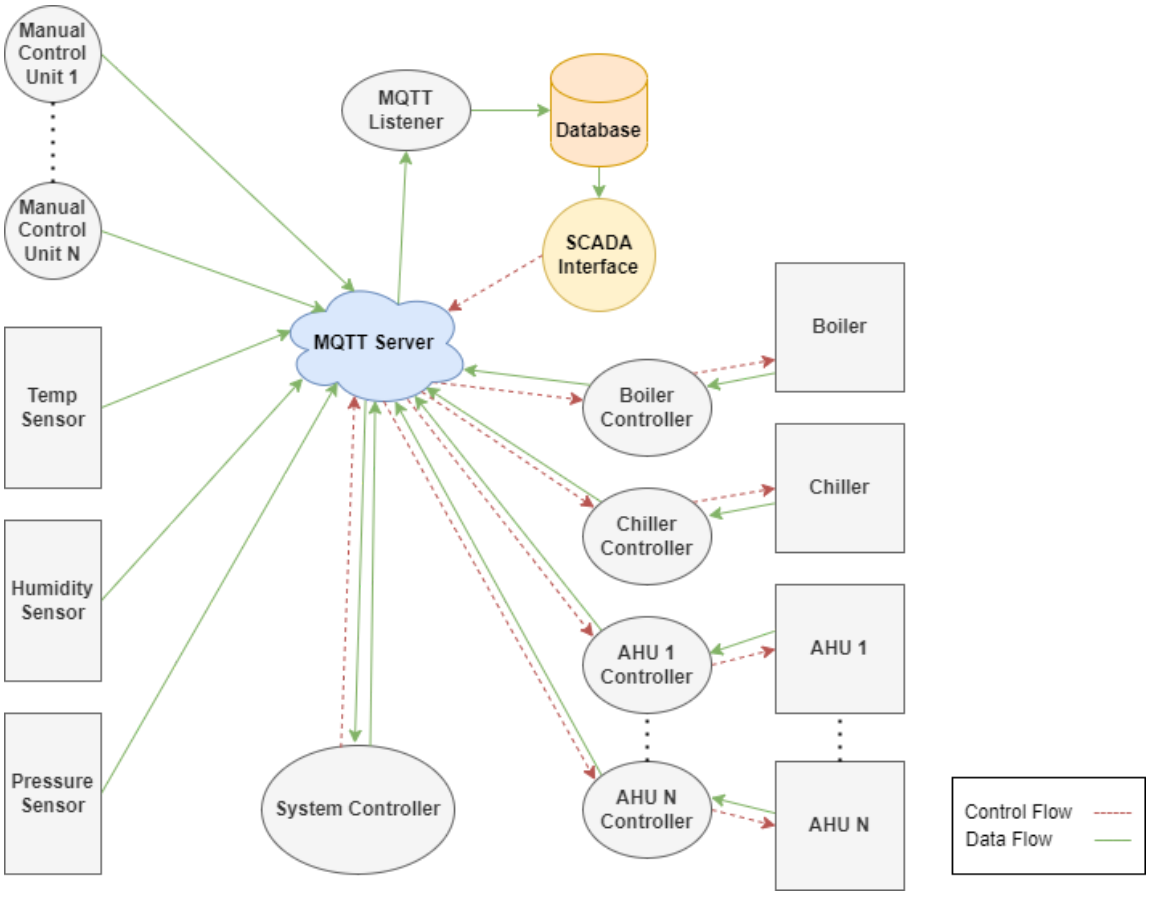
# CO326
# Computer Systems Engineering - Industrial Networks

## Mini-Project - Smart Building

## GROUP A - HVAC

# Table of Contents

# Solution Architecture




At a high level, the overall system will be controlled by a system controller while dedicated IoT controllers at each actuator take decisions on the behaviour of the actuators using the control inputs from the overall system controller and provide data regarding the status of the actuators to the system controller.

The system controller's decisions are guided by the sensor data which is conveyed to the system controller via MQTT-enabled IoT devices that are attached to the sensors. The climate settings for each zone can be adjusted using manual control units placed at each zone.

All communication between the nodes of the system will take place over an MQTT server and an MQTT Listener will process all data being transmitted over the server and log them into a database for analytics.

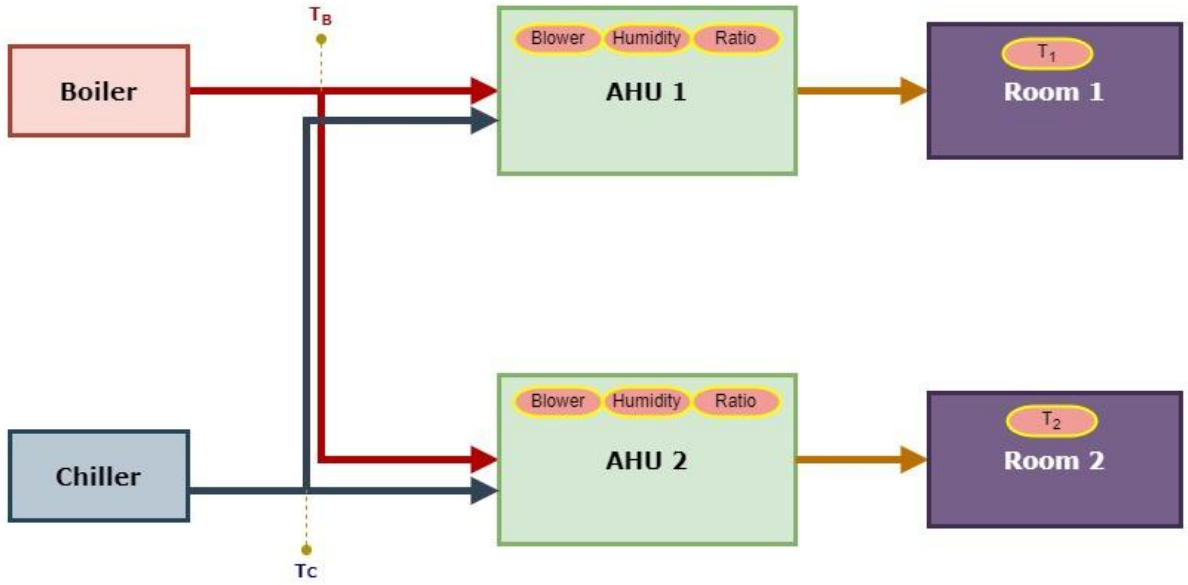
The SCADA interface can view the status of the system by retrieving the data logged into the database. It will also be able to control the climate control zones as well as the actuators themselves, overriding the settings of the manual control units, by sending MQTT messages to the system controller.

# Actuators

The actuators of this system include the boiler unit, chiller unit and the AHU units.

| **Actuator** | **Available Options** | **Notable Features** |
|---|---|---|
| Boiler | WDR Electric Heating Steam Boiler | ● Environmental protection<br>● Air ignition alarm<br>● Interlock<br>● Fully automatic control<br>● No noise<br>● Small occupation |
| Chiller | Water-Cooled Centrifugal | ● Low energy consumption during part load and full load operation.<br>● Dual inlet guide vanes for efficiency and stability.<br>● Quiet operation – less than 80 dBA. |
| AHU | CLIMACS AHU | ● Fresh Air Handling Unit<br>● Heat Recovery<br>● Automatic Control<br>● Pool Dehumidification Unit |

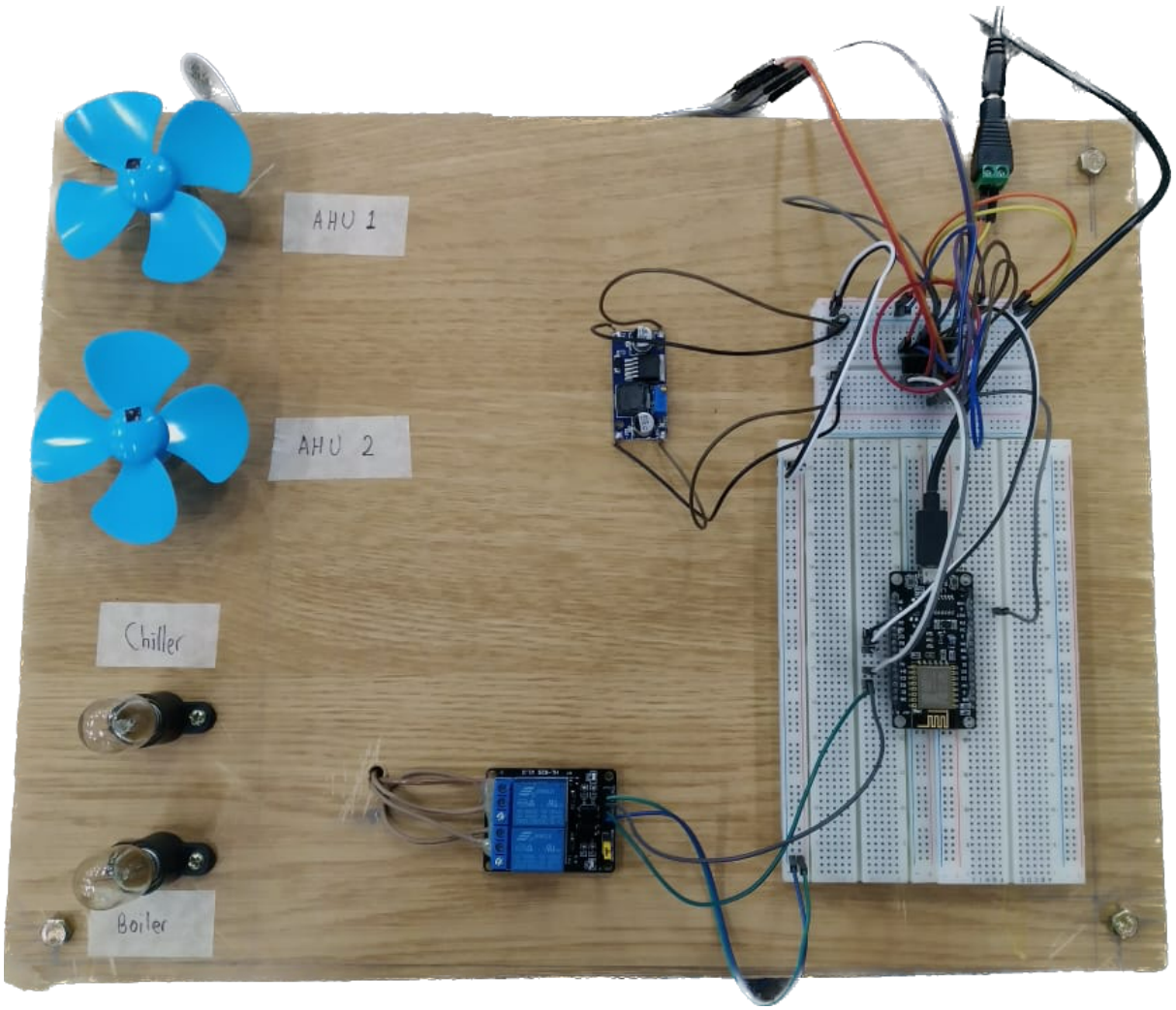
For practical reasons, we have not used these industrial grade actuators in our model. Instead, we have developed our final demonstration with dummy models for the aforementioned units (i.e. an AC lamp each for the boiler and the chiller, and two fans to represent the blowers of each of the AHUs.)




