A Smart Factory Ventilation System

Problem statement: A smart factory ventilation system includes two MCUs. MCU1 reads the *CO* and the *ambient temperature* levels and reports them to a cloud application. On the other hand, MCU2 receives commands form the cloud application to open/close a *window* and a *fan*.

You are required to build a simulation of the above application consisting of a *cloud* and an *edge layers*. The cloud layer consists of a single process which controls the system and displays a dashboard showing a chart of the reported readings and the status of the window and fan. The edge layer contains the two MCUs. Figure 1 shows an architectural view of the system.

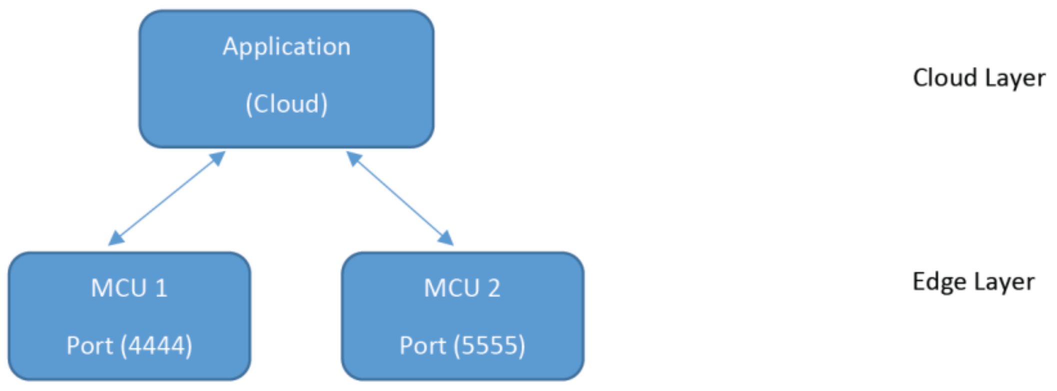


Figure 1: Architectural view of a Smart Factory Ventilation System.

The cloud application periodically polls the sensor nodes to retrieve the sensor data and plots the data using an *animated* graph. If the sensor reading exceeds certain thresholds, **a command is sent by the cloud** to turn on/off the devices. You need to implement the MCUs inside PT and the cloud service outside PT (using Python Script).

Use the TCP/UDP port number as shown in the Figure above. Your system must implement the following requirements:

1. The cloud service polls each sensor node once every 100 *ms* to request the sensors' readings.
2. The cloud service must display a live *line chart* showing the readings of each sensor and the status of the devices.
3. The reading should be continuous, and plots should keep updating continuously.
4. You need to define a simple protocol (messages) to request turning on/off the warning lights.
5. You need to define the threshold values for the sensors.

