ThermoFisher SCIENTIFIC

REFERENCE NUMBER:	
DATE:	**01 July 2024**
REVISION:	
PAGE:	**1 of 20**

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

Contents

Architecture 2

Overall Architecture **2**

POC Architecture **3**

Objective 3

Software 3

Frontend **3**

Backend **4**

Lab Setup 5

IP addresses for each control unit 6

System startup 6

Results 10

Code Snippets 15

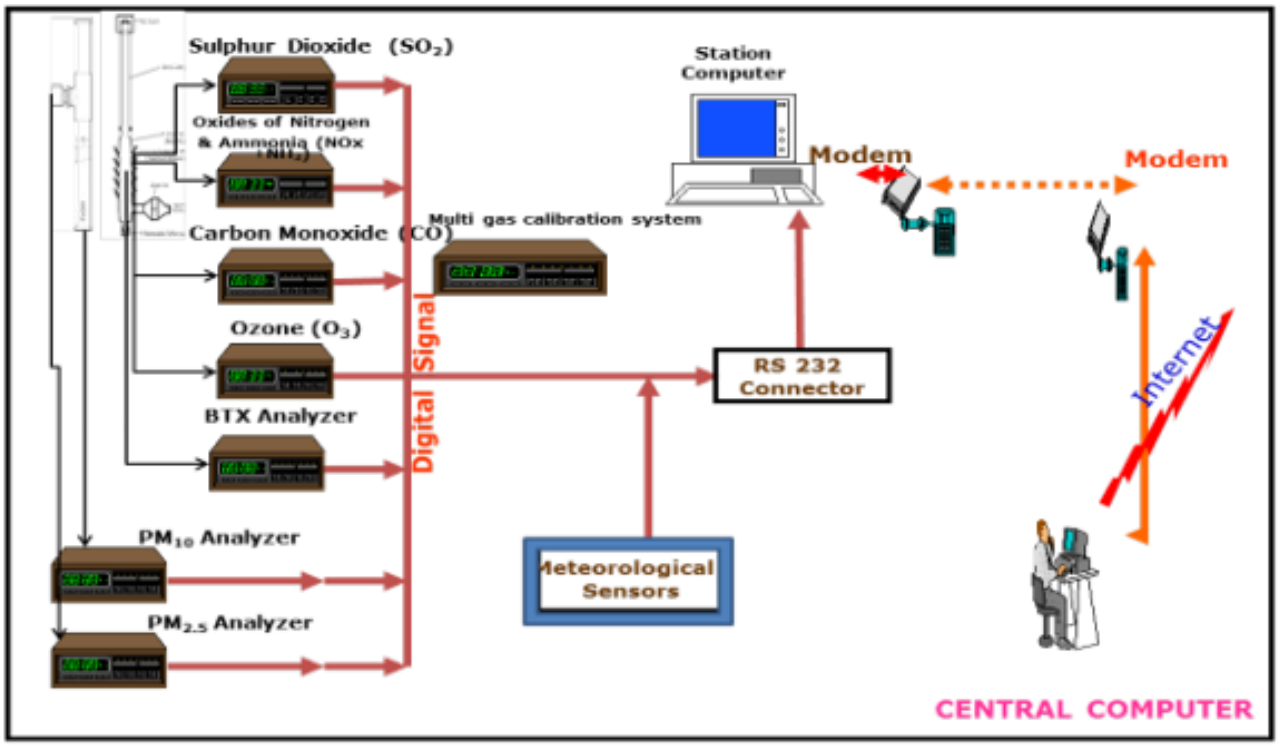
JavaScript to React 20

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

Architecture

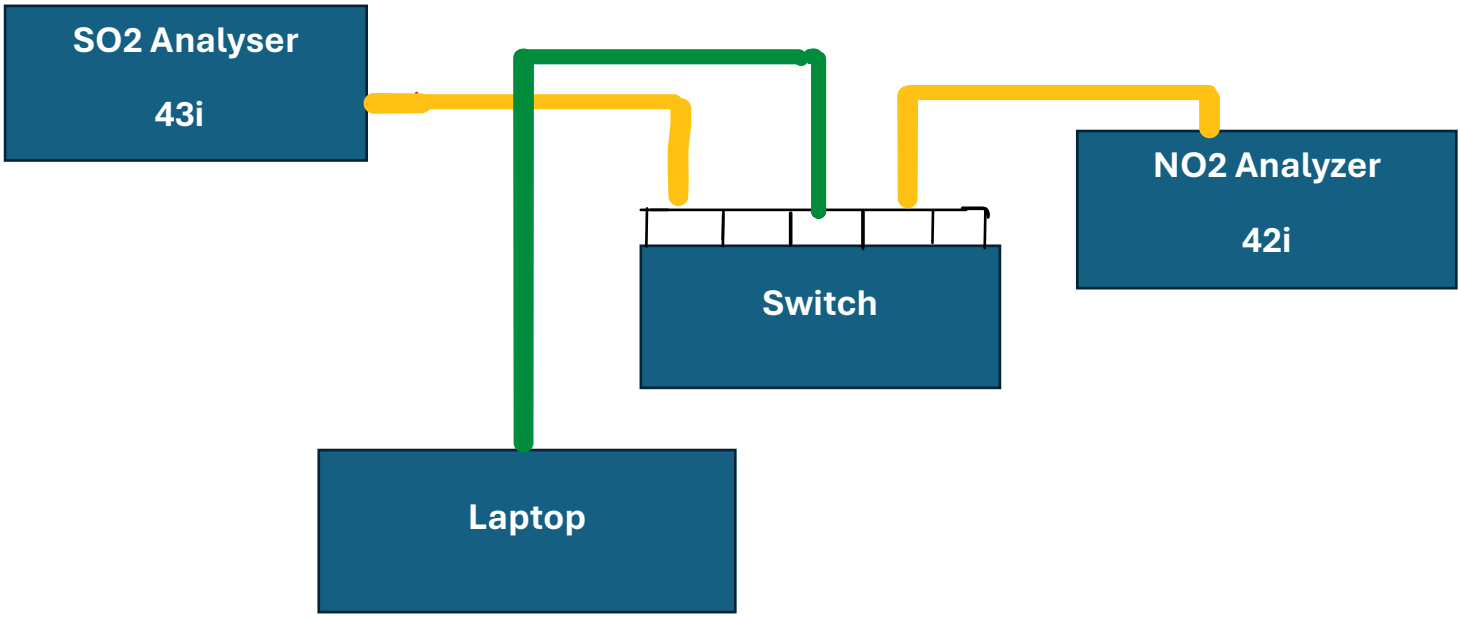
Overall Architecture



INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

POC Architecture



Objective: **Establish communication with the analyzers using Modbus protocol and develop software to communicate with the DAS.**

Software

Frontend

S.No	Technology	Version	Purpose
1.	HTML	HTML5	Blueprint of webpage