The above diagram represents the general flow of hot and cold air and the underlying details of how they mix in order to form the final room temperature.

The boiler and the chiller operate based on an MQTT control signal that switches them on and off. The boiler, while on, will continue to increase the temperature of the air duct connected to it whereas the chiller will provide cool air through its air duct. Under typical operating conditions, the boiler will provide hot air of a set temperature $T_B$ and the chiller will provide cold air of set temperature $T_c$.

The system contains an Air Handling Unit (AHU) for each room. The AHU controls the temperature for each room by mixing the hot air and cold air to produce air of the required temperature for the room. The AHU contains dampers on each of the air ducts to control the mix ratio between the hot air and cold air. This way, the boiler and chiller can operate at a constant temperature to maintain high efficiency while the AHU provides flexible control of temperature for each of the rooms individually.




The setup given above is the final model constructed for the demonstration of the project. The boiler and chiller states are represented using lamp indicators and the blower speeds of two AHU units are represented through the two fans. The NodeMCU ESP8266 acts as the field controller for the actuators by listening to control signals over MQTT and modifying the states of the actuators accordingly.

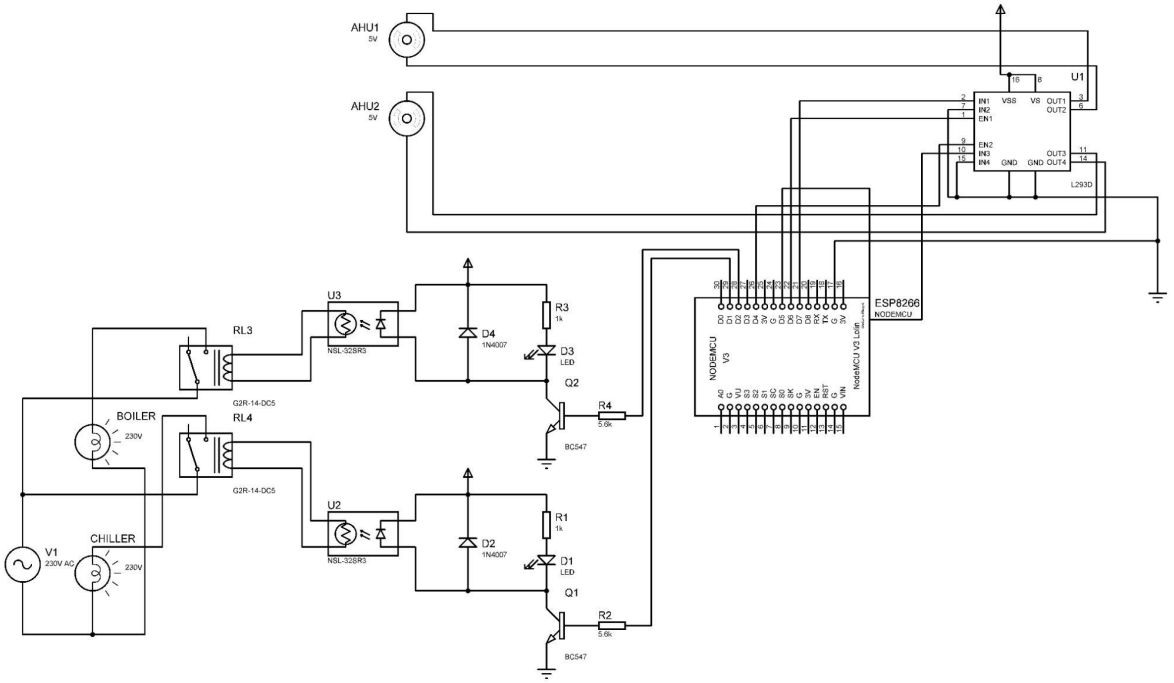
In a real implementation, each of the actuators would have a dedicated field controller of its own since it is unlikely that all the actuators will be in such close proximity to each other. It is also not very practical in terms of reliability since using a single field controller to control all the actuator units will introduce a single point of failure (the MCU) that will take down the entire actuator network if it happens to fail.

However, due to cost and practicality concerns, we have opted for the above setup as it properly demonstrates the capabilities of the designed system while keeping the hardware model small and inexpensive.

AC lamps operated via relay modules were chosen as dummy models for the boiler and the chiller since these are typically AC-powered units. However, while industrial

AHUs employ AC fans for their blower units, it is not practical to use such fans in our demonstration model since variable speed AC fans are bulky and expensive. Hence, we have opted for DC motors controlled by a L293D motor driver as a practical and economical alternative. Regardless, the operating principle of the control system remains the same whether the fans are AC or DC, making the DC motors suitable stand-ins for demonstrating the behaviour of the implemented control system.

The schematic diagram for the designed hardware model is shown below.



# Sensors (E/17/296)

The HVAC unit requires the use of the following sensors:

- SAN Electro Heat Type TM1-P - 1-step electronic room thermostat
- SAN Electro Heat Type EFR 012 - electronic hygrostat
- E+E EE600 Differential Pressure Sensor
- Saturn PR-276 Duct Pressure Sensor

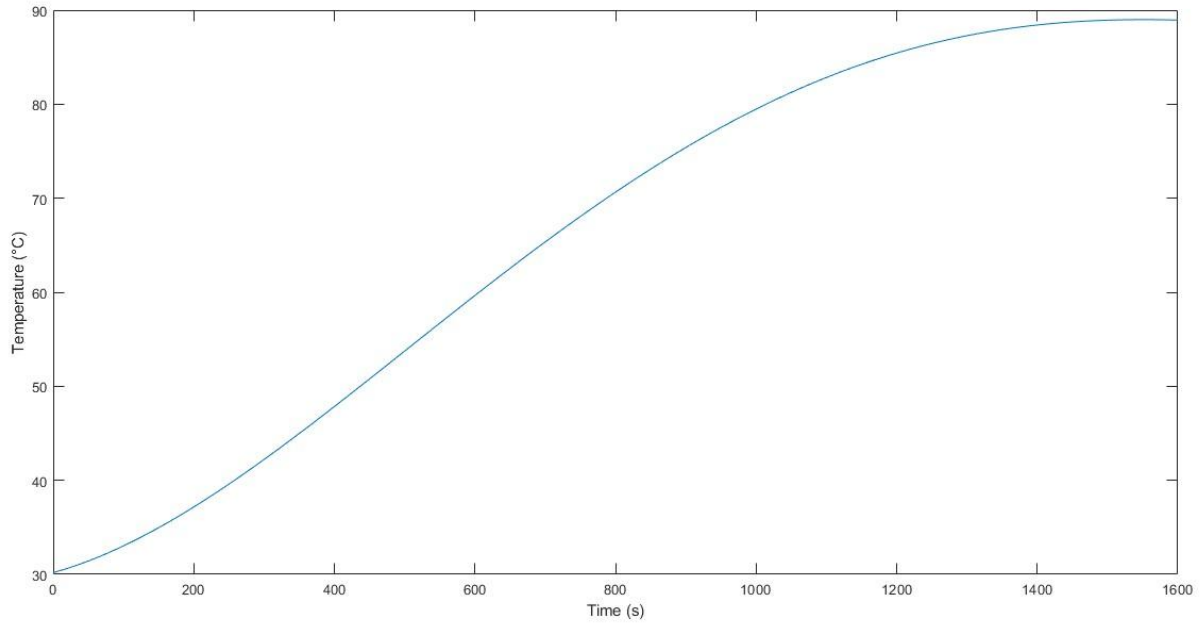
The issue with using dummy models for the actuators as mentioned above is that they do not provide any feedback to the rest of the system once they are on (e.g. turning on a lamp will not create a noticeable change in a temperature sensor). Hence, it is not very useful to include physical sensors in our demonstration since none of them will react to actuator states.