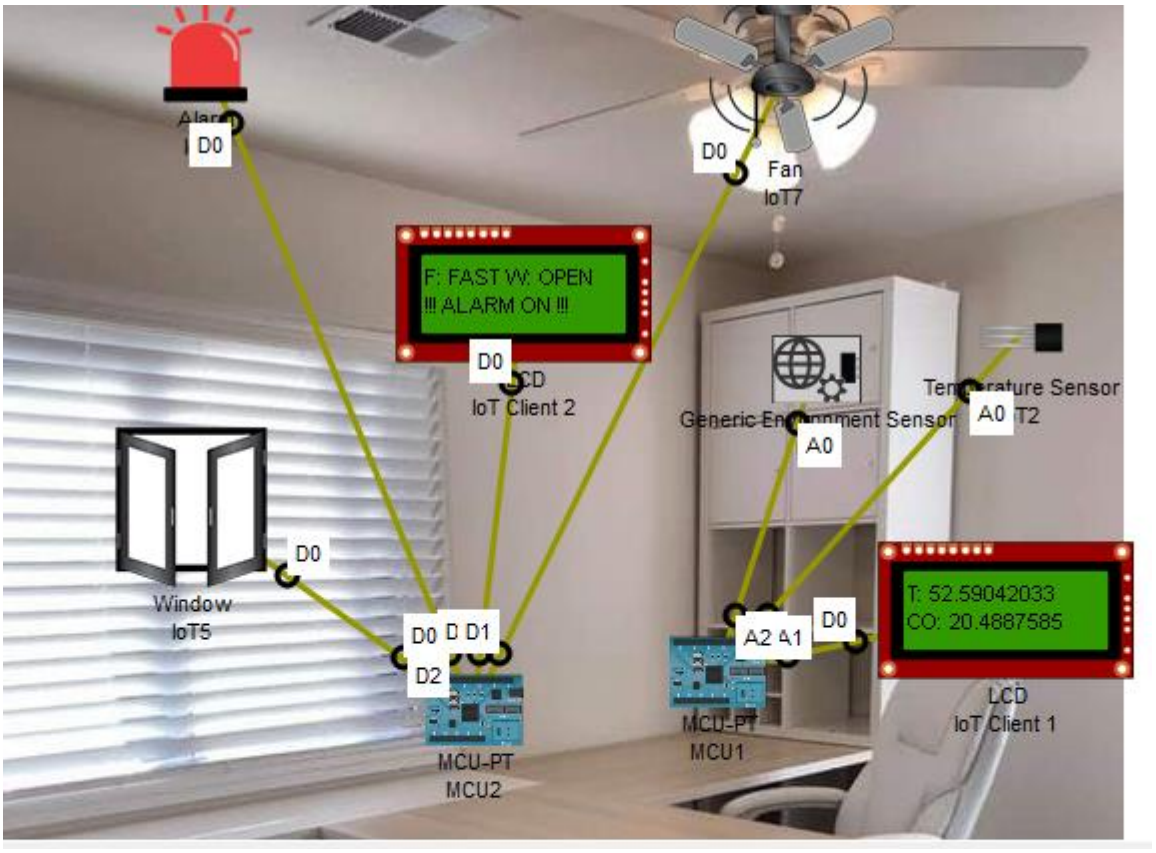
Connect your sensors to an SBC or MCU to enable connectivity. For programming in PT, use the RealTCPServer(), RealUDPServer() , RealTCPClient() and/or RealUDPClient() template.

Report: attach a report with your solution that includes:

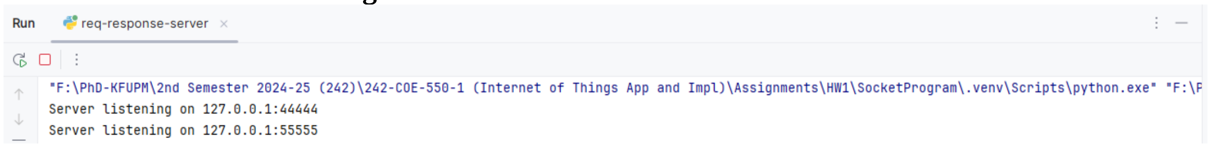
1. Screenshot of the program output.
2. Types of sensors used and threshold you define for each sensor.
3. Any problems or limitations in your solution.
4. Protocol messages you specify between the cloud and sensor nodes

Solution:
I have simulated the proposed system in a Packet tracer. I have used Python as a scripting language to program both the cloud and edge layer.