2.	CSS	CSS3	Styling the webpage
3.	JavaScript	ES6	Dynamic functionality

Backend

S.no	Technology	Version	Purpose
1.	Python	3.9.11	Establishing a connection to the instrument using Python
2.	Axios	0.21.4	Integrating Frontend & Backend
3.	Fast API	0.0.4	Framework to perform rest calls in Python
4.	Uvicorn server	0.29.0	To handle incoming HTTP requests and manage communication between clients & server

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

Lab Setup

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

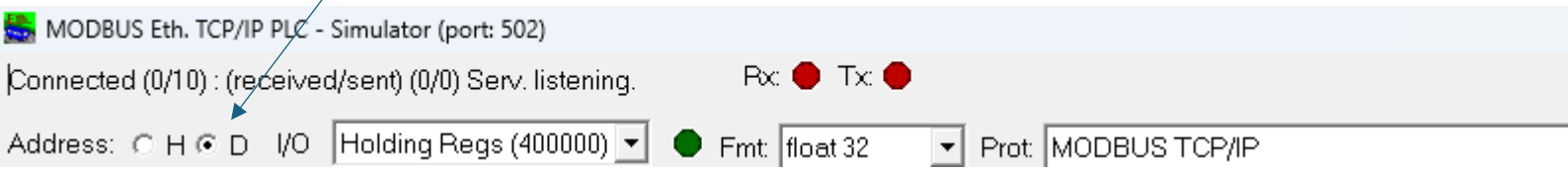
IP addresses for each control unit

Control Unit	IP address	Subnet mask
Laptop	192.168.0.200	255.255.254.0
SO2 Analyzer-43i	192.168.0.201	255.255.255.0
NO2 Analyzer-42i	192.168.0.202	255.255.254.0

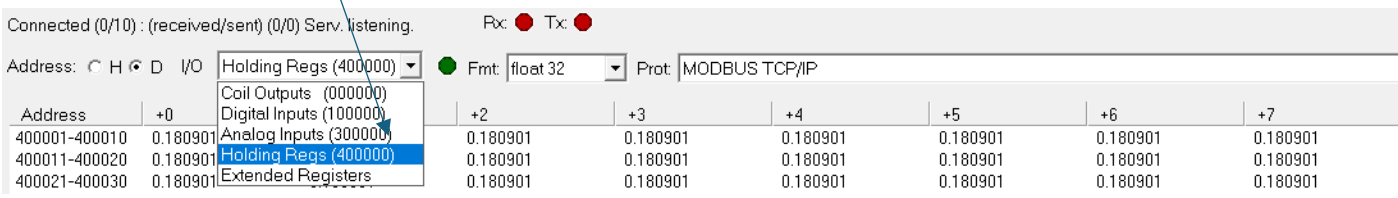
System startup

Modbus protocol setup using ModRSim2 Stimulator.