Hence, we have opted to simulate the consequent sensor values using realistic models instead of using physical sensors for the demonstration. This allows us to operate the system autonomously since the whole system will react in realtime to changes in actuator states.

The boiler and chiller make major contributions in maintaining the room temperature for both rooms. The behaviour of the boiler and chiller were modelled using MATLAB by referring the temperature responses of actual industrial-grade boilers and chillers, and their characteristic polynomial curves were used to obtain sets of data points (elapsed time and temperature at the air duct) as a lookup table. These lookup tables are then used by Python scripts that model the air duct temperatures ($T_B$ and $T_C$) according to the state of each of the actuators.

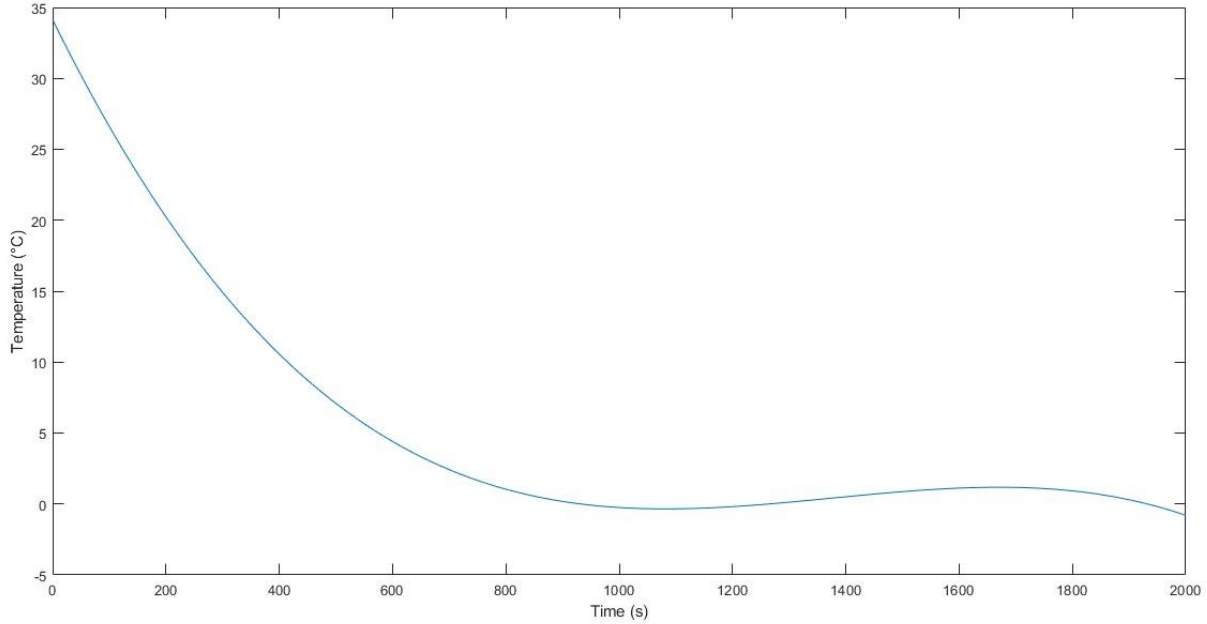
The corresponding boiler characteristic curve used:

```
p1 =   1.608e-11;
p2 =  -7.285e-08;
p3 =   8.549e-05;
p4 =     0.02057;
p5 =       30.19;
f = p1*(x^4) + p2*(x^3) + p3*(x^2) + p4*x + p5;
```




The corresponding chiller characteristic curve used:

```
p1 = -1.5e-08;
p2 = 6.192e-05;
p3 = -0.08131;
p4 = 34.14;
f = p1*x^3 + p2*x^2 + p3*x + p4;
```



These response curves approximately model the behaviour of a typical industrial boiler and chiller. For example, when the boiler starts at room temperature (cold start), the change in temperature is gradual and slow but once an adequate amount of time has passed, the hot air duct temperature increases rapidly. The gradient drops off as the temperature reaches the maximum heating capacity of the boiler and plateaus once the maximum heating is reached. A similar behaviour was modelled for the chiller.

For both the boiler and the chiller, the behaviour when the actuator is off was modelled following Newton's law of cooling (i.e. exponential movement over time of duct temperature towards the ambient room temperature). This ensures that the sensor readings provide an accurate depiction of the real temperature variation when the actuator is turned on and off periodically during operation by the system controller.

```
off_model = room_temp + (temp - room_temp)*exp(-t/120000)
```

The behaviour of the AHU is more complex since it involves both the hot air and cold air ducts and a blower which can operate at a varying speed as dictated by the system controller. The AHU primarily affects the readings of two sensors placed within the room that the AHU is connected to: the variation of the hot/cold air mix ratio induces changes in the temperature sensor, and the blower speed induces changes in the differential pressure sensor.

Following this idea, the temperature sensor within the room was modelled such that it adjusts its temperature according to the mix ratio. That is, a higher ratio (meaning more hot air) leads to a higher temperature reading within the room, while a lower ratio leads to lower temperature readings. Additionally, the blower speed is taken into account to determine how fast the temperature should change upon a change in mix ratio. This is in line with the realistic behaviour since a higher blower speed would mean that air of the new temperature is pushed in faster into the room.

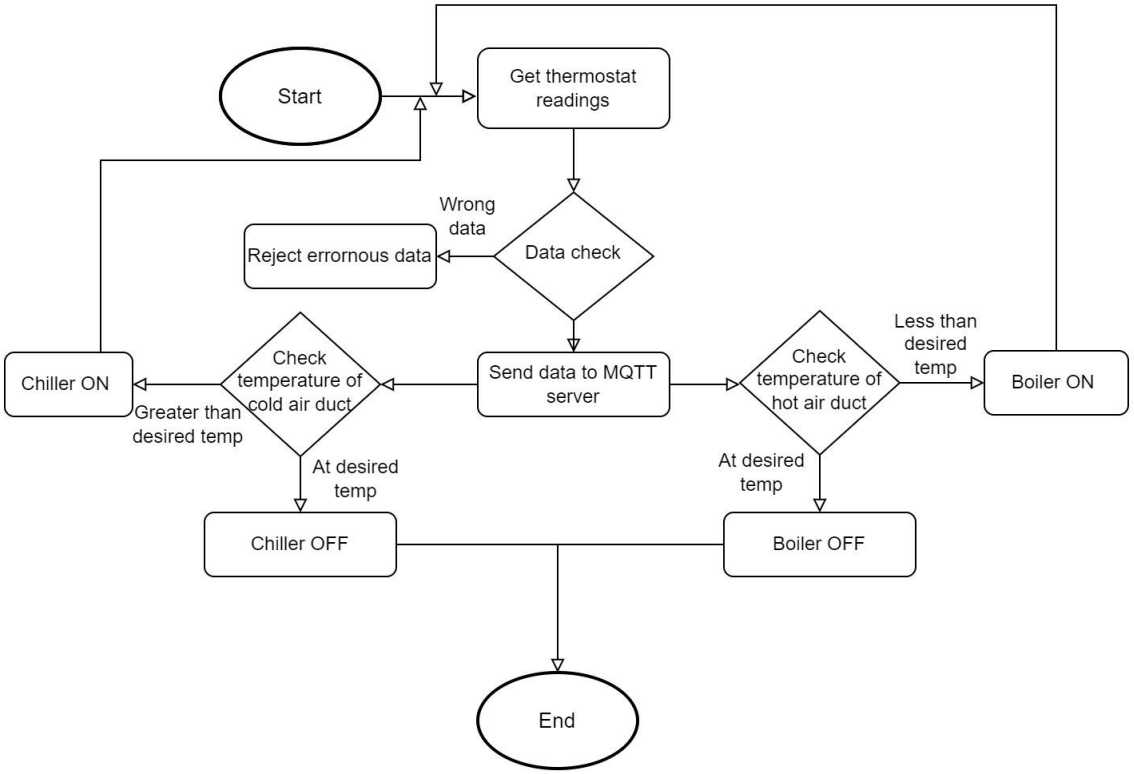
Accordingly, the models for the temperature sensor and the differential pressure sensor are as follows.

```
target_temp = (ratio * Tb  +  (1-ratio) * Tc
new_temp = target_temp+(current_temp-target_temp)*exp(-t/(500/(50+0.5*speed)))
```

```
target_pres = 0.012 * speed
new_pres = target_pres + (current_pres - target_pres) * exp(-t / 10)
```

# System Controller (E/17/006, E/17/206, E/17/297)

## Boiler/Chiller and Main Control




All the temperatures of the Cold air duct and the Hot air duct are obtained by means of subscribing to the following topics.

```
326project/smartbuilding/hvac/coldairduct/temperature
326project/smartbuilding/hvac/hotairduct/temperature
```

When there is a message available from the Cold air duct, we get the incoming temperature value and do the following operations.

If the incoming temperature is greater than the desired temperature range, we will turn on the chiller by publishing the control command to the following control topic.

```
326project/smartbuilding/hvac/control/chiller
```

If the incoming temperature is lower than the desired temperature range, we will turn off the chiller.