<u>Screenshot:</u>

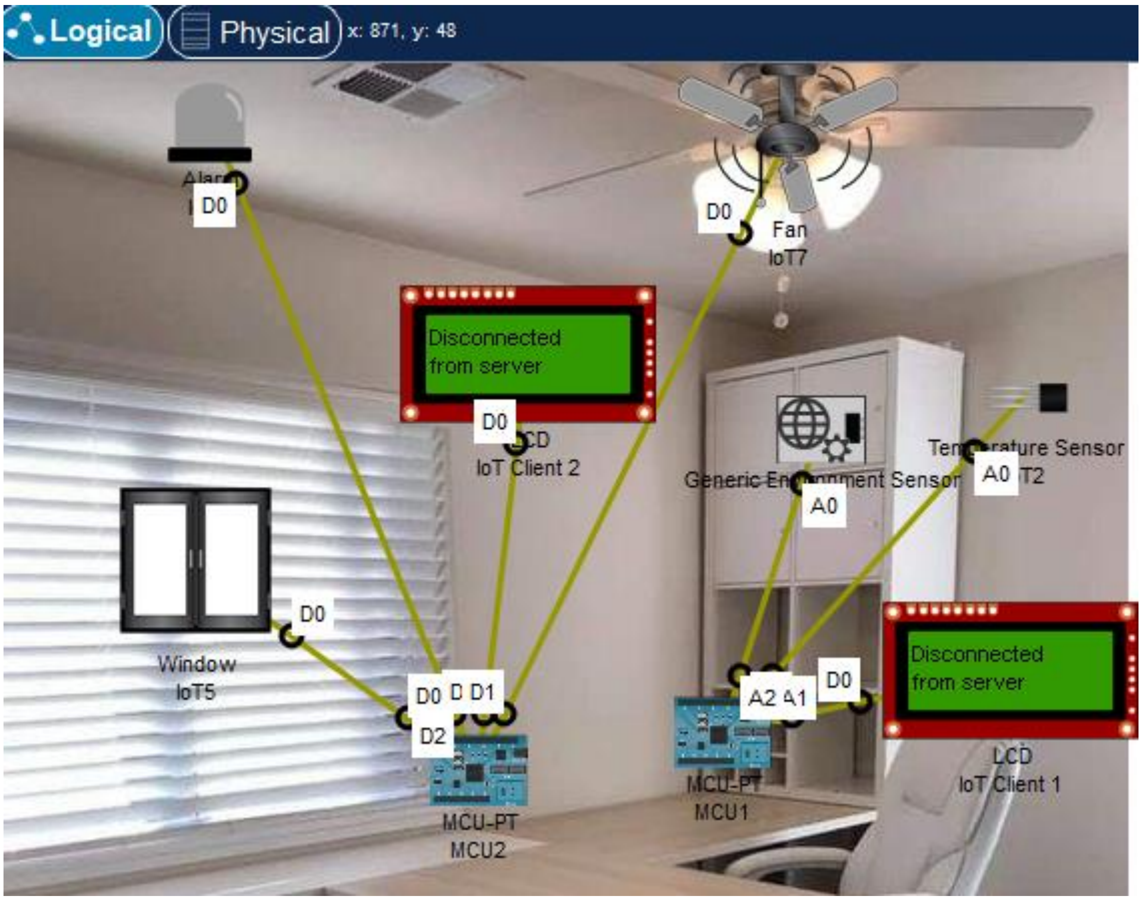


Server is started and listening for the MCU 1 and MCU 2



```
"F:\PhD-KFUPM\2nd Semester 2024-25 (242)\242-COE-550-1 (Internet of Things App and Impl)\Assignments\HW1\SocketProgram\.venv\Scripts\python.exe" "F:\P
Server listening on 127.0.0.1:44444
Server listening on 127.0.0.1:55555
```

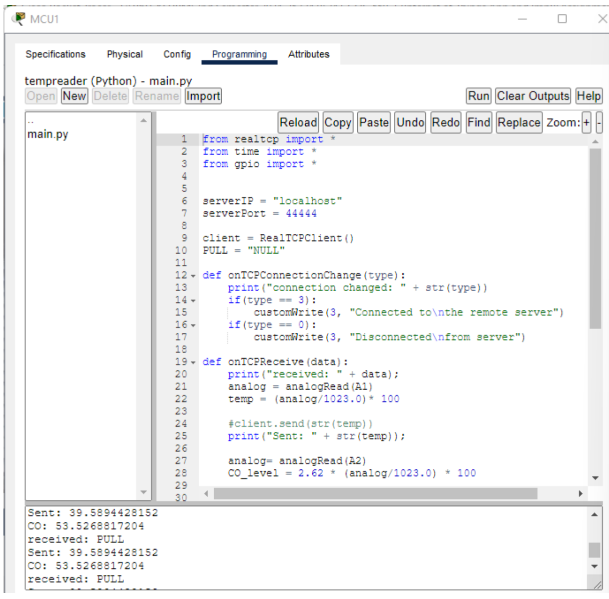
MCU 1 and 2 are disconnected from the server since we have not turned them on yet



The server has been connected to MCU's and sent request to MCU 1 to send sensor data