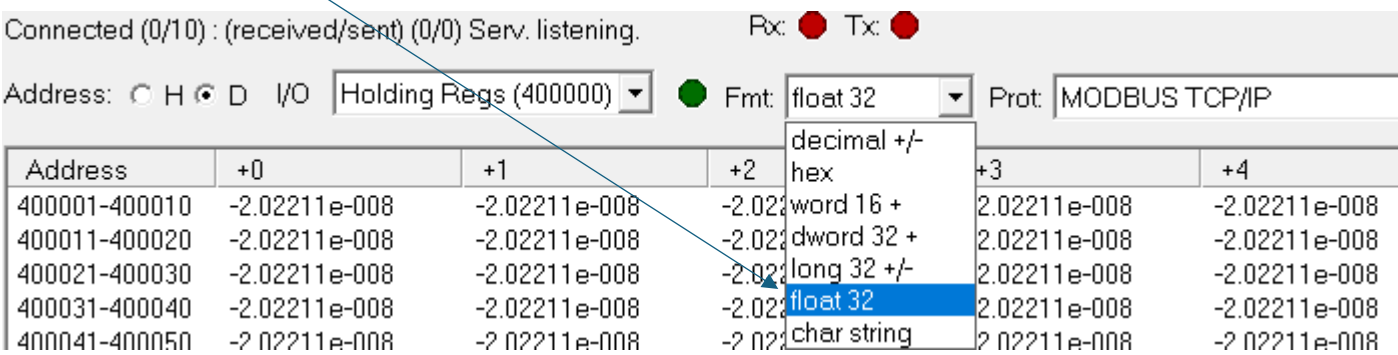
1. Set the register address to D (Decimal) format



2. Set I/O to the Holding registers in the Dropdown



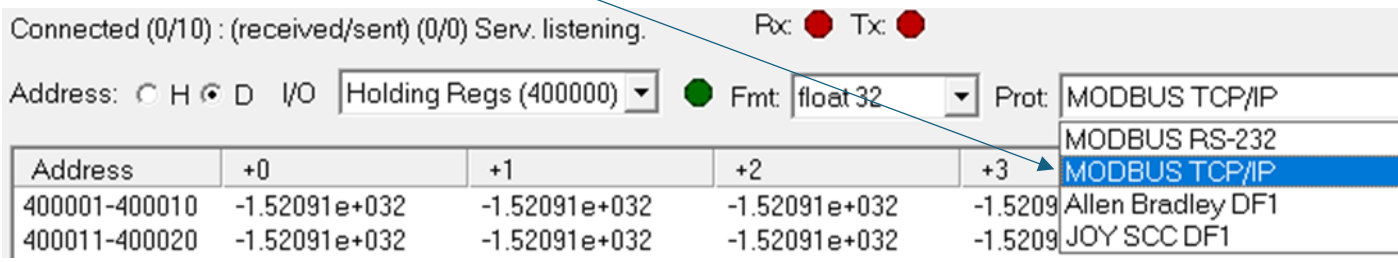
3. Set the format of data to float from the dropdown



INDIA ENGINEERING CENTER

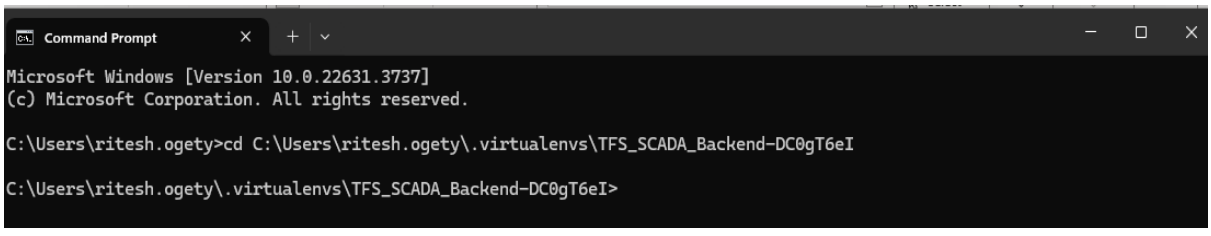
TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

4. Now set the prototype as Modbus TCP/IP for communication of registers.



Run the following commands in the command prompt

1. Set the virtual environment path in your laptop where it is installed e.g.., **C:\Users\ritesh.ogety\.virtualenvs\TFS_SCADA_Backend-DC0gT6eI**



2. Now enter the Scripts/activate command in the context of the virtual environment in Python for the above-created virtual environment.

```
C:\Users\ritesh.ogety\.virtualenvs\TFS_SCADA_Backend-DC0gT6eI>Scripts\activate

C:\Users\ritesh.ogety\.virtualenvs\TFS_SCADA_Backend-DC0gT6eI>()

(TFS_SCADA_Backend) C:\Users\ritesh.ogety\.virtualenvs\TFS_SCADA_Backend-DC0gT6eI>
```

