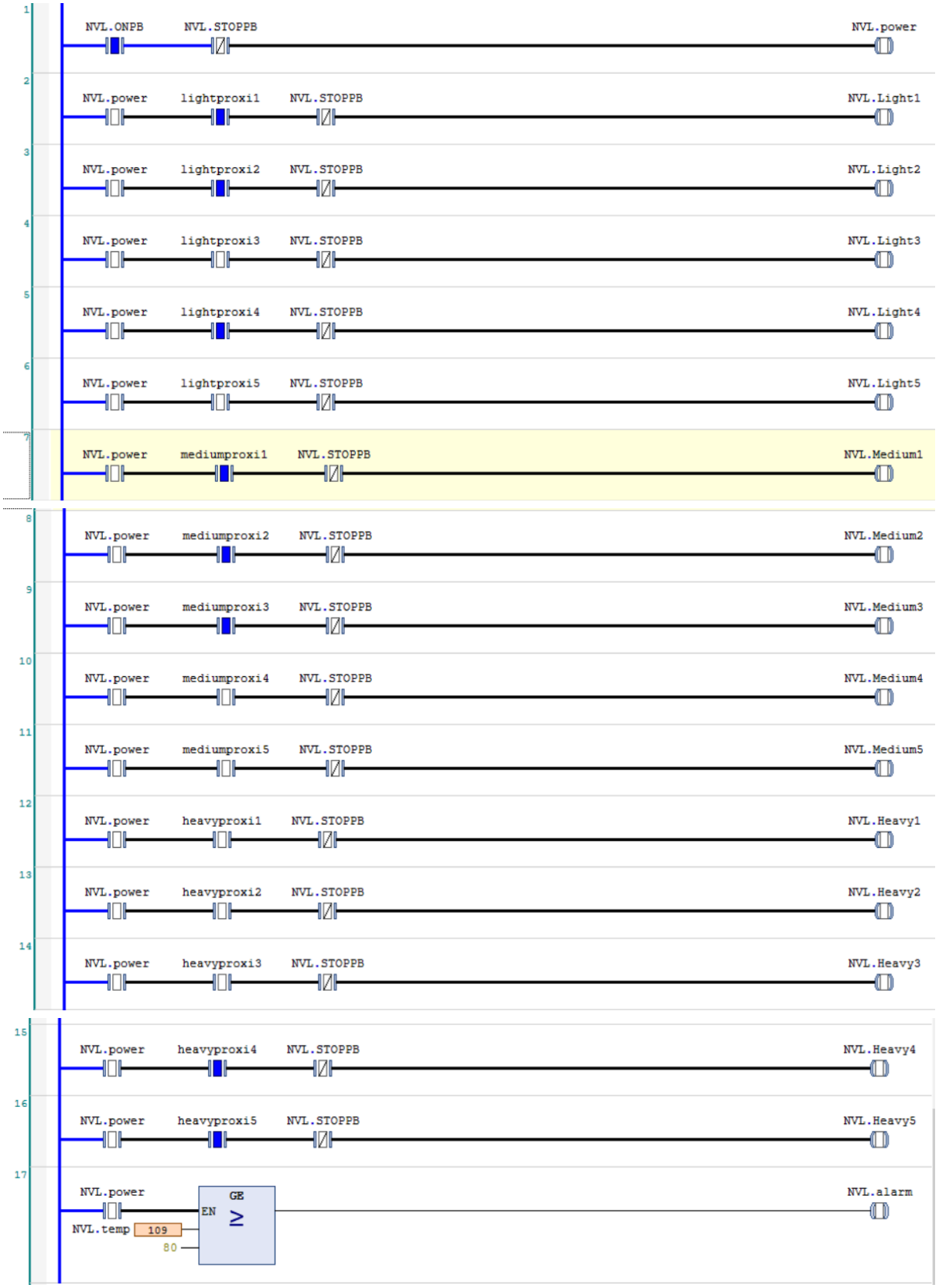
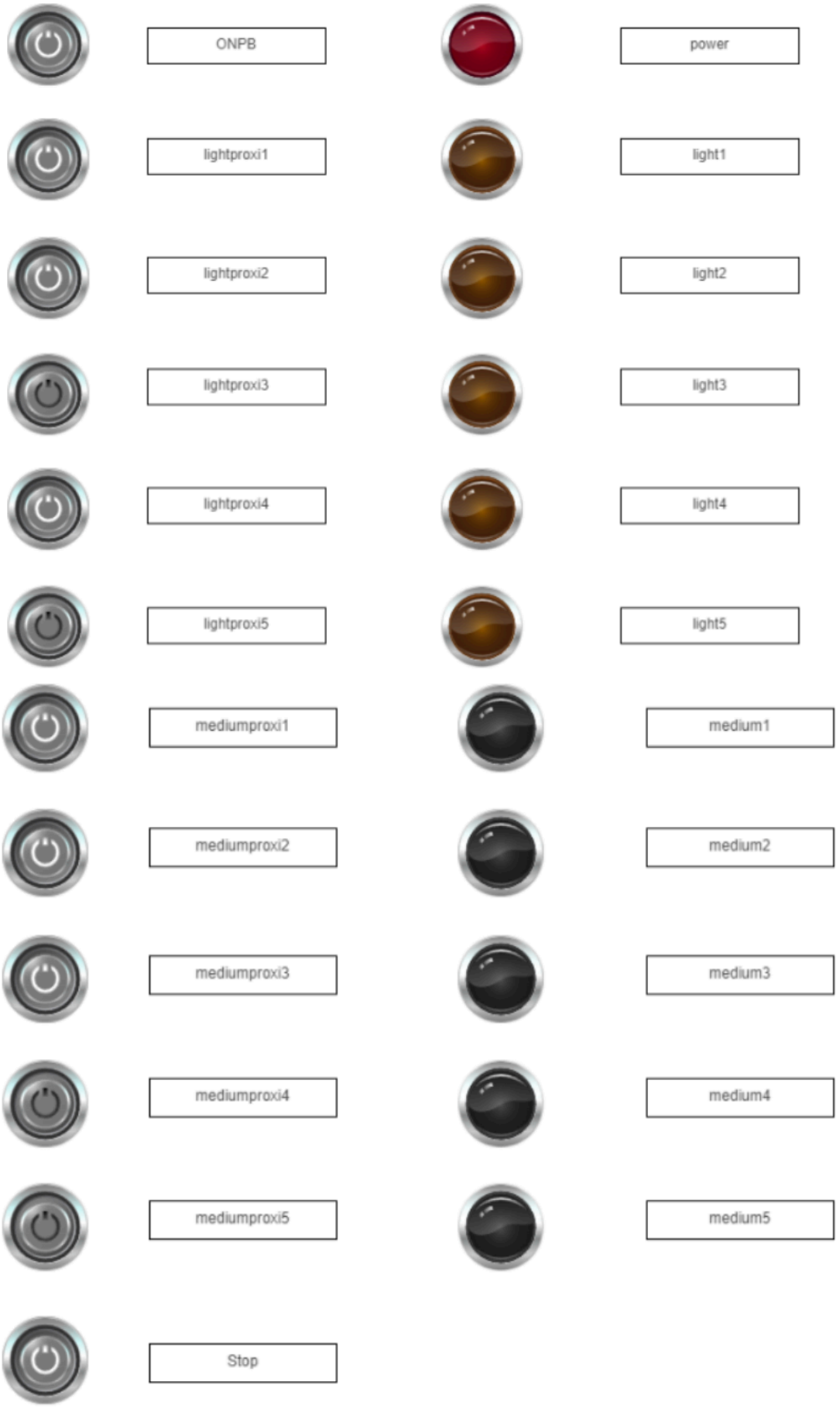


Figure 1.5.10 Ladder Diagram when stop button is triggered


ONPB
power
lightproxi1
light1
lightproxi2
light2
lightproxi3
light3
lightproxi4
light4
lightproxi5
light5
mediumproxi1
medium1
mediumproxi2
medium2
mediumproxi3
medium3
mediumproxi4
medium4
mediumproxi5
medium5
Stop