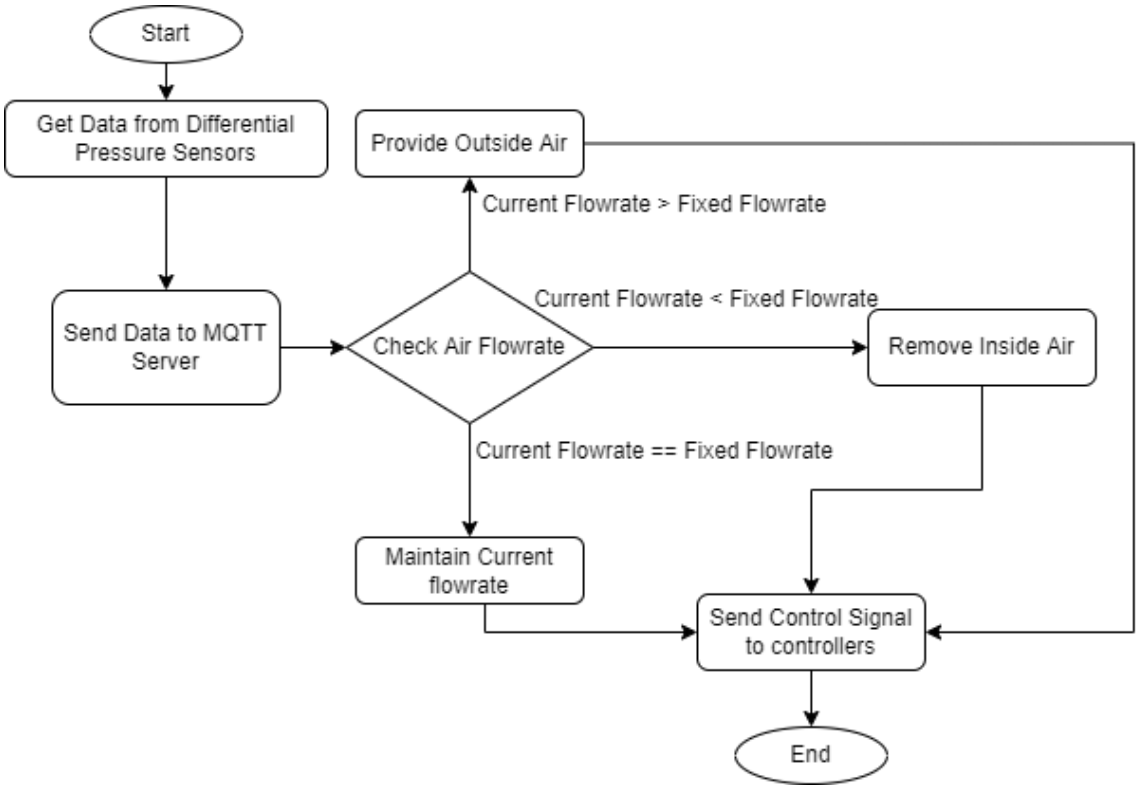
If it is from the Hot air duct, we get the incoming temperature and compare it with the desired temperature range. If the current temperature is greater than the desired temperature range we turn off the Boiler, otherwise we turn it on. We will publish the control command to the following topic.

`326project/smartbuilding/hvac/control/boiler`

Additionally we have subscribed to the following topics to change the threshold values.

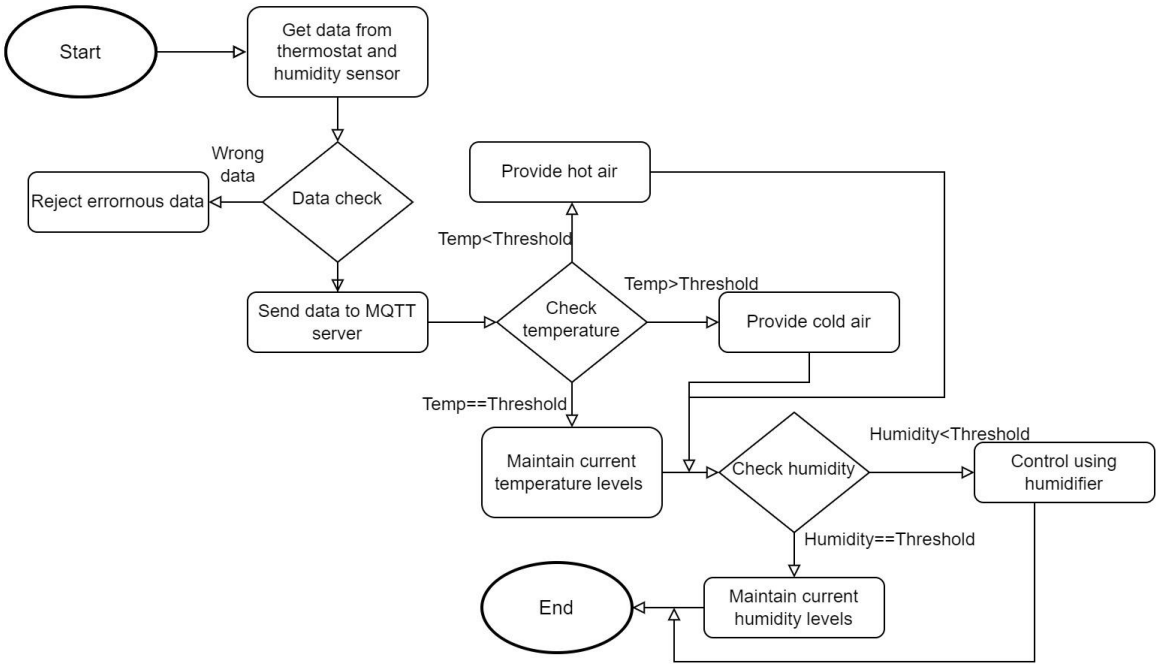
`326project/smartbuilding/hvac/control/temp-thresh`
`326project/smartbuilding/hvac/control/temp-thresh-coldairduct`
`326project/smartbuilding/hvac/control/temp-thresh-hotairduct`

## Ventilation Control in Rooms




In this section we get the temperature, humidity, pressure in a room by subscribing the following topics,

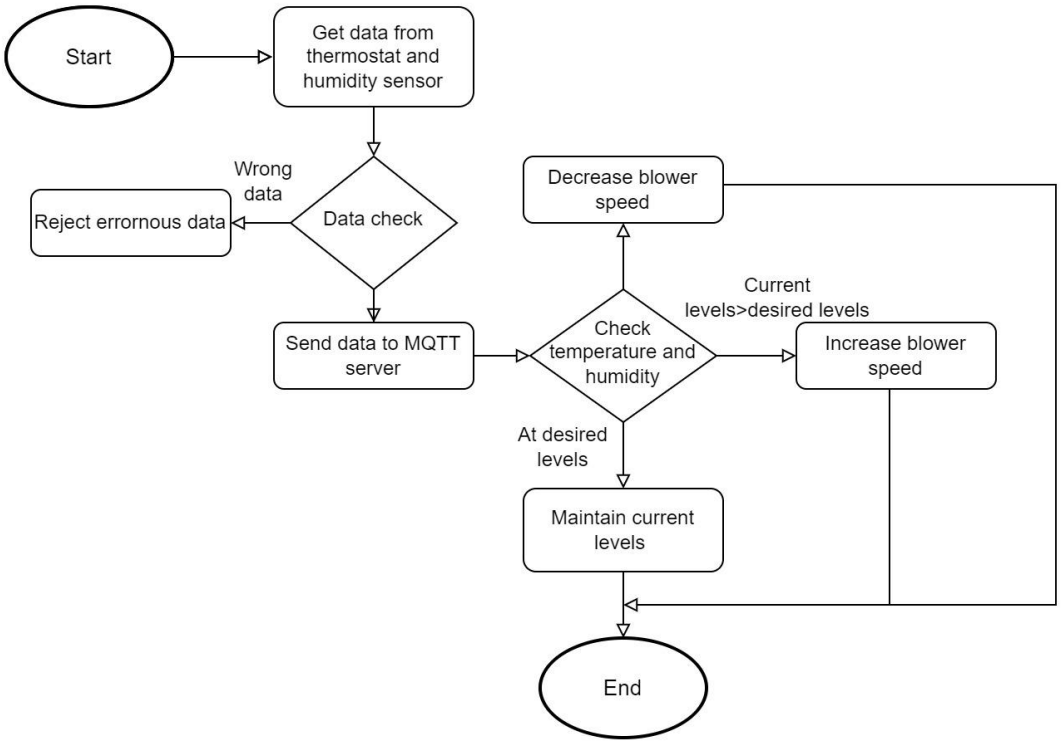
`326project/smartbuilding/hvac/<floorno>/<roomno>/temperature`
`326project/smartbuilding/hvac/<floorno>/<roomno>/pressure`




AHU will be operated according to the temperature and pressure in the room. The control command will be published to the following topic,

```
326project/smartbuilding/hvac/<floorno>/<roomno>/control/ahu
```

If the temperature is smaller than the desired temperature range, the ratio will be increased by 0.1 and published. And if the temperature is greater than than the desired range the ratio will be decreased by 0.1. Otherwise there is no change in the ratio.
If the pressure in the room is smaller than the desired pressure range, the blower speed will be increased by 1 in the AHU. If room pressure is greater than the desired range the blower speed will be decreased by 1. Otherwise there is no change in the blower speed.




**The above operations will be done until the room temperature and the pressure meet with the threshold values.**

The control command to the AHU contains time, speed and ratio.