3. Now change the path where your project is located say e.g..,

C:\Users\ritesh.ogety\OneDrive - Thermo Fisher Scientific\Desktop\DAS PROJECT

```
(TFS_SCADA_Backend) C:\Users\ritesh.ogety\.virtualenvs\TFS_SCADA_Backend-DC0gT6eI>cd C:\Users\ritesh.ogety\OneDrive - Th
ermo Fisher Scientific\Desktop\DAS PROJECT

(TFS_SCADA_Backend) C:\Users\ritesh.ogety\OneDrive - Thermo Fisher Scientific\Desktop\DAS PROJECT>
```

4. Now run the uvicorn command to start the server

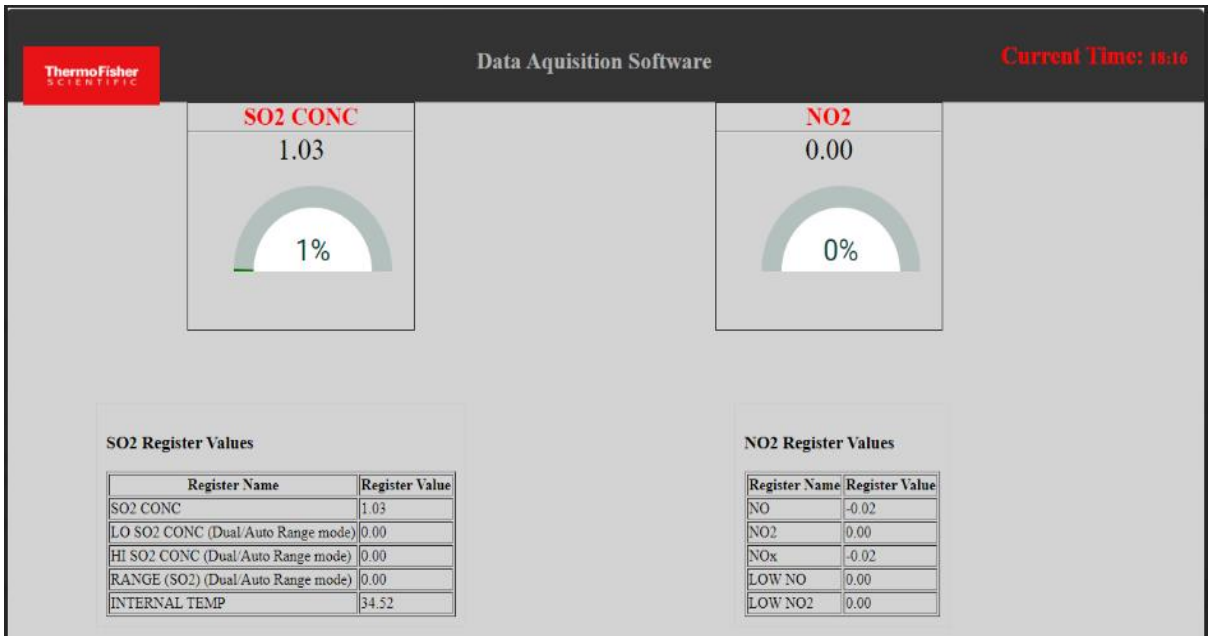
```
(TFS_SCADA_Backend) C:\Users\ritesh.ogety\OneDrive - Thermo Fisher Scientific\Desktop\DAS PROJECT>uvicorn index:app --ho
st 127.0.0.1 --port 8000
INFO:     Started server process [18788]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Note: Now the connection to the server is successful as you can see in the above screenshot as " INFO: Started Server process."

Then load your respective webpage and then you can see your real-time data display of the instrument on the software.

Results

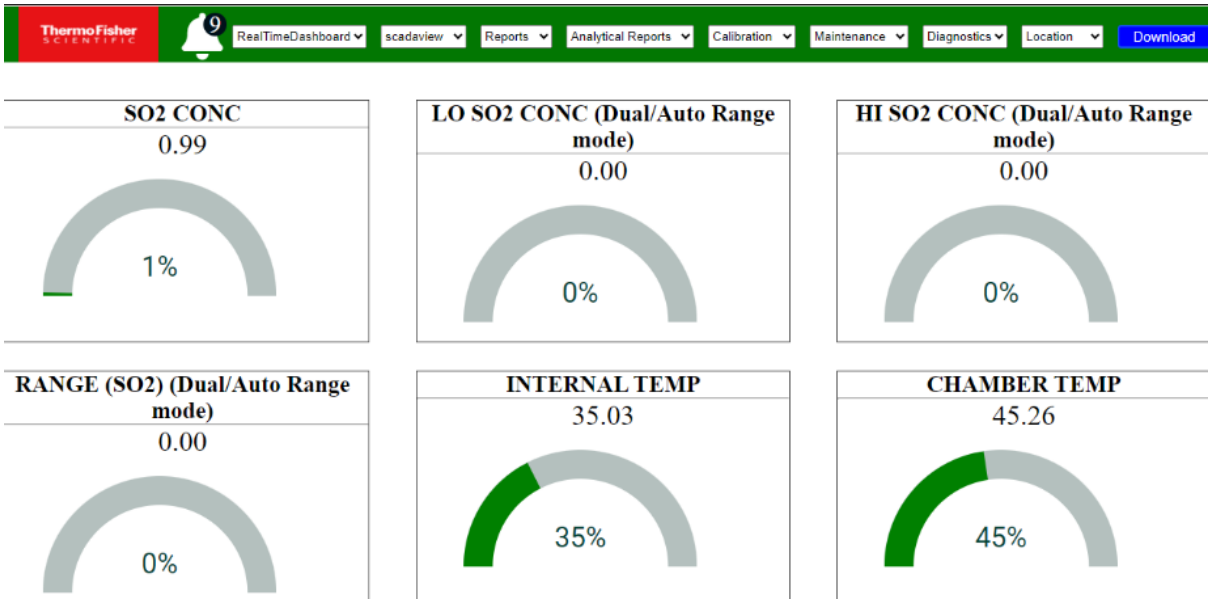
Dashboard



INDIA ENGINEERING CENTER

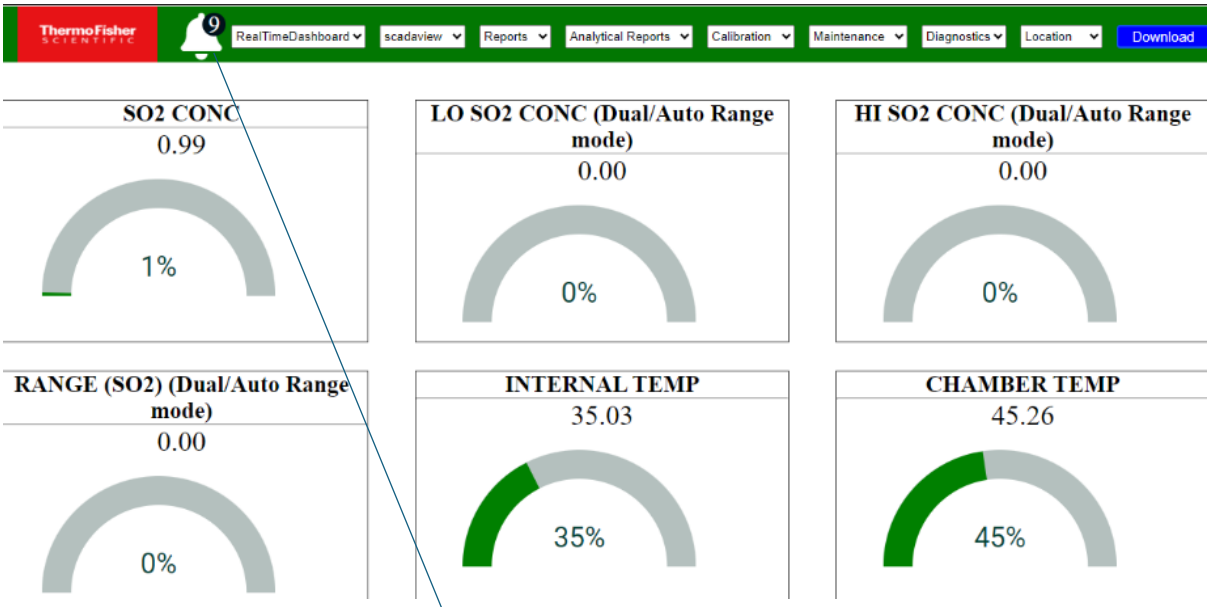
TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

SO2 home page



INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**