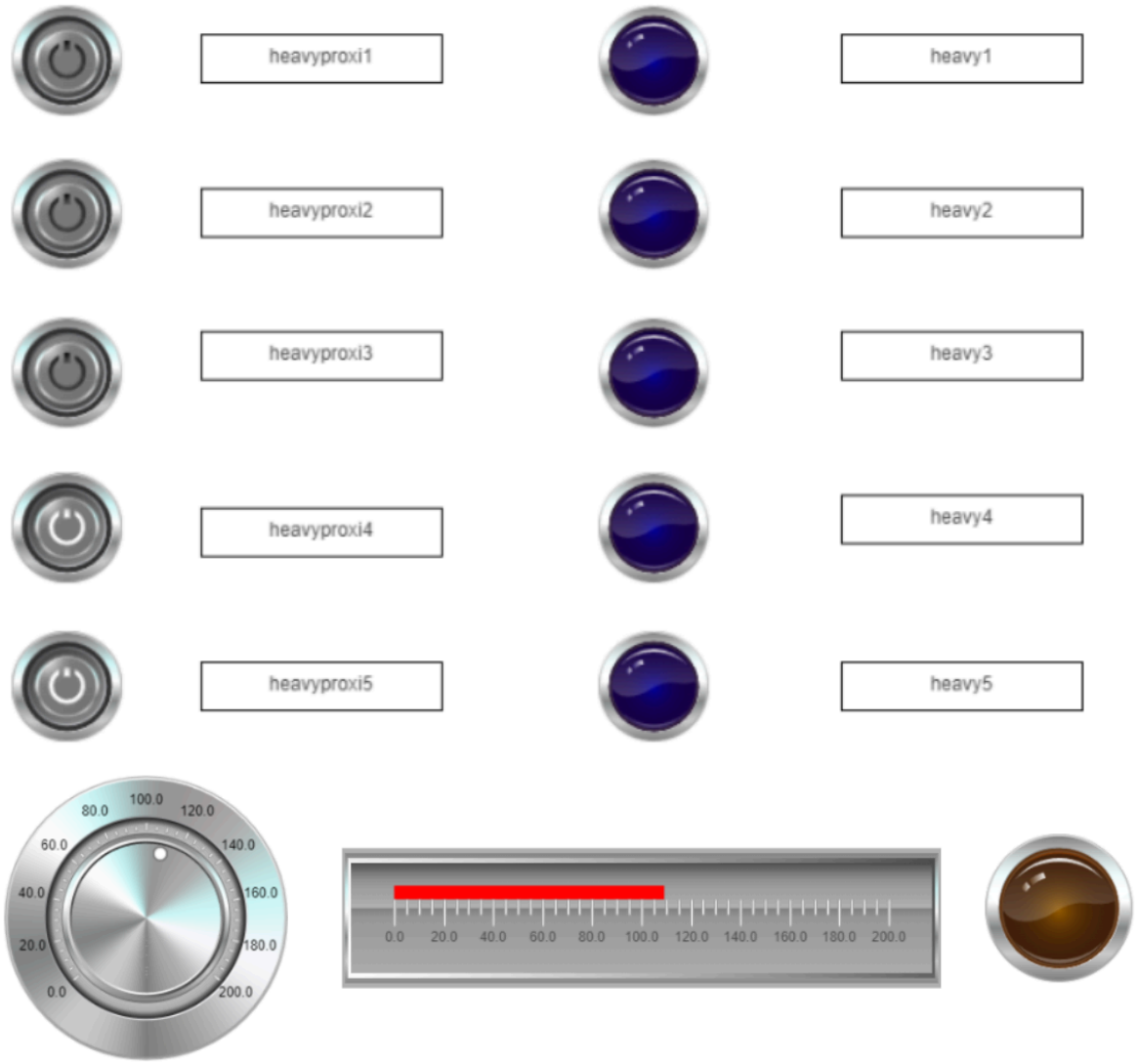


Figure 1.5.11 Visualisation when stop button is triggered

1.5.3 WEB DASHBOARD

In order to have remote monitoring and controlling of the system, a web dashboard is built using React.js. The dashboard consists of a home page, a records page, a temperature page and a camera page. The homepage consists of status indication and buttons to show and control the condition of the environment.

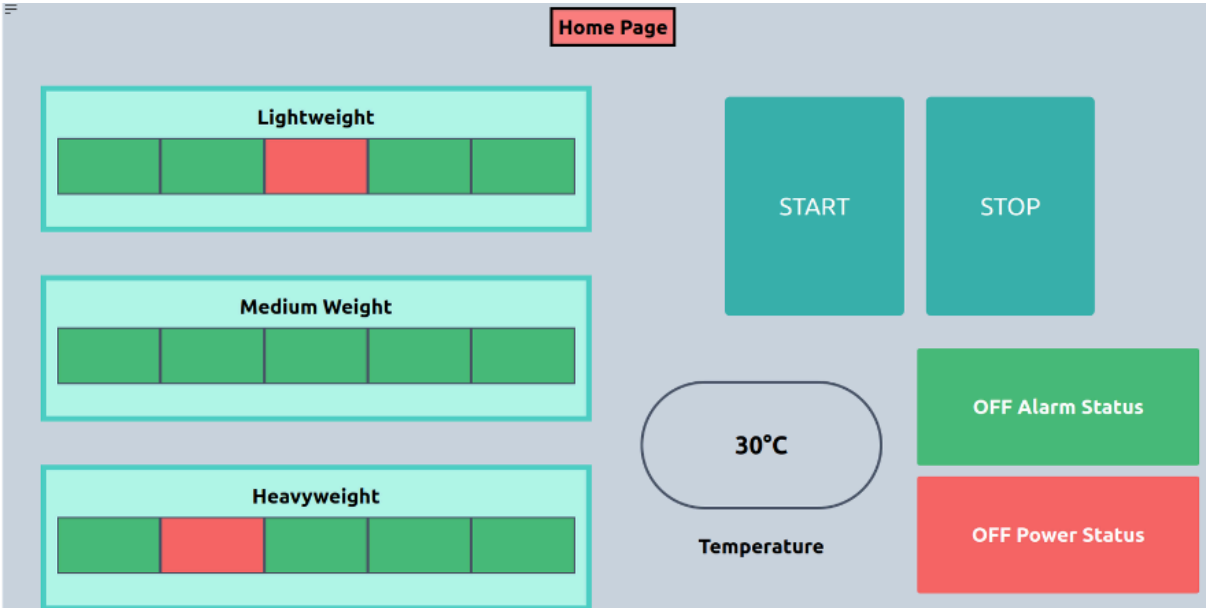


Figure 1.5.10 Home Page

The records page and the temperature page shows data from the sensors that are visualised in graphs. The records page shows the history of available space of each category in every hour shown in a stacked bar chart while the temperature page shows the temperature of the warehouse every hour in a line chart.