Additionally we have subscribed to the following topics to change the threshold values.

```
326project/smartbuilding/hvac/<floorno>/<roomno>/control/temp-thresh
326project/smartbuilding/hvac/<floorno>/<roomno>/control/pressure-thresh
```

# MQTT Communication

The MQTT communication will utilise the following topics for transmitting the sensor data as well as for control signal propagation and SCADA.

## Sensor Data

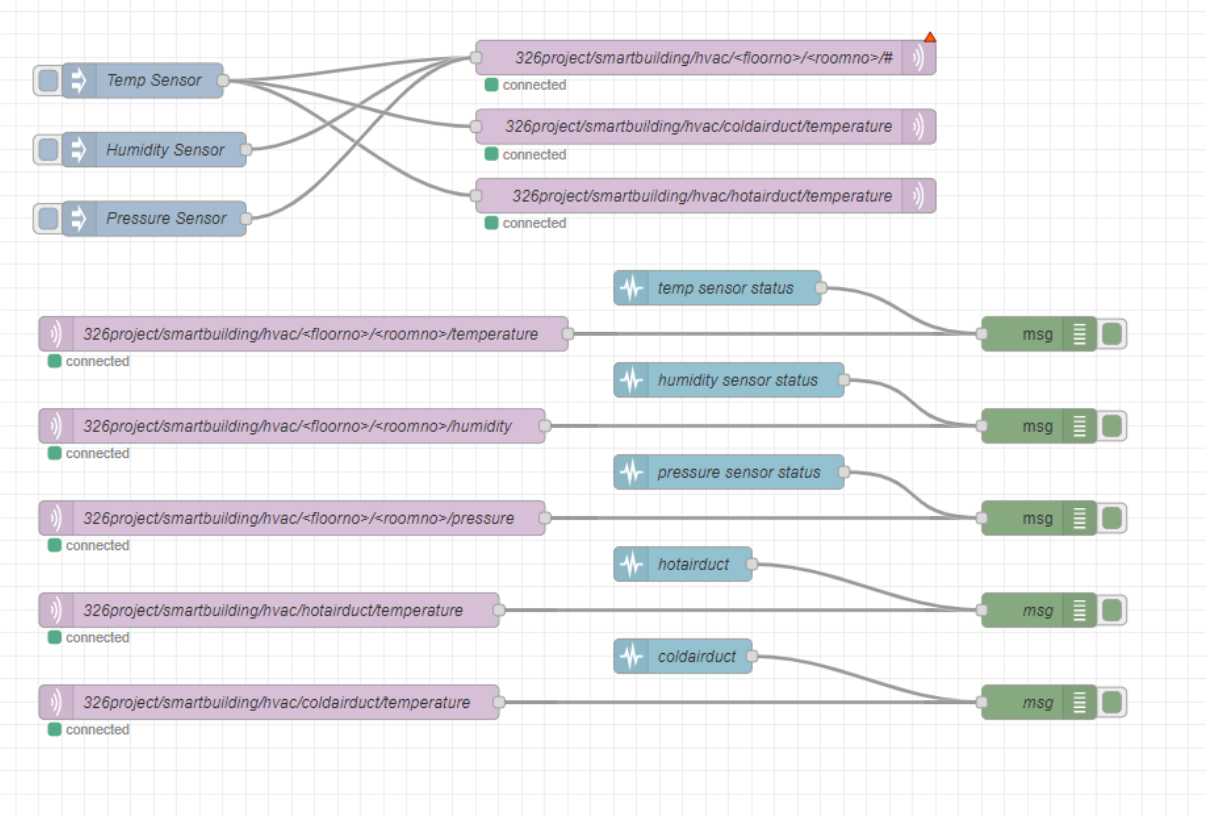
The following topic will contain all the information from the sensors subdivided into the type of sensor as well as the climate control zone that the sensor is in. Additionally, sensor data will be obtained from the occupancy topics to optimize the efficiency of the HVAC system.

```
Boiler/chiller and main control/acquisitions
      326project/smartbuilding/hvac/coldairduct/temperature
      326project/smartbuilding/hvac/hotairduct/temperature
```

```
Ventilation control in rooms
 Sensing
  326project/smartbuilding/hvac/<floorno>/<roomno>/temperature
  326project/smartbuilding/hvac/<floorno>/<roomno>/humidity
  326project/smartbuilding/hvac/<floorno>/<roomno>/pressure
  326project/smartbuilding/occupancy/<floorno>/<roomno>/count
```

Some example messages to illustrate the data format for the sensors are shown below.

| Sensor | Data Format | Example Message |
|---|---|---|
| Temperature | `time`: Time of sensor reading<br>`temp`: Temperature reading in Celsius | `{ “time”: “2022-04-11T14:15:48.245Z”, “temp”: 26.49 }` |
| Humidity | `time`: Time of sensor reading<br>`humid`: Relative humidity reading in % | `{ “time”: “2022-04-11T14:19:12.376Z”, “humid”: 0.63 }` |





```
msg : Object
▾ object
  ▾ status: object
      fill: "green"
      shape: "dot"
      text: "node-red:common.status.connected"
    ▾ source: object
        id: "182e533af2893996"
        type: "mqtt in"
        name: "326project/smartbuilding/hvac/coldairduct/temperature"
    _msgid: "2f82b68d86575ff0"
```


## Control

The following topics will be used to send control signals to the primary actuators of the HVAC system. The AHUs for each climate control zone will have a dedicated topic in order to control their environments independently. Since the boiler and chiller are common to the entire system they will operate through single topics for each actuator.

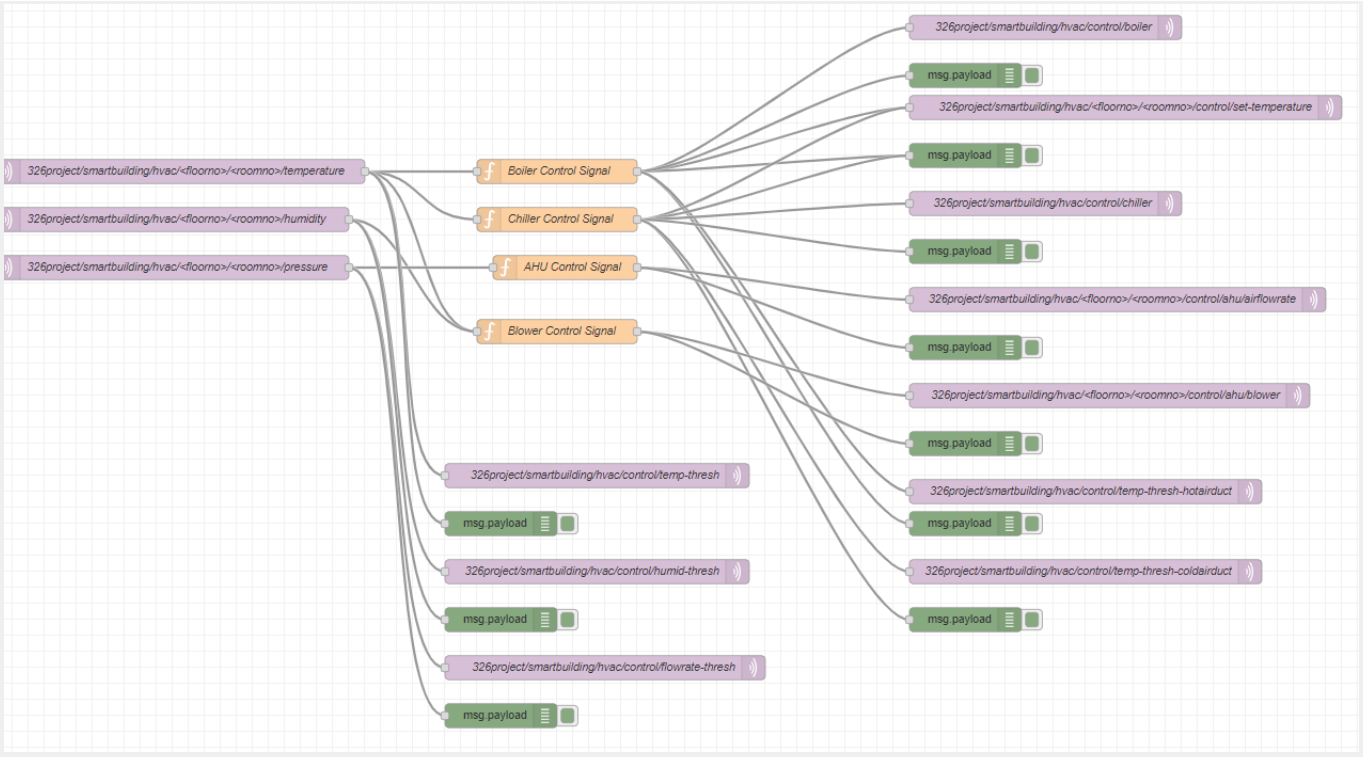
```
Boiler/chiller and main control/acquisitions
    Control
      326project/smartbuilding/hvac/control/boiler
      326project/smartbuilding/hvac/control/chiller
```

```
Ventilation control in rooms
 Control
 *326project/smartbuilding/hvac/<floorno>/<roomno>/control/set-temperature
  326project/smartbuilding/hvac/<floorno>/<roomno>/control/ahu/blower
 *326project/smartbuilding/hvac/<floorno>/<roomno>/control/ahu/airflowrate
```

```
set thresholds
      326project/smartbuilding/hvac/control/temp-thresh
      326project/smartbuilding/hvac/control/temp-thresh-coldairduct
      326project/smartbuilding/hvac/control/temp-thresh-hotairduct
      326project/smartbuilding/hvac/control/humid-thresh
      326project/smartbuilding/hvac/control/flowrate-thresh
```

A few examples of the data formats for control messages are shown below.