Popup Window - Work - Microsoft Edge

about:blank

User Alarms

Register Name	Register Value	Time Stamp	Register Threshold
LO SO2 CONC (Dual/Auto Range mode)	0	14/6/2024 3:31:33 pm	-1
HI SO2 CONC (Dual/Auto Range mode)	0	14/6/2024 3:31:33 pm	-1
RANGE (SO2) (Dual/Auto Range mode)	0	14/6/2024 3:31:33 pm	-1
INTERNAL TEMP	35.41303253173828	14/6/2024 3:31:33 pm	34
CHAMBER TEMP	45.15971374511719	14/6/2024 3:31:33 pm	20
FLASH REF	90	14/6/2024 3:31:33 pm	6
ANALOG IN 8 (I/O Expansion Board installed)	0.005239999853074551	14/6/2024 3:31:33 pm	-750

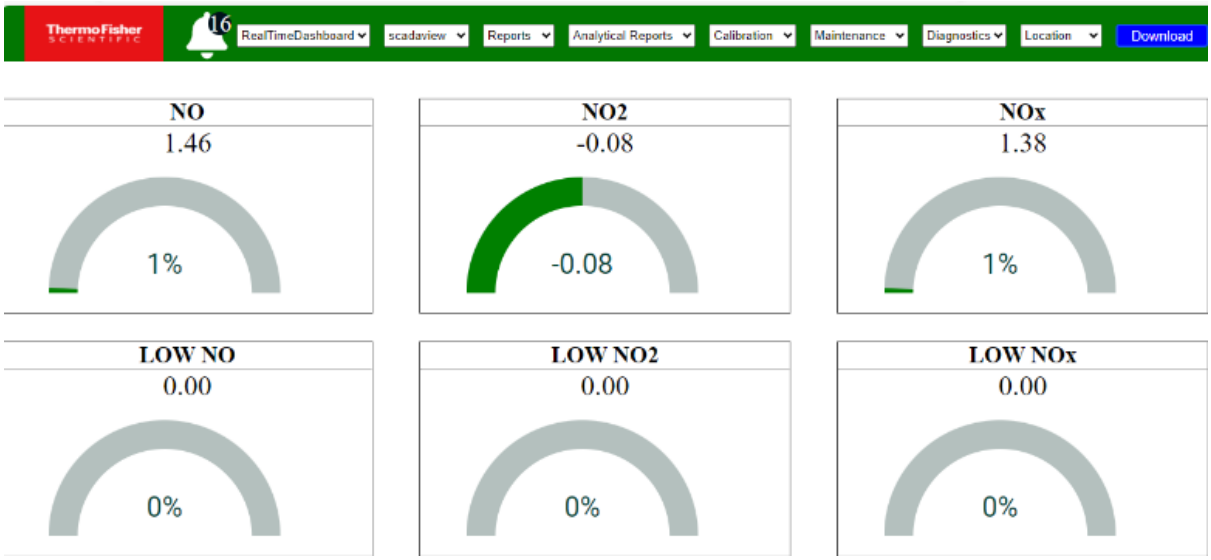
System Alarms

Register Name	Register Value	Time Stamp
SO2 CONC MAX ALARM	false	14/6/2024 3:31:33 pm
SO2 CONC MIN ALARM	false	14/6/2024 3:31:33 pm
INTERNAL TEMP ALARM	true	14/6/2024 3:31:33 pm
CHAMB TEMP ALARM	true	14/6/2024 3:31:33 pm
CONVERTER TEMP	false	14/6/2024 3:31:33 pm
PERM GAS TEMP	false	14/6/2024 3:31:33 pm
PRESSURE ALARM	false	14/6/2024 3:31:33 pm
SAMPLE FLOW ALARM	false	14/6/2024 3:31:33 pm
FLASH REF ALARM	false	14/6/2024 3:31:33 pm
FLASH VOLTAGE ALARM	false	14/6/2024 3:31:33 pm
MOTHERBOARD STATUS ALARM	false	14/6/2024 3:31:33 pm
MEASUREMENT INTERFACE BOARD STATUS ALARM	false	14/6/2024 3:31:33 pm
I/O EXP BD STATUS ALARM	false	14/6/2024 3:31:33 pm

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