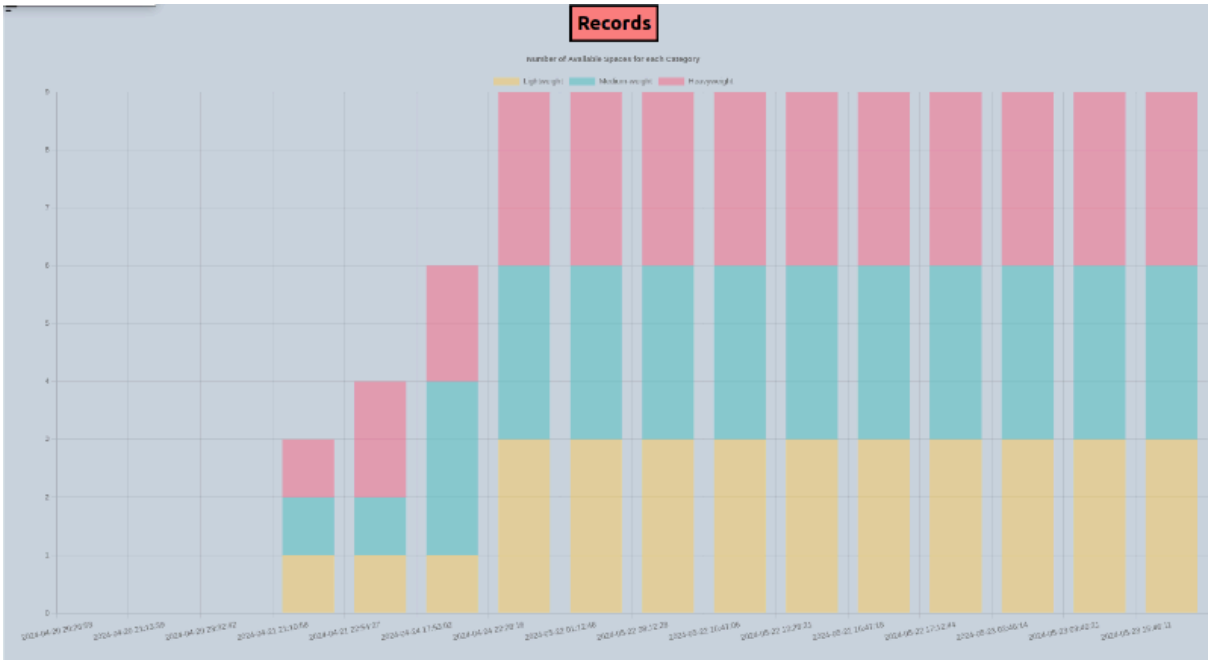


Figure 1.5.11 Records Page

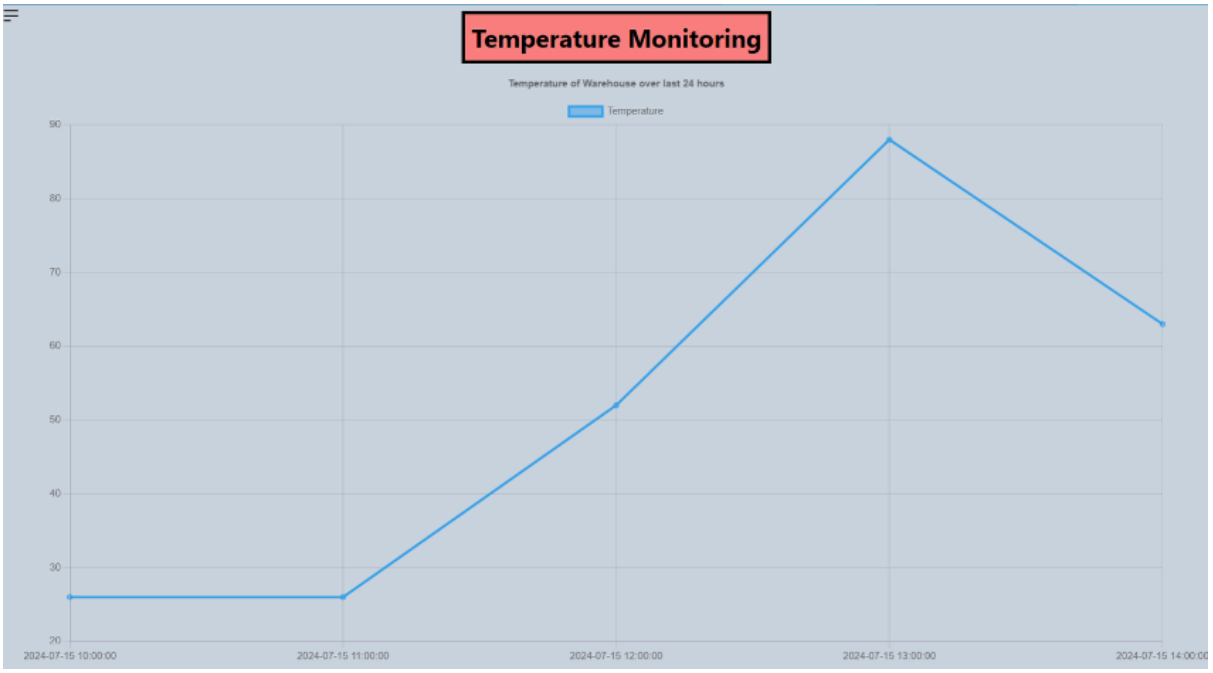


Figure 1.5.12 Temperature Page

The dashboard also provides a page for monitoring of the shelves from a camera directed at the shelves in the warehouse. Other than that, the dashboard offers a feature of capturing a photo of the videostream and saving it.

Figure 1.5.13 Camera Page

1.5.4 CONCLUSION

The implementation of an Automated Warehouse Shelf Space Monitoring and Warning System using CODESYS offers a robust solution for optimising warehouse management. By leveraging advanced automation technologies, the system effectively monitors shelf occupancy and temperature conditions in real-time, significantly enhancing operational efficiency and safety. The reduction in manual intervention minimises human error and labour costs, while the real-time visualisation capabilities provide immediate feedback on warehouse conditions, facilitating informed decision-making. This comprehensive approach addresses critical challenges in warehouse management, ensuring optimal use of space, timely hazard detection, and improved overall productivity. The successful deployment of this system underscores the potential of automation in transforming industrial operations, paving the way for more intelligent and efficient warehouse management practices.

REFERENCE

1. Palms Academy. (2021, June 23). What is Warehouse Management System? How WMS Works [Video]. YouTube. https://www.youtube.com/watch?v=_grpOkkd8p8
2. The Star. (2024). CelcomDigi partners ZTE for cutting-edge solutions in smart manufacturing and smart warehousing. https://www.thestar.com.my/business/business-news/2024/03/08/celcomdigi-partners-zte-for-cutting-edge-solutions-in-smart-manufacturing-and-smart-warehousing
3. Talk Business Magazine. (2024).The Evolution of Warehouse Operations: Trends and Technologies Shaping the Future. https://www.talk-business.co.uk/2024/02/16/the-evolution-of-warehouse-operations-trends-and-technologies-shaping-the-future/
4. Sasidhar, Kadiyam & Hussain, Shaik & Safdar, Syed & Uddin, Aleem. (2017). Design and Development of a PLC Based Automatic Object Sorting.
5. Adnan, M., Huda, N. U., & Zaman, U. K. U. (2022). Smart Warehouse Management System: architecture, Real-Time implementation and Prototype design. Machines, 10(2), 150. https://doi.org/10.3390/machines10020150

APPENDIX A: Meeting Minutes

Meeting Minutes 1

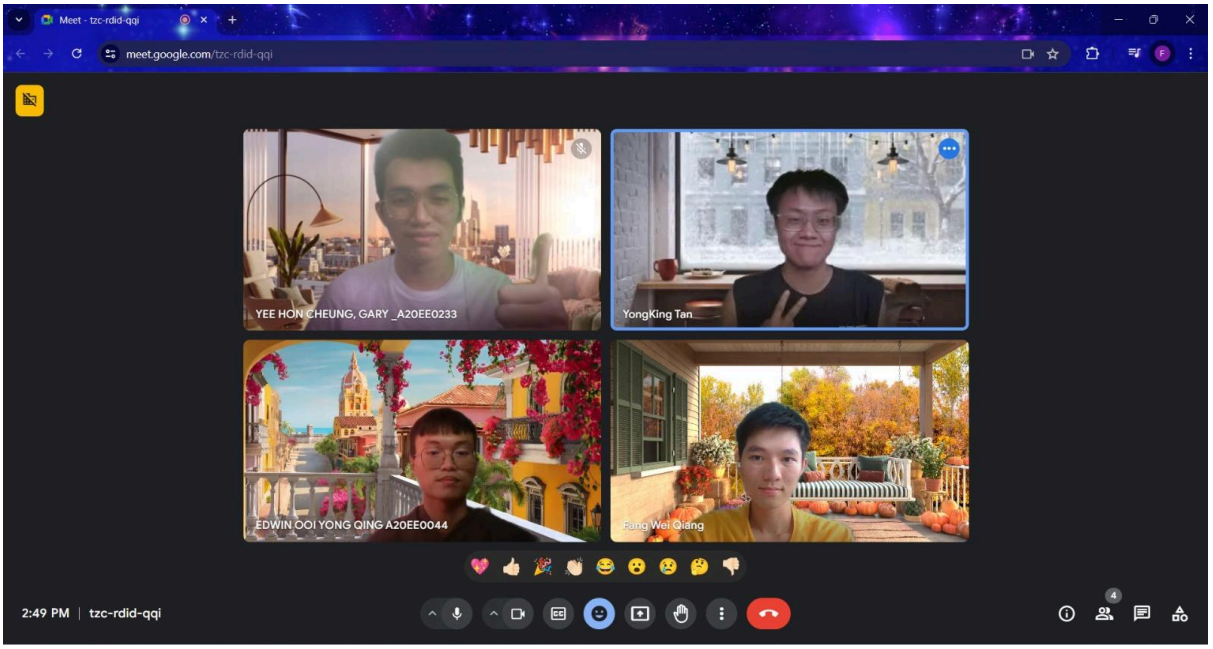
Date: 29/3/2024

Time: 11:00 am - 11:50 am

Location: Google Meet

Contents:

1. Discuss the details of the projects such as introduction, problem statement, objectives, project scope, methodology, system architecture and reference.
2. Distribution of tasks to each member of the team and set a deadline, 30/3/2024 for completing tasks.