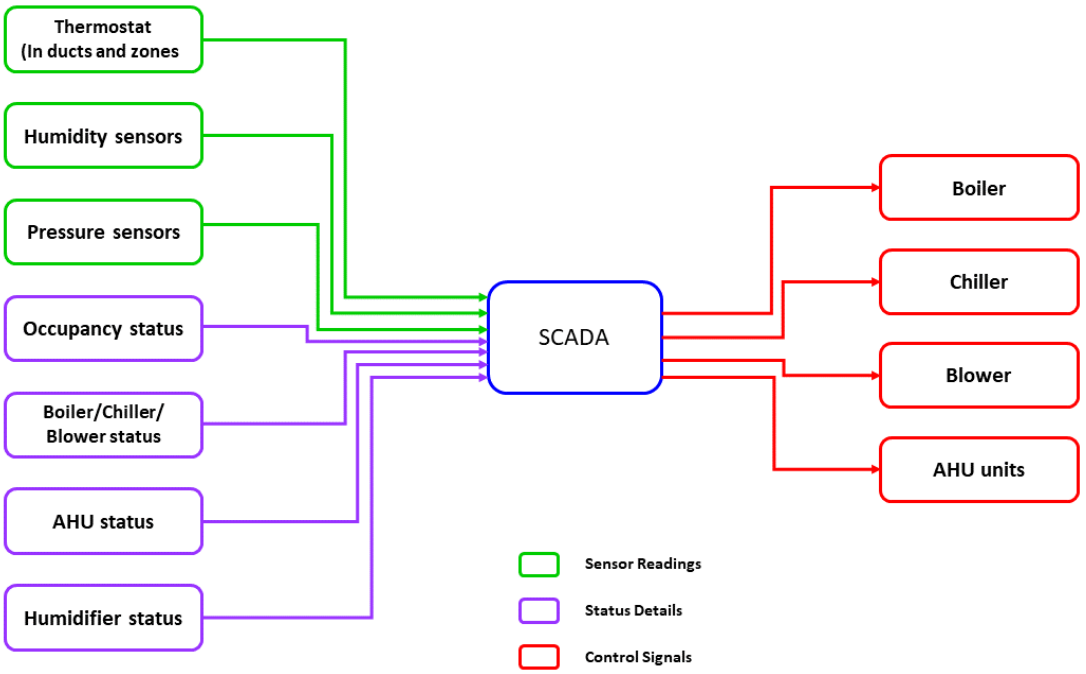
| Controller | Data Format | Example Message |
|---|---|---|
| Boiler/Chiller | `time`: Time of sensor reading<br>) state of the Boiler/Chiller | `{`<br>`  “time”: “2022-04-11T14:15:48.245Z”,`<br>`  “state”: 1`<br>`}` |
| AHU | `time`: Time of sensor reading<br>`speed`: Blower speed in %<br>`ratio`: Hot/Cold air mix ratio (0.0 means only cold air, 1.0 means only hot air) | `{`<br>`  “time”: “2022-04-11T14:19:12.376Z”,`<br>`  “speed”: 50`<br>`  “ratio”: 0.3`<br>`}` |



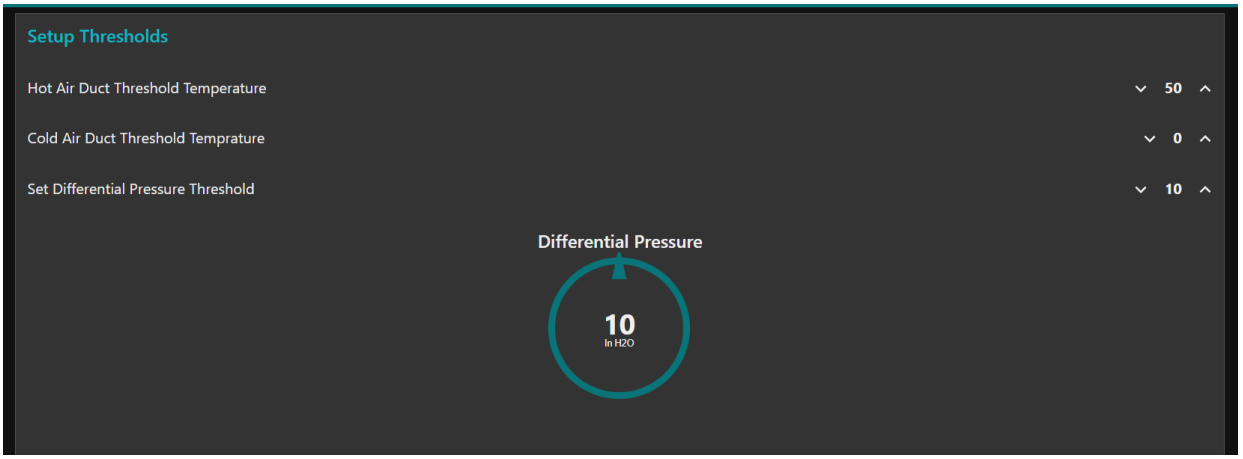
# SCADA

## Dashboard(E/17/176)

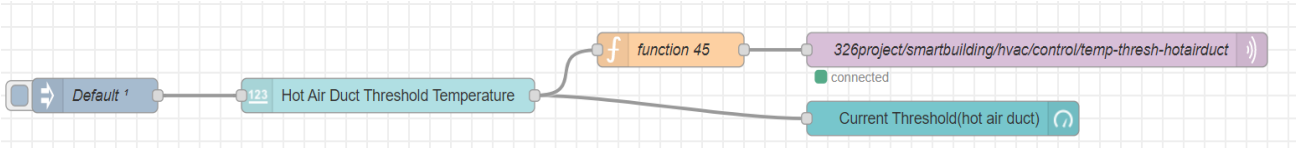
A SCADA interface is used to display the sensor readings, status of actuator that belong to the HVAC system. Also, there are some system controls which are provided through the SCADA interface. The diagram below shows a high level overview of the features provided in the SCADA interface.



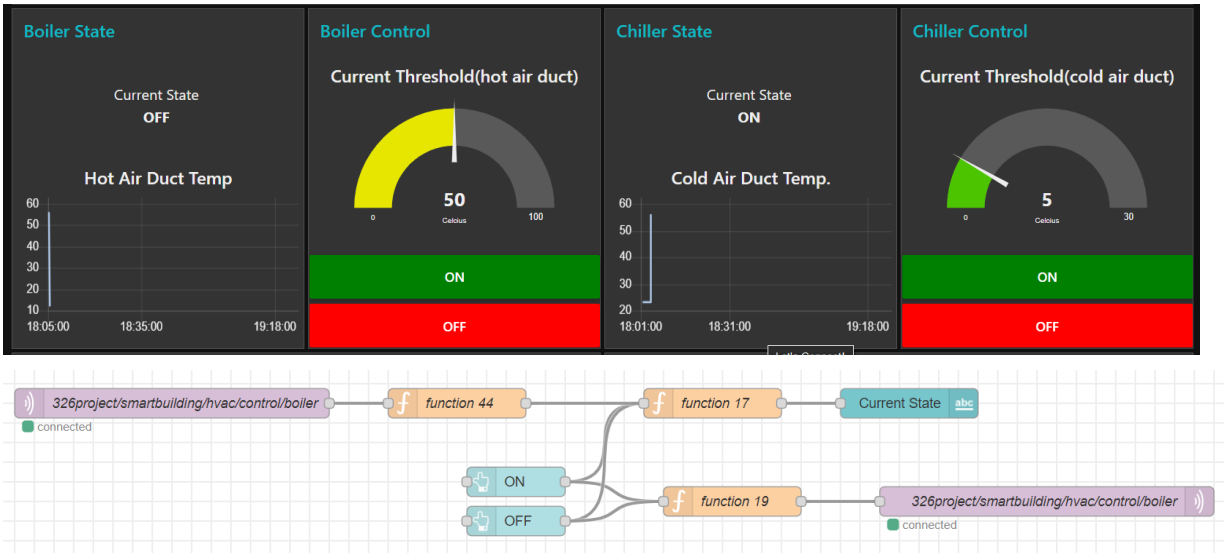

System administrators can set the threshold values such as cold air duct threshold and hot air duct threshold only through the SCADA interface. There is a separate group of inputs to set the threshold values. When a value change occurs in these inputs, the new value is published to relevant MQTT topics. Current threshold values for the system are displayed in the interface.



Following diagram shows the control signal flow of setting the threshold temperature value for the hot air duct temperature. There is a numerical input node to adjust the temperature value. The value is passed to a function node. In this function node the message is constructed in JSON format. Then the message is published to the topic to set the value. Same steps are followed to implement the other threshold setters.




Then the next main feature is, administrators can turn ON/OFF the boiler or the chiller manually from the interface. There is a specific section in the interface to show the current states of the chiller and the boiler. Also, the temperatures of the two main ducts are displayed in this section. The state of the boiler and chiller is observed by subscribing to the topics, to which the states of two actuators are published. Admins can see the variation of the temperatures of two ducts indicated in line charts. When an administrator pressed a button to turn ON/OFF the chiller or blower a control signal is published to relevant topics.