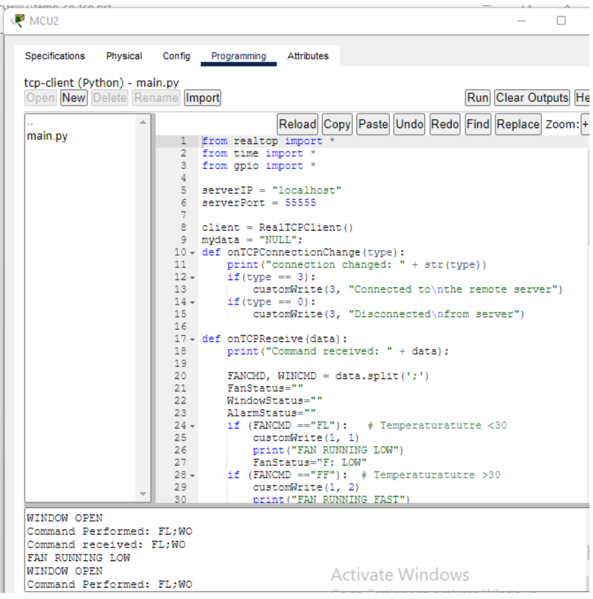
```
Client 1 connected from ('127.0.0.1', 53330)
Client 2 connected from ('127.0.0.1', 53321)
PULL request: 1
Received from Client 1: temperature:59.6285434995;co:48.4046920821
```

MCU 1 receives the PULL request and sends the sensor data to the server. And the server sends the actuator commands to the MCU 2. When the MCU 2 performs the commands it sends the ACK back to the server.



```python
from realtcp import *
from time import *
from gpio import *


serverIP = "localhost"
serverPort = 44444

client = RealTCPClient()
PULL = "NULL"

def onTCPConnectionChange(type):
    print("connection changed: " + str(type))
    if(type == 3):
        customWrite(3, "Connected to\nthe remote server")
    if(type == 0):
        customWrite(3, "Disconnected\nfrom server")

def onTCPReceive(data):
    print("received: " + data);
    analog = analogRead(A1)
    temp = (analog/1023.0)* 100

    #client.send(str(temp))
    print("Sent: " + str(temp));

    analog= analogRead(A2)
    CO_level = 2.62 * (analog/1023.0) * 100
```