Meeting Minutes 2

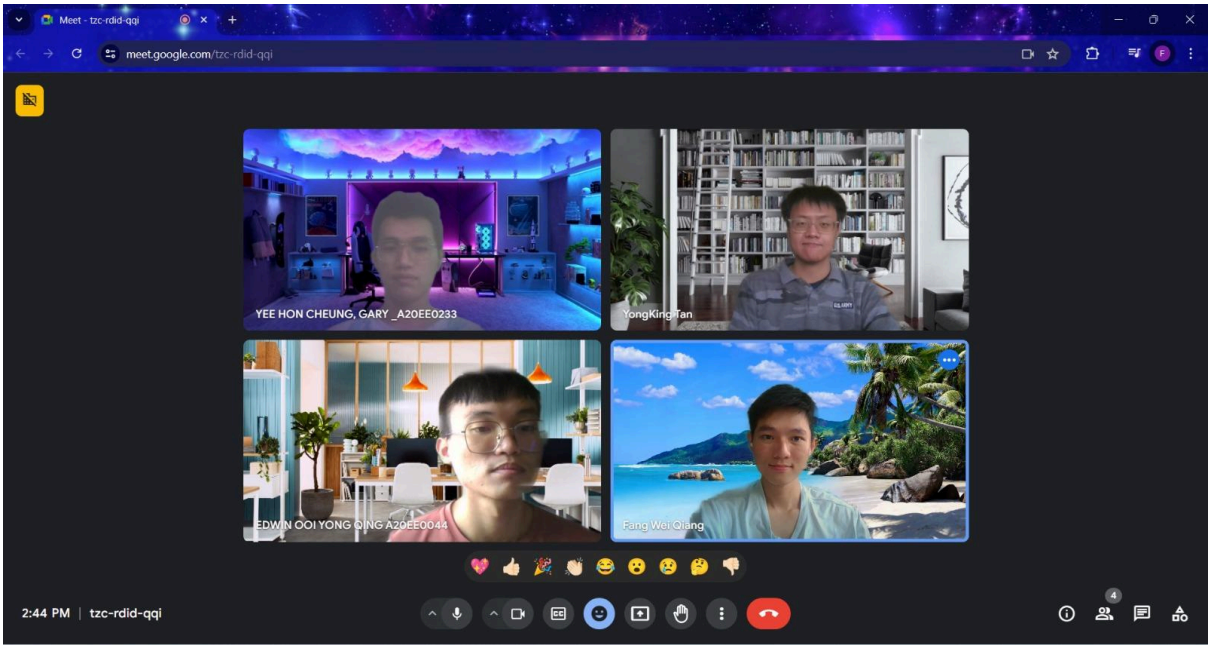
Date: 5/4/2024

Time: 11:00 am - 11:50 am

Location: Google Meet

Contents:

1. Discuss overall flow of the process and the overall sequence of the process.
2. Change value in codesys > data transfer to nodejs server > update data in database (SQL) > Web dashboard request data from database > Display result
3. Figure out the system architecture of the whole system and assign tasks to each member.



Meeting Minutes 3

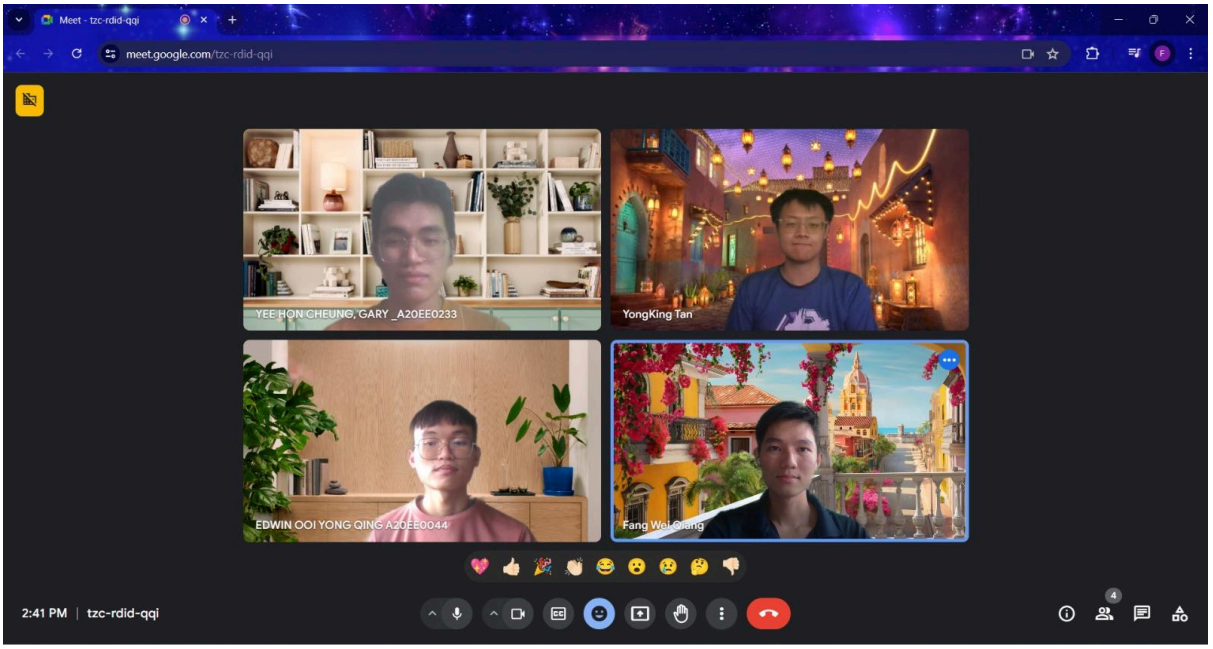
Date: 21/4/2024

Time: 11:00 am - 11:50 am

Location: Google Meet

Contents:

1. Update the progress of each task and discuss problems faced when completing the project.
2. Brainstorm and give suggestions to help other members solve the problem.
3. Set the deadline for the next stage prototype on 28/4/2024.