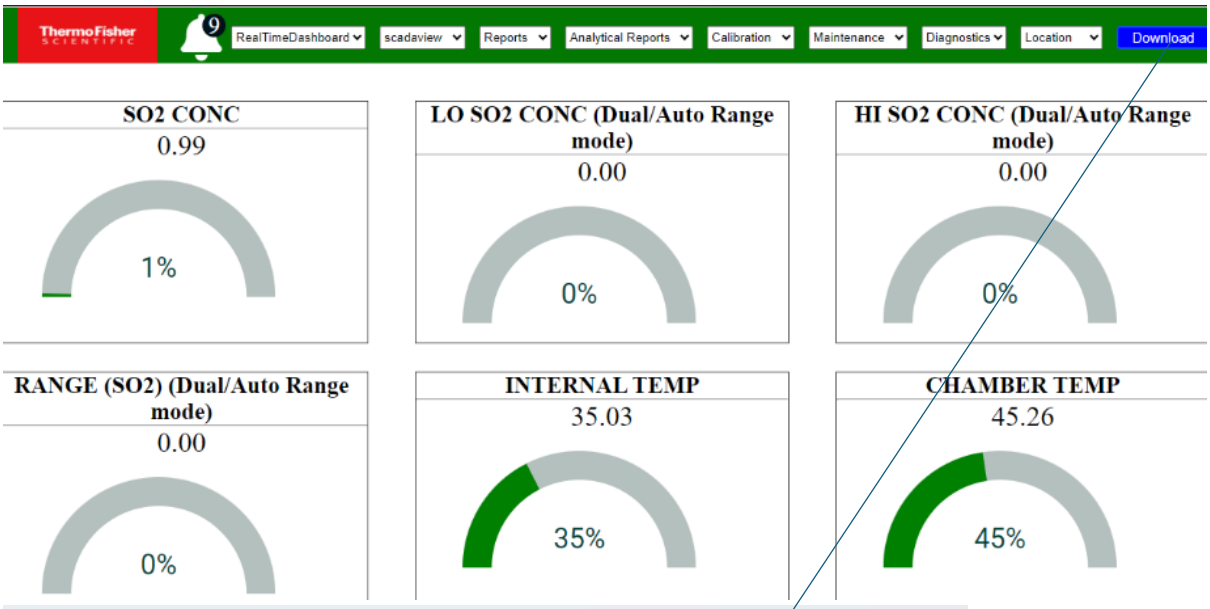
NO2 home page



Downloaded CSV file

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**



	A	B
	RegName	Value
	SO2 CONC	0.985543251
	LO SO2 CONC (Dual/Auto Range mode)	0
	HI SO2 CONC (Dual/Auto Range mode)	0
	RANGE (SO2) (Dual/Auto Range mode)	0
	INTERNAL TEMP	35.02821732
	CHAMBER TEMP	45.25749588
	PERM OVEN GAS TEMPERATURE	0
	PERM OVEN HEATER TEMPERATURE	0
	CHAMBER PRESSURE	665.5291138
	SAMPLE FLOW	0.42479378
	PMT VOLTS	-609.0206299
	FLASH VOLTS	847.7615356
	FLASH REF	89
	ANALOG IN 1 (I/O Expansion Board installed)	0.00262
	ANALOG IN 2 (I/O Expansion Board installed)	0.00262
	ANALOG IN 3 (I/O Expansion Board installed)	0
	ANALOG IN 4 (I/O Expansion Board installed)	0.00262
	ANALOG IN 5 (I/O Expansion Board installed)	0
	ANALOG IN 6 (I/O Expansion Board installed)	0.00262
	ANALOG IN 7 (I/O Expansion Board installed)	0.00262
	ANALOG IN 8 (I/O Expansion Board installed)	0.00524
	EXTERNAL ALARMS	0
	O2 % (O2 Sensor installed)	0
	O2 SENSOR TEMP (O2 Sensor installed	0
	CORRECTED SO2 (O2 Sensor installed)	0
	LO CORRECTED SO2 (O2 Sensor installed)	0
	HI CORRECTED SO2 (O2 Sensor installed)	0
	SO2 BACKGROUND	11.90635395
	LO SO2 COEFFICIENT	1.025217772
	HI SO2 COEFFICIENT	1

Code Snippets