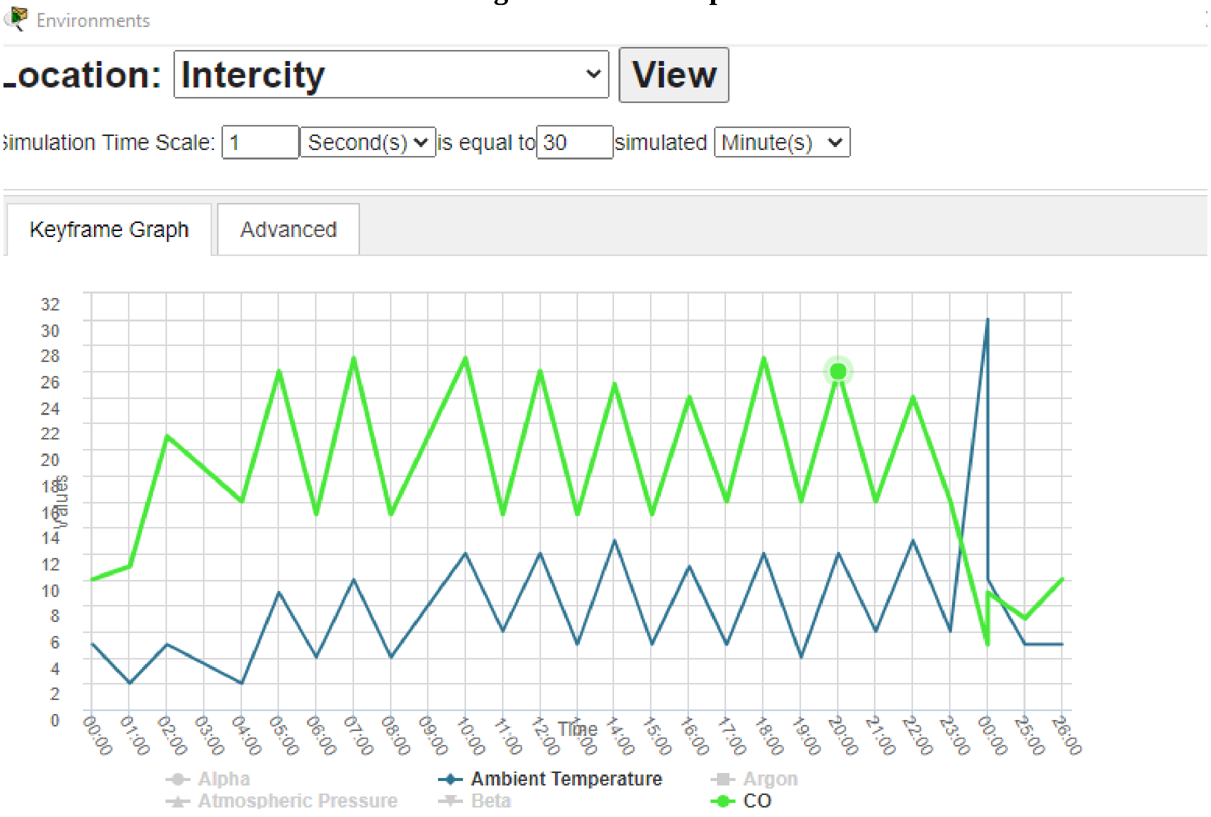
```
Sent: 39.5894428152
CO: 53.5268817204
received: PULL
Sent: 39.5894428152
CO: 53.5268817204
received: PULL
```



```python
from realtcp import *
from time import *
from gpio import *

serverIP = "localhost"
serverPort = 55555

client = RealTCPClient()
mydata = "NULL";
def onTCPConnectionChange(type):
    print("connection changed: " + str(type))
    if(type == 3):
        customWrite(3, "Connected to\nthe remote server")
    if(type == 0):
        customWrite(3, "Disconnected\nfrom server")

def onTCPReceive(data):
    print("Command received: " + data);

    FANCMD, WINCMD = data.split(';')
    FanStatus=""
    WindowStatus=""
    AlarmStatus=""
    if (FANCMD =="FL"):   # Temperaturatutre <30
        customWrite(1, 1)
        print("FAN RUNNING LOW")
        FanStatus="F: LOW"
    if (FANCMD =="FF"):  # Temperaturatutre >30
        customWrite(1, 2)
        print("FAN RUNNING FAST")
```

```
WINDOW OPEN
Command Performed: FL;WO
Command received: FL;WO
FAN RUNNING LOW
WINDOW OPEN
Command Performed: FL;WO
```