Meeting Minutes 4

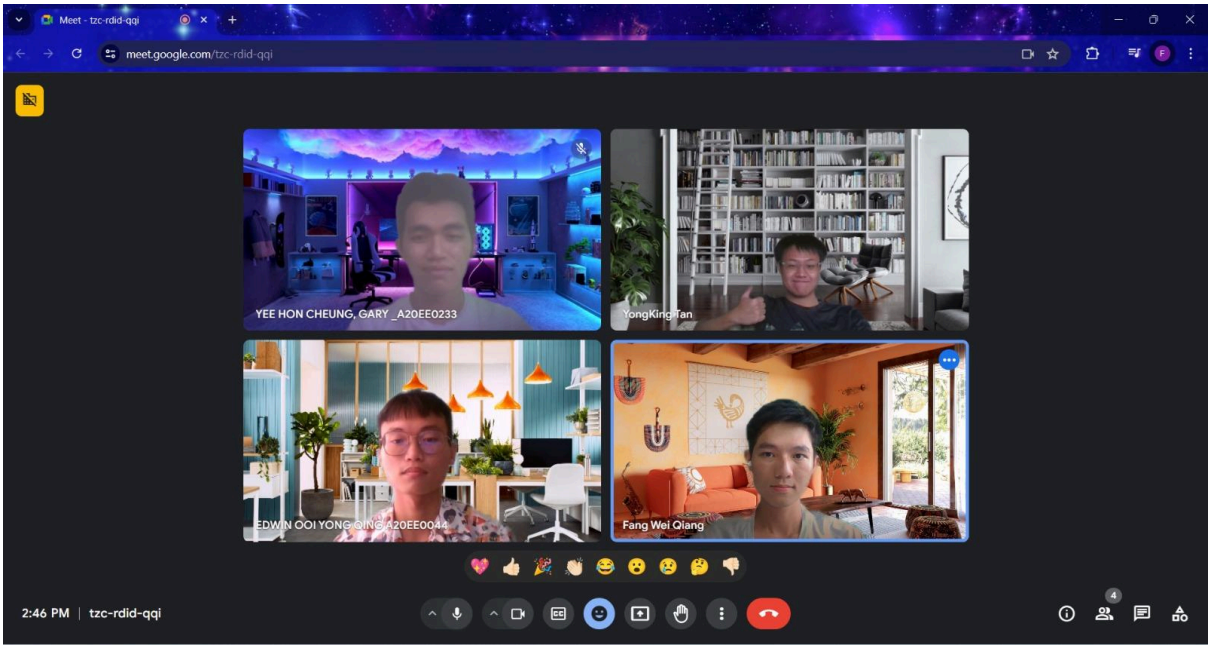
Date: 24/5/2024

Time: 11:00 am - 11:50 am

Location: Google Meet

Contents:

1. Update on progress of completing each part of system architecture.
2. Set a deadline for completing all components of the system so that it can proceed to system integration.



Meeting Minutes 5

Date: 23/6/2024

Time: 11:00 am - 11:50 am

Location: Google Meet

Contents:

1. Finalisation on slides.
2. Distribution of tasks on report writing

Appendix B

server.js:

```
import express from 'express'
import cors from 'cors'
import faker from 'faker';
import { getStatus, getTempData} from './database.js';

const app = express()
app.use(express.json())
app.use(cors({
   origin: 'http://localhost:5173' // Allow requests from this origin only
}));

let status = {
   light1: false,      //true: object detected
   light2: false,      //false: no object detected
   light3: true,
   light4: false,
   light5: false,
   medium1: false,
   medium2: false,
   medium3: false,
   medium4: false,
   medium5: false,
   heavy1: false,
   heavy2: true,
   heavy3: false,
   heavy4: false,
   heavy5: false,
   temp: 30,
   alarm: false,
   power: false,
}

app.use((err, req, res, next) => {
   console.error(err.stack)
   res.status(500).send('Something broke!')
})

app.get('/', (req, res) => {
   console.log('Getting request /')
   // res.send(status);
   setTimeout(() => {
       res.send(status)
   }, 2000)
});

app.get('/home', (req, res) => {
   console.log('Getting request /home')
   // res.send(status);
   setTimeout(() => {
       res.send(status)
   }, 2000)
});
```

```
app.get('/temperature', async (req, res) => {
   console.log('Getting request /temperature')
   // res.send(status);
   try {
       let tempd = await getTempData();
       tempd = tempd.reverse();
       const labels = tempd.map(item => item.date);
       const tempData = tempd.map(item => item.temp);
       const LineData = {
           labels: labels,
           datasets: [
               {
                   label: 'Temperature',
                   data: tempData,
               }
           ]
       }

       setTimeout(() => {
           res.send(LineData);
       }, 500);

   } catch (error) {
       res.status(500).send('Error retrieving status');
   }


});

app.get("/records", async (req, res) => {
   console.log('Getting request /records')
   try {
       let data = await getStatus();
       data = data.reverse();
       // Extract data for chart
       const labels = data.map(item => item.date);
       const lightData = data.map(item => item.light);
       const mediumData = data.map(item => item.medium);
       const heavyData = data.map(item => item.heavy);
       // Create chart configuration
       const barData = {
           labels: labels,
           datasets: [
               {
               label: 'Lightweight',
               data: lightData,
               backgroundColor: 'rgba(255, 206, 86, 0.5)'
               },
               {
               label: 'Medium-weight',
               data: mediumData,
               backgroundColor: 'rgba(75, 192, 192, 0.5)'
               },
               {
               label: 'Heavyweight',
               data: heavyData,
               backgroundColor: 'rgba(255, 99, 132, 0.5)'
```

```
                }
            ]
        }

        // Send HTML with embedded chart
        setTimeout(() => {
            res.send(barData);
        }, 500)

    } catch (error) {
    res.status(500).send('Error retrieving status');
    }
});


app.listen(8080, () => {
    console.log('Listening on port 8080')
})

// Turn off apache 2 server in terminal
// sudo systemctl stop apache2

// Turn on xampp in terminal
// sudo /opt/lampp/./manager-linux-x64.run

let server;
let isServerRunning = false;
const PORT = 12020;

// Start the server
function startServer() {
    if (!isServerRunning) {
        server = app.listen(PORT, () => {
            console.log(`Server is listening on port ${PORT}`);
            isServerRunning = true;
        });
    }
}

// Stop the server
function stopServer() {
    if (isServerRunning) {
        server.close((err) => {
            if (err) {
                console.error('Error while stopping the server:', err);
            } else {
                console.log('Server has stopped listening on port', PORT);
                isServerRunning = false;
            }
        });
    }
}

app.post('/home/start', (req, res) => {
    console.log('Getting request post /home/start')
    startServer();
    res.send('Server is starting...');
```

```
})

app.post('/home/stop', (req, res) => {
   console.log('Getting request post /home/stop')
   stopServer();
   res.send('Server is stopping...');
})
```

test.js:

```
import express from 'express';
import { getStatus, getTempData} from './database.js'; // Assuming you have defined the
appropriate functions in database.js

const app = express();

app.use(express.json());

app.get("/status", async (req, res) => {

 let status = await getStatus()
 setTimeout(() => {
   res.send(status)
 }, 500)
})

app.get("/tempd", async (req, res) => {

 let tempd = await getTempData()
 setTimeout(() => {
   res.send(tempd)
 }, 500)
})


app.get('/temp', async (req, res) => {
 try {
   const data = await getTempData();
   const reversedData = data.reverse();  // Reverse to show the latest data on the right

   // console.log('Data:', reversedData);  // Log the data for debugging

   const chartConfig = {
     type: 'line',
     data: {
       labels: reversedData.map(item => item.date),
       datasets: [{
         label: 'Temperature',
         data: reversedData.map(item => item.temp),
         borderColor: 'rgba(75, 192, 192, 1)',
         borderWidth: 2,
         fill: false
       }]
     },
     options: {
```

```
      scales: {
        x: {
               type: 'category',   //'category' instead of 'time' to avoid the automatic
generation of time intervals
            labels: reversedData.map(item => item.date), // Use the exact labels from your
data
          title: {
            display: true,
            text: 'Timestamp'
          }
        },
        y: {
          title: {
            display: true,
            text: 'Temperature (°C)'
          }
        }
      }
    }
  };

  res.send(`
    <!DOCTYPE html>
    <html>
    <head>
      <title>Temperature Chart</title>
      <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
      <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns"></script>
    </head>
    <body>
      <h1>Temperature Chart</h1>
      <canvas id="tempChart"></canvas>
      <script>
        document.addEventListener("DOMContentLoaded", function() {
          console.log('Chart.js is loaded:', Chart);
          const ctx = document.getElementById('tempChart').getContext('2d');
          new Chart(ctx, ${JSON.stringify(chartConfig)});
        });
      </script>
    </body>
    </html>
  `);
 } catch (error) {
  console.error('Error retrieving data from the database:', error);
  res.status(500).send('Error retrieving data from the database');
 }
});


app.get("/data", async (req, res) => {
  try {
      let status = await getStatus();
      res.send(status);
  } catch (error) {
      console.error("Error:", error);
      res.status(500).send("Internal Server Error");
  }
```

```
});

app.get("/chart", async (req, res) => {
   try {
       let data_ori = await getStatus();
       const data = data_ori.reverse();  // Reverse to show the latest data on the right
       // Extract data for chart
       const labels = data.map(item => item.date);
       const lightData = data.map(item => item.light);
       const mediumData = data.map(item => item.medium);
       const heavyData = data.map(item => item.heavy);
       // Create chart configuration
       const chartConfig = {
           type: 'bar',
           data: {
           labels: labels,
           datasets: [
               {
               label: 'Light',
               data: lightData,
               backgroundColor: 'rgba(255, 206, 86, 0.5)'
               },
               {
               label: 'Medium',
               data: mediumData,
               backgroundColor: 'rgba(75, 192, 192, 0.5)'
               },
               {
               label: 'Heavy',
               data: heavyData,
               backgroundColor: 'rgba(255, 99, 132, 0.5)'
               }
           ]
           },
           options: {
           scales: {
               x: {
               stacked: true
               },
               y: {
               stacked: true
               }
           }
           }
       };
       // Send HTML with embedded chart
       setTimeout(() => {
           res.send(`
               <!DOCTYPE html>
               <html>
               <head>
                   <title>Stacked Bar Chart</title>
                   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
               </head>
               <body>
                   <canvas id="barChart"></canvas>
                   <div style="width:95%">
```

```
                    <canvas id="stackedBarChart"></canvas>
                    </div>
                    <script>
                    var ctx = document.getElementById('stackedBarChart').getContext('2d');
                    new Chart(ctx, ${JSON.stringify(chartConfig)});
                    </script>
                </body>
                </html>
            `);
        }, 500)
    } catch (error) {
    res.status(500).send('Error retrieving status');
    }
});

app.listen(8080, () => {
console.log('Server is running on port 8080')
})
```