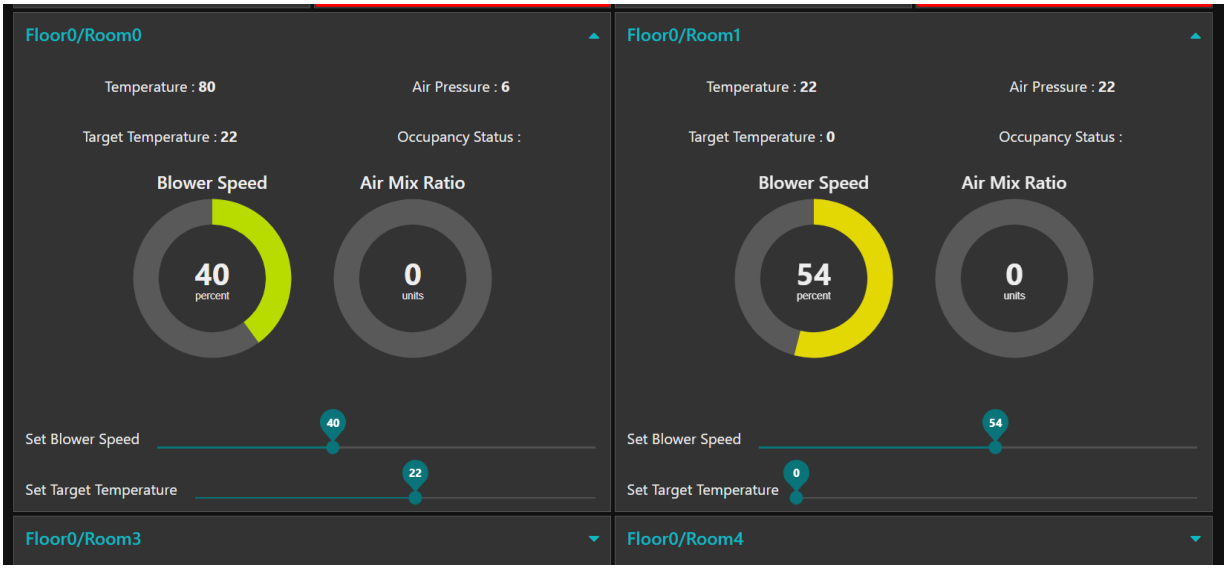

Above data flow is the data flow for boiler control. The state of the boiler is observed from the latest control signal that is published to the topic,

`326project/smartbuilding/hvac/control/boiler`

Two buttons are provided to ON/OFF. The function node(function 19), creates the message in relevant JSON format. Then the message is published to the boiler control topic to set the state. An identical data flow is used for the chiller control as well.

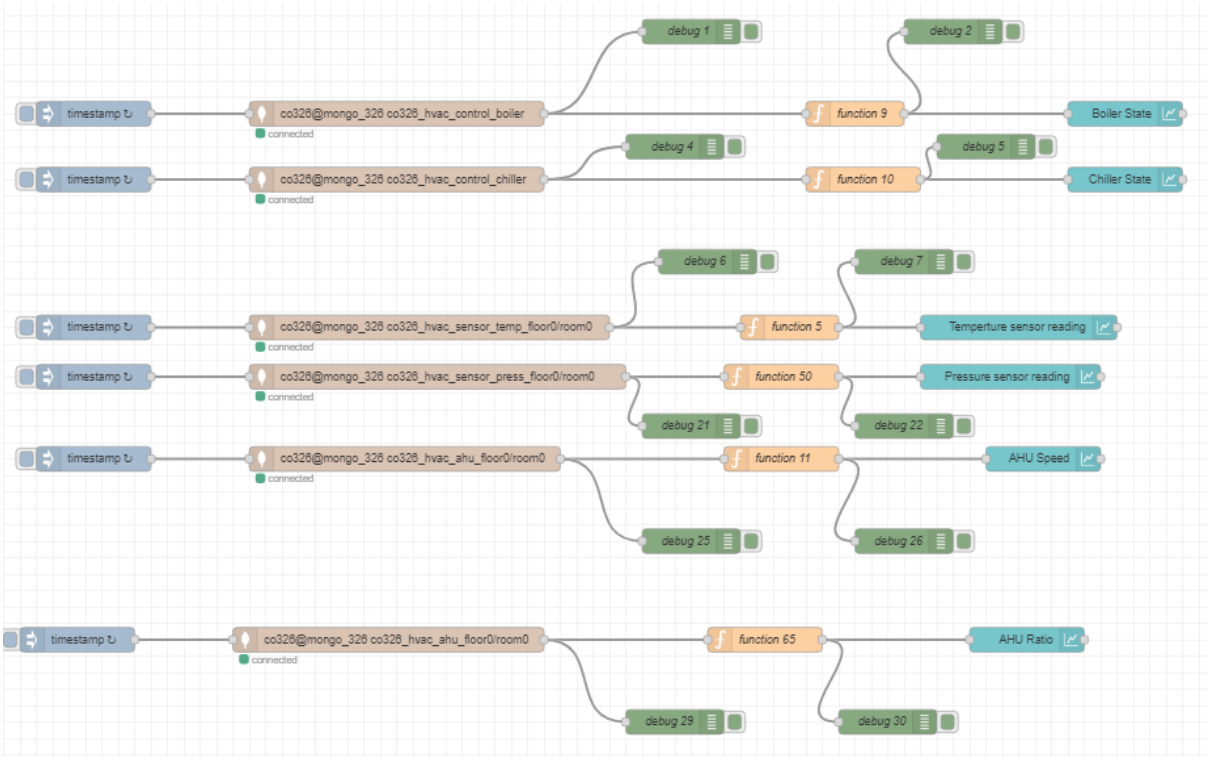
For each and every room in the building there is a separate group of UI components to display the current sensor readings of the room and the state of the AHU unit of the room. Also, there is an option provided for admins to control the blower speed and the target temperature of the room manually. Occupancy state of the room is observed by subscribing to the MQTT topic, to which the count of people

in the room is published. If the count is greater than zero, the room is considered as an occupied room.



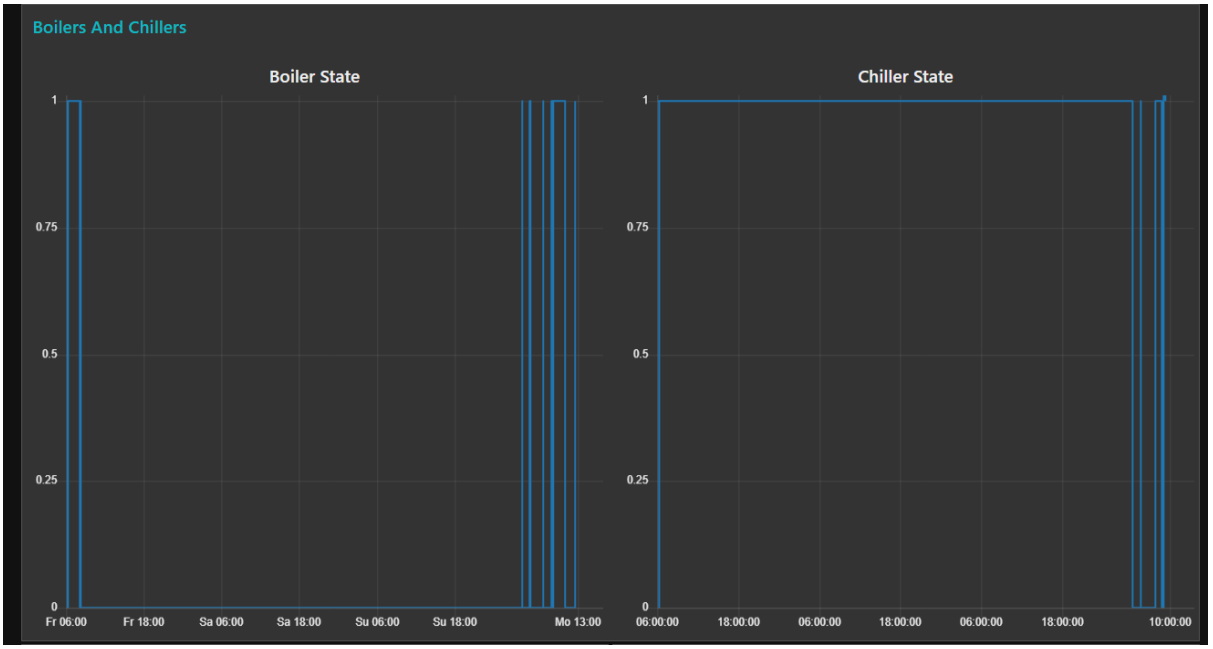

## Historical Data Visualisation (E/17/168)

States of Boiler and Chiller change from time to time. So those data are stored in the mongodb. Those data are imported and analysed.