The environment has been set according to the timestamp



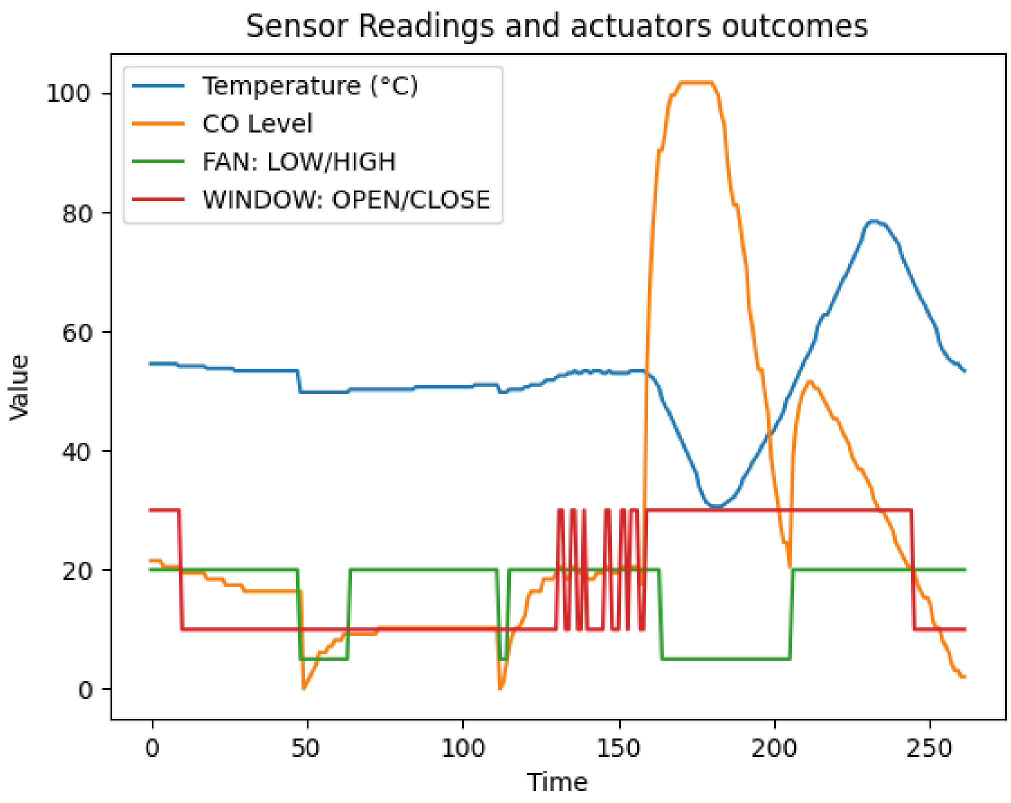
Environments

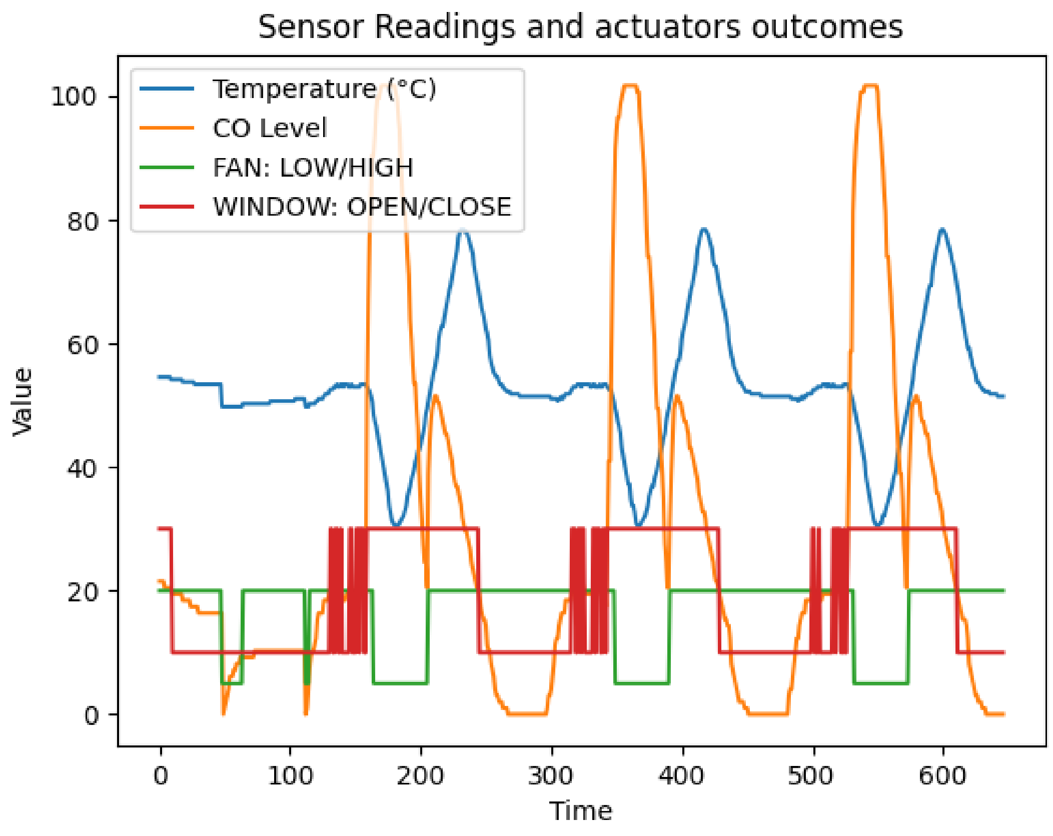
Location: Intercity View

Simulation Time Scale: 1 Second(s) is equal to 30 simulated Minute(s)

Keyframe Graph | Advanced

The changes shown in graph.