database.js:

```
import mysql from 'mysql2';
import dotenv from 'dotenv';

dotenv.config()

const db = mysql.createPool({
 host: "localhost",
 user: "root",
 password : "",
 database: "plc"
}).promise();


// latest 10 rows with different hours
export const getTempData = async () => {
 try {
   // Execute the SQL query to get the latest 50 records from the scada table
   // This is a safeguard to have enough data to filter out duplicates by hour
   const [rows] = await db.query(`
     SELECT Status, Date
     FROM scada
     ORDER BY Date DESC
     LIMIT 20
   `);

   // Map the rows to extract temp and date, and parse the JSON in Status
   const mappedRows = rows.map(row => {
     const status = JSON.parse(row.Status);
     return {
       temp: status.temp,
       date: new Date(row.Date)
     };
   });
```

```
    // Filter rows to only include one entry per hour
    const filteredRows = [];
    const seenHours = new Set();

    for (const row of mappedRows) {
      const hourString = row.date.toISOString().substring(0, 13); // Extract 'YYYY-MM-DDTHH'
      if (!seenHours.has(hourString)) {
        filteredRows.push(row);
        seenHours.add(hourString);
      }
      if (filteredRows.length >= 20) {
        break; // Stop when we have 10 unique hour entries
      }
    }

    // Return the filtered rows, converting dates back to strings if needed
    return filteredRows.map(row => ({
      temp: row.temp,
      date: formatDate(new Date(row.date.toISOString()))
    }));
  } catch (error) {
    // Handle any errors that occur during the database query or JSON parsing
    console.error('Error fetching temperature data:', error);
    throw error; // Optionally rethrow the error to be handled by the caller
  }
};

//latest 12 rows with different hours
export async function getStatus() {
  // Fetch more rows initially to ensure we have enough data to filter out duplicates by
hour
  const [rows] = await db.query("SELECT * FROM scada ORDER BY Date DESC LIMIT 50");

  let result = []; // Define an empty array to store results
  const seenHours = new Set(); // Set to keep track of seen hours

  // Loop through each row returned by the query
  for (const row of rows) {
    const jsonString = row.Status;
    const date = new Date(row.Date);

    // Generate a string representing the year, month, day, and hour of the date
    const hourString = date.toISOString().substring(0, 13); // 'YYYY-MM-DDTHH'

    // Only include the row if we haven't seen this hour before
    if (!seenHours.has(hourString)) {
      // Parse the JSON string to a JavaScript object
      const jsonObject = JSON.parse(jsonString);

       // Count true values for properties in the JavaScript object using countTrueValues()
function
      const counts = countTrueValues(jsonObject, date);

      // Push counts to the result array
      result.push(counts);

      // Add the hour to the seen hours set
```

```
      seenHours.add(hourString);
    }

    // Stop if we have 12 unique hour entries
    if (result.length >= 20) {
      break;
    }
  }

  return result;
}

function countTrueValues(data, date) {
  let counts = {
    light: 0,
    medium: 0,
    heavy: 0,
    date: formatDate(date) // Initialize the date property with the provided date
  };

  for (let key in data) {
    if (data.hasOwnProperty(key)) {
      if (key.startsWith('light') && data[key]) {
        counts.light++;
      } else if (key.startsWith('medium') && data[key]) {
        counts.medium++;
      } else if (key.startsWith('heavy') && data[key]) {
        counts.heavy++;
      }
    }
  }

  return counts;
}

// Function to format date as "YYYY-MM-DD HH:mm:ss"
function formatDate(date) {
  // Add 8 hours to the date
  date.setHours(date.getHours() + 8);

  const formattedDate = date.toISOString().replace(/T/, ' ').replace(/\..+/, '');
  return formattedDate;
}
```

Home.jsx:

```
import React from 'react';
import axios from 'axios'
import { useEffect, useState } from 'react'
import {DashboardLayout} from '../components/Layout';
import OneShelf from '../components/oneshelf'
import StartStopBtn from '../components/startstopbtn'
import AlarmPowerStatus from '../components/alarmpowerstatus'
import TemperatureMonitor from '../components/temperaturemonitor'
import PageTitle from '../components/pagetitle';


const HomePage = () => {
```

```
const [Status, setStatus] = useState([]);
const [LightStatus, setLightStatus] = useState([]);
const [MediumStatus, setMediumStatus] = useState([]);
const [HeavyStatus, setHeavyStatus] = useState([]);
const [AlarmStatus, setAlarmStatus] = useState(false);
const [PowerStatus, setPowerStatus] = useState(true);
const [Temperature, setTemperature] = useState(0);
const [error,setError] = useState(false)
const [loading,setLoading] = useState(false)
const host = 'localhost:8080'
const ShelfName = [
  {
    name: 'Lightweight',
    status: LightStatus
  },
  {
    name: 'Medium Weight',
    status: MediumStatus
  },{
    name: 'Heavyweight',
    status: HeavyStatus
  },
];

useEffect(() => {
  const controller = new AbortController()
  ;(async () => {
    setError(false)
    if (!Status) {
      setLoading(true)
    }

    try {
      const result = await axios.get('http://'+ host +'/', {
        signal: controller.signal
      })
      console.log(result.data)
      setStatus(result.data)
      parseData(result.data);

    } catch (error) {
      if (axios.isCancel(error)) {
        console.log('Request canceled', error.message)
        return
      }
      setError(true)
      console.log(error)
    }
    setLoading(false)
  })()

  return () => {
    // cleanup
    controller.abort()
  }
```

```
}, [Status])

const parseData = (data) => {
  let L = []
  let M = []
  let H = []
  let A = false
  let P = false
  let T = 0
  let index = 0

  for (let key in data) {
    if (key.includes('light')) {
      index = key.match(/\d/) - 1
      L[index] = data[key]
    }
    if (key.includes('medium')) {
      index = key.match(/\d/) - 1
      M[index] = data[key]
    }
    if (key.includes('heavy')) {
      index = key.match(/\d/) - 1
      H[index] = data[key]
    }
    if (key.includes('alarm')) {
      A = data[key]
    }
    if (key.includes('power')) {
      P = data[key]
    }
    if (key.includes('temp')) {
      T = data[key]
    }
  }

  setLightStatus(L)
  setMediumStatus(M)
  setHeavyStatus(H)
  setAlarmStatus(A)
  // setPowerStatus(P)
  setTemperature(T)

  // console.log(LightStatus)
  // console.log(MediumStatus)
  // console.log(HeavyStatus)
  // console.log(AlarmStatus)
  // console.log(PowerStatus)
  // console.log(Temperature)

}

const handleStart = async () => {
  setPowerStatus(true);
  try {
    const result = await axios.post('http://localhost:8080/home/start', {});
    console.log('Response from server:', result.data);
  } catch (error) {
```

```
    console.error('Error sending POST request:', error);
  }
}

const handleStop = async () => {
  setPowerStatus(false)
  try {
    const result = await axios.post('http://localhost:8080/home/stop', {});
    console.log('Response from server:', result.data);
  } catch (error) {
    console.error('Error sending POST request:', error);
  }
}

return (
  <DashboardLayout>

    <PageTitle title="Home Page" />

    {loading && <p>Loading...</p>}


    <div className="flex grid grid-cols-2 h-full bg-gray-400 mt-8">
      {error
      ?
        <p>Something went wrong</p>
      :
        <div className='flex grid grid-rows-3 content-evenly'>
          {
            ShelfName.map((shelf, index) => {
              return <OneShelf key={index} name={shelf.name} status={shelf.status} />
            })
          }
        </div>
      }

      <div className='flex grid grid-rows-2 bg-gray-400'>
        <div className='flex justify-center'>
          <StartStopBtn onStart={handleStart} onStop={handleStop} />
        </div>

        <div className='flex row-span-2 grid grid-cols-2'>
          <div>
            <TemperatureMonitor temperature={Temperature} />
          </div>

          <div>
            <AlarmPowerStatus alarmStatus={AlarmStatus} powerStatus={PowerStatus} />
          </div>
        </div>

      </div>
    </div>
```

```
    </DashboardLayout>
  )
}

export default HomePage;
```