```
import { setGaugeValue } from "./util/gauge.js";
let SO2reg=[];
let newScreen=``;
let newScreen2=``;
let a='';
axios.get('http://localhost:8000/getRegVal')
  .then(response => {
    let dashboardSO2HTML='';
    console.log(response.data);
    SO2reg = response.data.formatted_data;
    let firstKey = Object.keys(SO2reg)[0];
    let firstValue = SO2reg[firstKey];
    // console.log(firstKey);
    console.log(firstValue.RegName);


        dashboardSO2HTML+=`<div class="grid-item">
        <p>
            ${firstValue.RegName}
        </p>
        <hr>
        <div class="data">
            <div class="regvalue">
                ${parseFloat(firstValue.Value).toFixed(2)}
            </div>
            <div class="gauge_SO2">

                <div class="gauge__body">

                    <div class="gauge__fill">

                    </div>

                    <div class="gauge__cover">

                    ${parseFloat(firstValue.Value).toFixed(2)}

                    </div>

                </div>
            </div>
        </div>
    </div>`;
    document.querySelector(".js-regvalueSO2").innerHTML = dashboardSO2HTML;
```

```
for (let regName in SO2reg) {
    if (SO2reg.hasOwnProperty(regName)) {
        let value = SO2reg[regName];

        newScreen += `
        <tr>
            <td>${value.RegName}</td>
            <td>${value.Value.toFixed(2)}</td>
        </tr>`;

        count++;
        if (count === 5) {
            break; // Exit the loop after 5 iterations
        }
    }
}
document.querySelector(".js-SO2registervalues").innerHTML=newScreen;
```

Figure 1 Dynamic display of 5 register values

```
  document.querySelector(".js-regvalueSO2").innerHTML = dashboardSO2HTML;
  let gaugeElement = document.querySelectorAll(".gauge_SO2");
  console.log(gaugeElement);
  // gaugeElement.forEach((item)=>{
  // setGaugeValue(item,(SO2reg["Value"][i]))/100;})
  gaugeElement.forEach((item) => {
    setGaugeValue(item, firstValue.Value / 100); // Assuming setGaugeValue function expects a value between 0 and 1.