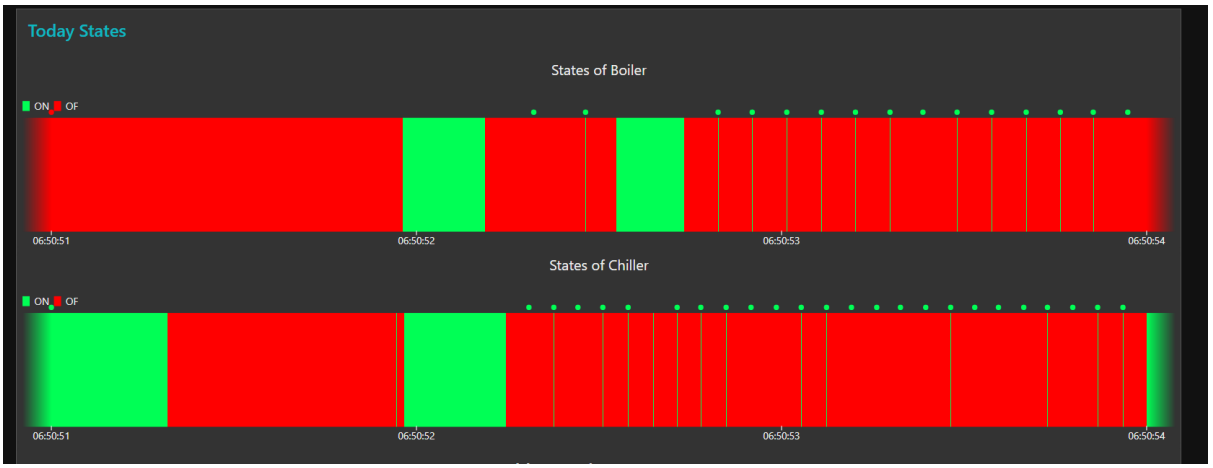

In the mongodb there are different number of data collections and lots of informations are stored as json files. Initially connects to the database and get data from each topic as message and Parse to a json object Then according to visualisation title information are grouped.

Then they are visualised in a graph. By observing graphs, can get historical data of states of Boilers and Chillers. There are two states and they are ON and OFF. In the graph 1 is indicating the ON state and 0 is indicating the OFF state.

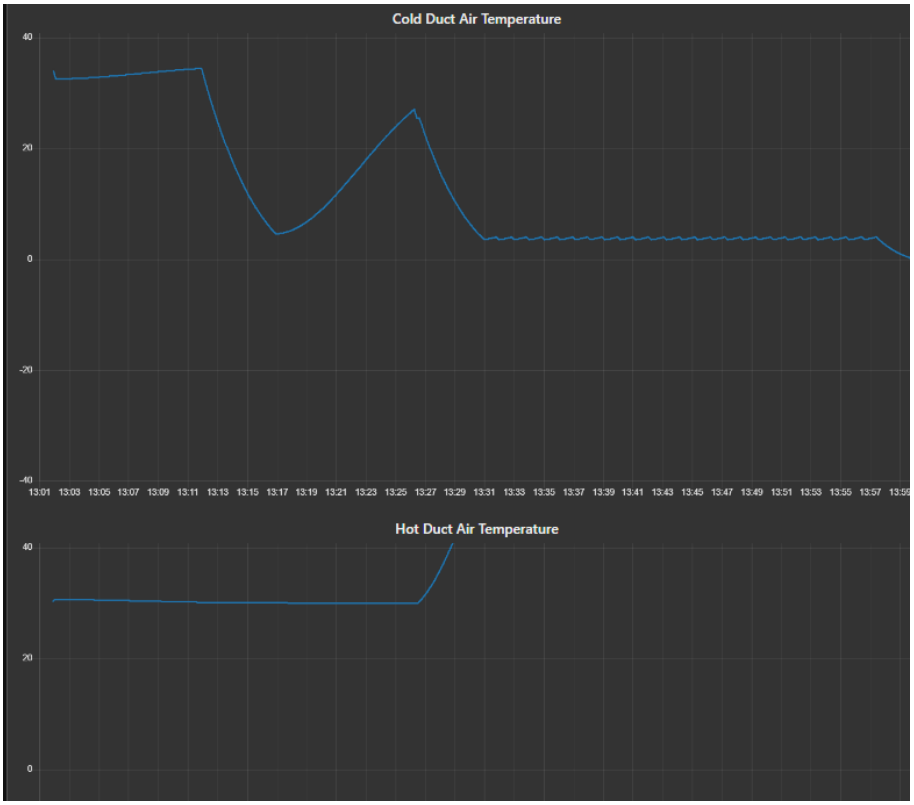



Also there are two floors and two rooms in the building. We are measuring Temperature, Pressure and Blower speed.

Also after analysing demonstrate the data related to the current date. First get data from the database and then according to the current date store all states of the boilers and chillers in an array with time stamps. This bar shows the timeline of the day.



So the red colour indicates the OFF state and Green colour indicates the ON state. There are main two segments as Boiler and Chiller and there shows in the container of Today States. Also there are other two graphs which show Cold Duct Temperature and Hot Duct Temperature.




For each roome there is a different container and in the top of the container there is the title of the room. There are two rooms in this interface and they are Room0 and Room1. In each container there are four main titles and they are

1. Temperature sensor reading
2. Pressure sponsor reading
3. AHU speed
4. AHU ratio

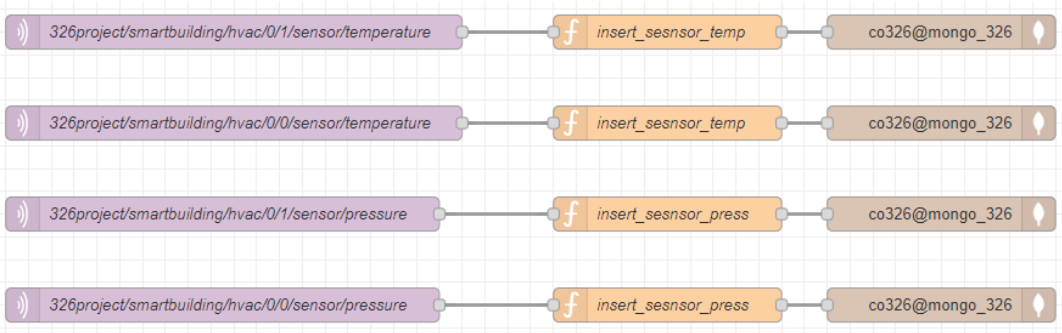
Those values are analysed and plotted against time. So this graphs are refresh in every second and then we can observe real time data.

# Storing MQTT Data, Commands and Events in the Database(E/17/091)


***Complete Node-RED work-flow***

When considering a room, basically there is a temperature sensor, a pressure and an air handling unit (AHU). The temperature sensor and the pressure sensor are used to measure room temperature and pressure respectively. Data from these sensors are published to following topics based on floor and room number.

- ***326project/smartbuilding/hvac/<floor number>/<room number>/temparature***
- ***326project/smartbuilding/hvac/<floor number>/<room number>/pressure***




***Node-RED work-flow for entering temperature & pressure sensor data***

Following python script is used to enter the published data into the database. Here, the values of 'floor', 'room' & 'description' variables are generated directly from the topic name. And the payload published to the topic is stored in the variable called 'data'. Moreover, the 'time' variable is used to store the current timestamp.
Finally these five variable values are used to override the existing payload and push it to the database.
Following MongoDB collections are used to insert temperature and pressure data based on room & floor number.

- **co326_hvac_temp_floor0/room0**
- **co326_hvac_temp_floor0/room1**
- **co326_hvac_press_floor0/room0**
- **co326_hvac_press_floor0/room1**

```
let str = msg.topic;
str = str.substring(30, str.length);

let floor = str.substring(0, 1);
let room = str.substring(2, 3);
let description = str.substring(4, str.length);
let data = msg.payload;
let time = new Date().toLocaleString();

msg.payload = {
    "floor": floor,
    "room": room,
    "description": description,
    "data": data,
    "time": time
};

msg.collection = "co326_hvac_sensor_temp_floor0/room1";

return msg
```

Control signals generated from the main controller unit to the AHU are stored in the same manner which is explained above.
Following MongoDB collections are used to store the control data, based on room & floor number.

- **co326_hvac_ahu_floor0/room0**
- **co326_hvac_ahu_floor0/room1**

The whole heating unit of the system consists of a boiler and a chiller and air ducts that are connected to them. The temperatures of the air ducts and the control signals to boiler and chiller are stored in the database as well. For storing those data the following python script is used.

```
let str = msg.topic;
str = str.substring(30, str.length);

let description = str.substring(0, str.length);
let data = msg.payload;
let time = new Date().toLocaleString();

msg.payload = {
    "description": description,
    "data": data,
    "time": time
};

msg.collection = "co326_hvac_coldAirduct_temp";

return msg
```

According to the python script the value of ‘description’ variables is generated directly from the topic name. And the payload published to the topic is stored in the variable called ‘data’. Moreover, the ‘time’ variable is used to store the current timestamp.
Finally these three variable values are used to override the existing payload and push it to the database.
Following MongoDB collections are used to insert hot & cold air duct temperature data and control signals to the boiler and the chiller.

- **co326_hvac_coldAirduct_temp**
- **co326_hvac_hotAirduct_temp**
- **Co326_hvac_control_chiller**
- **co326_hvac_control_boiler**

# Data Analysis(E/17/091)

The central processing unit of the HVAC collects data from the following sensors:

- Temperature sensors installed in each room
- Temperature sensors installed in hot & cold air ducts.
- Pressure sensors installed in each room.

It generates control commands based on that data to achieve the following functionalities:

- Regulate the pressure in each room via AHU air blower speed.
- Regulate the air mix ratio for each room via dampers in the AHU to control the temperature of the room.

When generating control commands they are stored in the database as well. We can use this data to create ML models and increase the efficiency of the HVAC. For example, training an ML model on historical data over a long period of time will allow us to identify trends in HVAC utilization such as peak times (i.e. times where the building is busiest during the day) and idle times to optimize the control algorithm for greater efficiency by predictively activating or deactivating the boiler or chiller in anticipation of expected thermal loads.