Records.jsx:

```
import React from 'react';
import { useEffect, useState } from 'react'
import Chart from 'chart.js/auto';
import {DashboardLayout} from '../components/Layout';
import axios from 'axios';
import RecordsChart from '../components/RecordsChart';
import PageTitle from '../components/pagetitle';


const RecordsPage = () => {

 const result = {};
 const [error, setError] = useState(false)
 const [loading, setLoading] = useState(false)
 const [labels, setLabels] = useState([])
 const [datasets, setDatasets] = useState([])
 const host = 'localhost:8080'

 useEffect(() => {
   const controller = new AbortController()
   ;(async () => {
     setError(false)
     setLoading(true)
     try {
       const result = await axios.get('http://'+ host +'/records', {
         signal: controller.signal
       })
       console.log(result);
       setLabels(result.data.labels);
       setDatasets(result.data.datasets);
     } catch (error) {
       if (axios.isCancel(error)) {
         console.log('Request canceled', error.message)
         return
       }
       setError(true)
       console.log(error)
     }
     setLoading(false)
   })()

   return () => {
     // cleanup
     controller.abort()
   }

 }, [])
```

```
 return (
   <DashboardLayout>
       <PageTitle title="Records" />

       {loading && <p>Loading...</p>}

       <div className='flex'>
         <RecordsChart labels={labels} datasets={datasets}/>
       </div>

   </DashboardLayout>
 )
}

export default RecordsPage;
```

Temperature.jsx:

```
import React from 'react';
import TemperatureChart from '../components/TemperatureChart';
import {DashboardLayout} from '../components/Layout';
import { useEffect, useState } from 'react'
import axios from 'axios';
import PageTitle from '../components/pagetitle';


const TemperaturePage = () => {

 const result = {};
 const [error, setError] = useState(false)
 const [loading, setLoading] = useState(false)
 const [labels, setLabels] = useState([])
 const [datasets, setDatasets] = useState([])
 const host = 'localhost:8080'

 useEffect(() => {
   const controller = new AbortController()
   ;(async () => {
     setError(false)
     setLoading(true)
     try {
       const result = await axios.get('http://'+ host +'/temperature', {
         signal: controller.signal
       })
       console.log(result.data);
       setLabels(result.data.labels);
       setDatasets(result.data.datasets);
       // console.log(context);
     } catch (error) {
       if (axios.isCancel(error)) {
         console.log('Request canceled', error.message)
         return
       }
       setError(true)
       console.log(error)
     }
```

```
      setLoading(false)
    })()

    return () => {
      // cleanup
      controller.abort()
    }

  }, [])

  return (
    <DashboardLayout>
      <PageTitle title="Temperature Monitoring" />

      <div>
        <TemperatureChart labels={labels} datasets={datasets} />
      </div>

      {/* <div>
        <ChartComponent />
      </div> */}


    </DashboardLayout>
  )
}

export default TemperaturePage;
```

Camera.jsx:

```
import React from 'react';
import Webcam from "react-webcam";
import { useCallback, useRef, useState } from "react";
import {DashboardLayout} from '../components/Layout';
import PageTitle from '../components/pagetitle';


const CameraPage = () => {

  const [img, setImg] = useState(null);
  const webcamRef = useRef(null);


  const videoConstraints = {
    width: { min: 800 },
    height: { min: 500 },
    facingMode: "user",
  };

  const capture = useCallback(() => {
    const imageSrc = webcamRef.current.getScreenshot();
    setImg(imageSrc);
  }, [webcamRef]);

  return (
    <DashboardLayout>
```

```
      <PageTitle title="Camera" />

      <div className="Container">
        {img === null
          ? (
            <>
            <div className="flex justify-center">
              <Webcam
              audio={false}
              mirrored={true}
              width={1500}
              height={800}
              ref={webcamRef}
              screenshotFormat="image/jpeg"
              videoConstraints={videoConstraints} />
            </div>

            <div>
               <button className='bg-teal-400 rounded-full text-2xl p-4 mt-4 w-1/6 font-bold
border-4 border-black' onClick={capture}>Capture photo</button>
            </div>
            </>
            )
          : (
            <div className=''>
              <div className="flex justify-center">
                <img src={img} alt="screenshot" />
              </div>

              <div>
                       <button  className='bg-teal-400  rounded-full  text-2xl  p-4  mt-4  w-1/6
font-bold border-4 border-black' onClick={() => setImg(null)}>Retake</button>
              </div>
            </div>
          )
        }
      </div>
    </DashboardLayout>
  )
}

export default CameraPage;
```

About.jsx:

```
import React from 'react';
import MemberInfo from '../components/memberInfo';
import {DashboardLayout} from '../components/Layout';
import PageTitle from '../components/pagetitle';
import Edwin from '../images/Edwin.jpeg';
import WeiQiang from '../images/WeiQiang.jpeg';
import Gary from '../images/Gary.jpg';
import YongKing from '../images/YongKing.jpeg';

const AboutPage = () => {
  const memberData = [
    {
```

```
    name: 'Edwin Ooi Yong Qing',
    matric: 'A20EE0044',
    pic: Edwin,
  },
  {
    name: 'Fang Wei Qiang',
    matric: 'A20EE0049',
    pic: WeiQiang,
  },
  {
    name: 'Yee Hon Cheung, Gary',
    matric: 'A20EE0233',
    pic: Gary,
  },
  {
    name: 'Tan Yong King',
    matric: 'A20EE0278',
    pic: YongKing,
  }
]
return (
  <DashboardLayout>
    <div className="mb-8">
      <PageTitle title="About the Team"/>
    </div>

    <div className="flex grid grid-cols-4">
      {memberData.map((obj, index) => {
        return <MemberInfo key={index} name={obj.name} matric={obj.matric} pic={obj.pic}/>
      })}
    </div>


  </DashboardLayout>
)
}

export default AboutPage;
```

Alarmpowerstatus.jsx:

```
export default function AlarmPowerStatus({ alarmStatus, powerStatus }) {
  return (
    <div className="flex justify-center grid grid-rows-2 gap-4 content-evenly">
         <div className={`text-sm font-semibold bg-${alarmStatus ? 'red' : 'green'}-500
text-white px-20 py-16 text-3xl rounded`}>
        {alarmStatus ? 'ON' : 'OFF'} Alarm Status
      </div>
         <div className={`text-sm font-semibold bg-${powerStatus ? 'green' : 'red'}-500
text-white px-20 py-16 text-3xl rounded`}>
        {powerStatus ? 'ON' : 'OFF'} Power Status
      </div>
    </div>
  );
};
```