});


  newScreen = `
  <h3> SO2 Register Values </h3>

  <table border="1">

  <tr>

  <th> Register Name </th>
  <th> Register Value</th>


  </tr>
  `;
```

Figure 2 Display of data with Gauge

INDIA ENGINEERING CENTER

TITLE: **Development of Data Acquisition Software (DAS) for CAAQMS.**

```
axios.get('http://localhost:8000/getNo2RegVal')
.then(response => {
  let dashboardSO2HTML='';
  console.log(response.data);
  SO2reg = response.data.formatted_data;
  let firstKey = Object.keys(SO2reg)[1];
  let firstValue1 = SO2reg[firstKey];
  // console.log(firstKey);
  console.log(firstValue1.RegName);


    dashboardSO2HTML+=`<div class="grid-item">
    <p>
        ${firstValue1.RegName}
    </p>
    <hr>
    <div class="data">
        <div class="regvalue">
            ${parseFloat(firstValue1.Value).toFixed(2)}
        </div>
        <div class="gauge_NO2">

            <div class="gauge__body">

                <div class="gauge__fill">

                </div>

                <div class="gauge__cover">

                ${parseFloat(firstValue1.Value).toFixed(2)}

                </div>

            </div>
        </div>
    </div>
  </div>`;
  document.querySelector(".js-regvalueNO2").innerHTML = dashboardSO2HTML;
   let gaugeElement = document.querySelectorAll(".gauge_NO2");
  console.log(gaugeElement);
  // gaugeElement.forEach((item)=>{
  // setGaugeValue(item,(SO2reg["Value"][i]))/100;})
  gaugeElement.forEach((item1) => {
    setGaugeValue(item1, firstValue1.Value / 100); // Assuming setGaugeValue function expects a value between 0 and 1.

});
```

Figure 3URL generation for the webpage

```
from pymodbus.client import AsyncModbusTcpClient
from pymodbus.payload import BinaryPayloadDecoder
from pymodbus.constants import Endian
from pymodbus.exceptions import ModbusException
```

Figure 4 Import Statements to use standard libraries

```
def __init__(self,host,port,slave):
    self.client = AsyncModbusTcpClient(host,port)
    self.slave = slave

async def connect(self):
    try:
        await self.client.connect()
        if(self.client.connected):
            print("Connected to Modbus Server")
            status = "Connected"

    except ModbusException:
        raise Exception("Could not connect to Modbus Server")
        status = "Could not connect to Modbus Server"
    return status


def close(self):
    self.client.close()
    print("Disconnected from Modbus Server")
    status = "Disconnected"

async def readHoldingReg(self) -> object:
    decoded_values = []
    responseObj = {}
    try:
        result = await self.client.read_holding_registers(0,86, slave=self.slave)
        for i in range(0,len(result.registers),2):
            decoder = BinaryPayloadDecoder.fromRegisters(result.registers[i:i+2], Endian.BIG, wordorder=Endian.LITTLE)
            print("decoder"+str(decoder))
            value = decoder.decode_32bit_float()
            print("value"+str(value))
            decoded_values.append(value)
        for i in range(0,len(decoded_values)):
            if(So2ReadReg.registerVariables[i] == 'NOT USED'):
                continue

            responseObj[So2ReadReg.registerVariables[i]] = decoded_values[i]

        alarmreg =  await self.client.read_coils(0, 38,slave=self.slave)
        print("Coil values:", alarmreg.bits)



        formatted_data = [{"RegName": key, "Value": value,}for key, value in responseObj.items()]
        response_to_return={"formatted_data":formatted_data,"Alarms":alarmreg.bits}

        return response_to_return
    except ModbusException as exc:
        print(f"Received ModbusException({exc}) from library")
        raise exc
    except Exception as exc:
        print(f"An exception occurred: {exc}")
        raise exc
```

Figure 5 Logic for communicating to the instrument from Backend

```python
from fastapi import FastAPI
from Backend.SO2ReadReg import So2ReadReg
import time
from asyncio import Event
from fastapi.middleware.cors import CORSMiddleware
from Backend.NO2ReadReg import No2ReadReg
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE"],
    allow_headers=["*"],
)
So2Device =  So2ReadReg("192.168.0.201",502,43)
@app.get("/getRegVal")
async def sendSo2Data():
    response = None
    try:
        print("Got get data request from client")
        devConStat = await So2Device.connect()

        if(devConStat == "Connected"):
            response = await So2Device.readHoldingReg()
            print(response)
        else:
            response = devConStat
        time.sleep(1)
    except Exception as ex:
        print("Some error occured")
    return response


No2Device =  No2ReadReg("192.168.0.202",502,42)
@app.get("/getNo2RegVal")
async def sendNo2Data():
    response = None
    try:
        print("Got get data request from client")
        devConStat = await No2Device.connect()

        if(devConStat == "Connected"):
            response = await No2Device.readHoldingReg()
            print(response)
        else:
            response = devConStat
        time.sleep(1)
    except Exception as ex:
        print("Some error occured")
    return response
```

nt

Figure 6 fast API library for connection establishment

JavaScript to React

We can convert the JavaScript code to React by using the JSX converter, which mainly helps us convert JS's logical part to React.

https://codepen.io/zenbrent/pen/vYQLXb

The JS2 flowchart converts the JavaScript code to a flowchart representation which mainly helps in visualizing the logical flow so that the plan becomes easy to structure the React components.

The React converter as a tool helps us to convert the JavaScript code to React components.

Finally, the above-mentioned points provide an overview, of how to transform the code of JS to React components by understanding the flow of code in JS and its logical representation through flowcharts which provides a keen way of building the logic to React components.