Types of sensors used:

The MCU 1 includes the following sensors

1. **Generic Environment Sensor (CO detector)**
2. **Temperature Sensor**

Threshold defined for each sensor

Sensor	MAX	MIN	Threshold
Temperature	-100	100	50
Generic Environment Sensor	0	25	20

Problems and limitations:
In this simulation I have faced the following problems and limitations and overcome the problems as per the table below

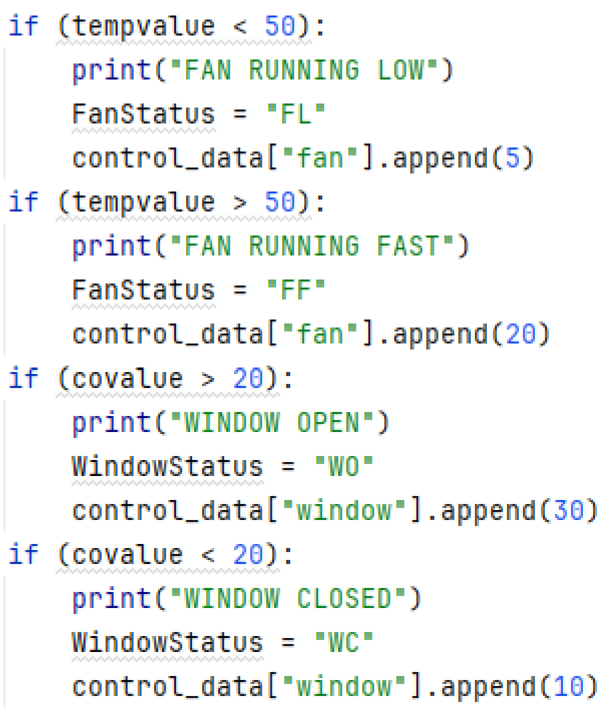
SL	Problems	Proposed solution
1	Can't read the exact Ambient temperature shown in the environment section using the temperature sensor	I kept what is found in my sensor and found that temperature slightly changes more or less 50°C. So I have used 50°C as a threshold for this sensor

2	Can’t read the exact % CO shown in the environment section using the generic environment sensor	The sensor reading I got from the generic environment sensor and the environment were not similar then I tried to make correlation and multiplied my reading with 2.62 so that I was slightly near to the environment reading.
3	There is no change happens even if I simulated an old car near by the sensors.	I didn’t use any old car. I set the temperature and co level in the environment section according to the timestamp

Protocol messages you specify between the cloud and sensor nodes

Cloud to MCU 1: Server sends “**PULL**” message to the MCU 1 then MCU 1 send the sensors data return

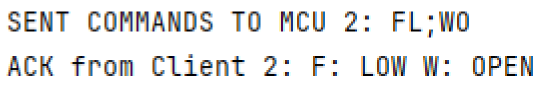
Cloud to MCU 2: According to the threshold value the server set the following commands



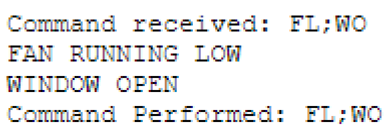
```
if (tempvalue < 50):
    print("FAN RUNNING LOW")
    FanStatus = "FL"
    control_data["fan"].append(5)
if (tempvalue > 50):
    print("FAN RUNNING FAST")
    FanStatus = "FF"
    control_data["fan"].append(20)
if (covalue > 20):
    print("WINDOW OPEN")
    WindowStatus = "WO"
    control_data["window"].append(30)
if (covalue < 20):
    print("WINDOW CLOSED")
    WindowStatus = "WC"
    control_data["window"].append(10)
```

FAN		WINDOW		ALARM
FL	SLOW	WO	OPEN	ON
FF	HIGH	WC	CLOSE	OFF

The server sends the commands as **FL:WO** to MCU 2 if it requires to operate the FAN in LOW speed and OPEN the WINDOW also to turn ON the ALARM when temperature is less than 50 C but CO level >20.



```
SENT COMMANDS TO MCU 2: FL;WO
ACK from Client 2: F: LOW W: OPEN
```



```
Command received: FL;WO
FAN RUNNING LOW
WINDOW OPEN
Command Performed: FL;WO
```

As soon as MCU 2 performs the commands it sends the ACK to the server.