Bodywrapper.jsx:

```
import React from 'react';

const BodyWrapper = ({children}) => {
 return (
     <div className="relative max-h-screen">
       <main className="w-full max-h-screen">{children}</main>
     </div>
 );
};

export default BodyWrapper;
```

Layout.jsx:

```
import React from "react";

import { NavSidebar } from "./NavSidebar";
import BodyWrapper from "./BodyWrapper";

export const DashboardLayout = ({ children }) => {
 return (
   <BodyWrapper>
     <div className="flex h-screen bg-gray-400">
       <NavSidebar />

       <div className="flex flex-col flex-1 max-h-screen">
         <main className="content">
           <section className="lg:flex-row flex flex-col flex-1">
             <div
               className="content-box"
               style={{ flexGrow: 1, flexBasis: "0%" }}
             >
               {children}
             </div>
           </section>
         </main>
       </div>
     </div>
   </BodyWrapper>
 );
};
```

memberInfo.jsx:

```
export default function MemberInfo( {name, matric, pic} ) {
   return (
         <div className="flex border-8 border-teal-400 p-4 bg-teal-200 justify-center grid
grid-rows-10 items-center">
           <img src={pic} alt={name} className="block row-span-8"
           style={{ width: '200px', aspectRatio: '9/12'}}
           />
           <h2 className="font-semibold mb-2 text-2xl ">{name}</h2>
           <p className="text-2xl">{matric}</p>
       </div>
   )
}
```

NavSideBar.jsx:

```
/* eslint-disable react/display-name, jsx-a11y/click-events-have-key-events */
import { Navigation } from "react-minimal-side-navigation";
import { useNavigate, useLocation } from "react-router-dom";
import Icon from "awesome-react-icons";
import React, { useState } from "react";

import "react-minimal-side-navigation/lib/ReactMinimalSideNavigation.css";

export const NavSidebar = () => {
 const navigate = useNavigate();
 const location = useLocation();
 const [isSidebarOpen, setIsSidebarOpen] = useState(true);

 return (
   <React.Fragment>
     {/* Sidebar Overlay */}
     <div
       onClick={() => {
         setIsSidebarOpen(false)
         console.log('Sidebar Overlay clicked')
         console.log(isSidebarOpen)
       }}
       className={`fixed inset-0 z-20 block transition-opacity bg-black opacity-50 ${
         isSidebarOpen ? "block" : "hidden"
       }`}
     />
{/*
     <div className="absolute right-0">
       <a href="https://github.com/abhijithvijayan/react-minimal-side-navigation">
         View on GitHub
       </a>
     </div> */}

     <div>
       <button
         className="btn-menu"
         onClick={() => {
           setIsSidebarOpen(true)
           console.log('Menu button clicked')
           console.log(isSidebarOpen)
         }}
         type="button"
       >
         <Icon name="burger" className="w-6 h-6" />
       </button>
     </div>

     {/* Sidebar */}
     <div
       className={`fixed inset-y-0 left-0 z-30 w-64 overflow-y-auto transition duration-300
ease-out transform translate-x-0 bg-white border-r-2 ${
         isSidebarOpen ? "ease-out translate-x-0" : "ease-in -translate-x-full"
       }`}
```

```
    >
      <div className="flex items-center justify-center mt-10 text-center py-6">
        <span className="mx-2 text-2xl font-semibold text-black">
          Warehouse Management App
        </span>
      </div>

      {/* https://github.com/abhijithvijayan/react-minimal-side-navigation */}
      <Navigation
        activeItemId={location.pathname}
        onSelect={({ itemId }) => {
          navigate(itemId);
        }}
        items={[
          {
            title: "Home",
            itemId: "/home",
            // Optional
            elemBefore: () => <Icon name="coffee" />
          },
          {
            title: "Camera",
            itemId: "/camera",
            elemBefore: () => <Icon name="user" />
          },
          {
            title: "Records",
            itemId: "/records",
            elemBefore: () => <Icon name="activity" />
          },
          {
            title: "Temperature",
            itemId: "/temperature",
            elemBefore: () => <Icon name="activity" />
          }
        ]}
      />

      <div className="absolute bottom-0 w-full my-8">
        <Navigation
          activeItemId={location.pathname}
          items={[
            {
              title: "About",
              itemId: "/about",
              elemBefore: () => <Icon name="calendar" />
            }
          ]}
          onSelect={({ itemId }) => {
            navigate(itemId);
          }}
        />
      </div>
    </div>
  </React.Fragment>
);
};
```

Oneself.jsx:

```
export default function OneShelf({ name, status }) {
   return (
       <div className="border-8 border-teal-400 p-4 bg-teal-200 m-8">
           <h2 className="font-semibold mb-2 text-3xl ">{name}</h2>
           <div className="grid grid-cols-5 ">
               {status.map((item, index) => (
                   <div className="border-2 border-gray-700 p- w-full ">
                       <div
                       key={index}
                           className={`flex h-20 justify-center ${ !item ? 'bg-green-500' :
'bg-red-500'}`}
                       ></div>
                   </div>
               ))}
           </div>
       </div>
   )
}
```

Pagetitle.jsx:

```
export default function PageTitle({ title }) {
   return (
       <div className="m-4">
                    <h1  className="border-4  border-black  p-2  bg-red-400  inline  text-3xl
font-bold">{title}</h1>
       </div>
   )
}
```

RecordsChart.jsx:

```
import React from 'react';
import {
 Chart as ChartJS,
 CategoryScale,
 LinearScale,
 BarElement,
 Title,
 Tooltip,
 Legend,
} from 'chart.js';
import { Bar } from 'react-chartjs-2';


ChartJS.register(
 CategoryScale,
 LinearScale,
 BarElement,
 Title,
 Tooltip,
 Legend
);

export const options = {
```

```
 plugins: {
   title: {
     display: true,
     text: 'Number of Available Spaces for each Category',
   },
 },
 responsive: true,
 scales: {
   x: {
     stacked: true,
   },
   y: {
     stacked: true,
   },
 },
};

export default function RecordsChart({labels, datasets}) {
 const recordsData ={
   labels,
   datasets
 }
 return <Bar options={options} data={recordsData} />;
};
```

Startstopbtn.jsx:

```
export default function StartStopBtn({ onStart, onStop }) {
   return (
     <div className="flex justify-center mt-12 mb-12">
             <button  className="bg-teal-500  text-white  text-4xl  px-20  rounded-lg  mr-8"
onClick={onStart}>START</button>
               <button  className="bg-teal-500  text-white  text-4xl  px-20  rounded-lg"
onClick={onStop}>STOP</button>
     </div>
   );
 };
```

Temperaturechart.jsx:

```
import React from 'react';
import {
 Chart as ChartJS,
 CategoryScale,
 LinearScale,
 PointElement,
 LineElement,
 Title,
 Tooltip,
 Legend,
} from 'chart.js';
import { Line } from 'react-chartjs-2';
import faker from 'faker';

ChartJS.register(
 CategoryScale,
 LinearScale,
 PointElement,
```

```
 LineElement,
 Title,
 Tooltip,
 Legend
);

export const options = {
 responsive: true,
 plugins: {
   legend: {
     position: 'top',
   },
   title: {
     display: true,
     text: 'Temperature of Warehouse over last 24 hours',
   },
 },
};

export default function TemperatureChart({labels, datasets}) {
 const tempData = {
   labels,
   datasets,
 }
 return <Line options={options} data={tempData} />;
}
```

TemperatureMonitor.jsx:

```
export default function TemperatureMonitor({ temperature }) {
   return (
     <div className="flex flex-col mt-12">
       <div className="border-4 rounded-full p-16 mb-8 ml-10 mr-10 border-gray-700">
         <div className="text-4xl font-bold">{temperature}°C</div>
       </div>
       <div className="text-3xl font-bold">Temperature</div>
     </div>
   );
 };
```