

Whole House Combined HVAC System Project Report 3

Judson A. Hartley

Fort Hays State University

CSCI441: Software Engineering

Author Note

Project website: <https://jahartley.github.io/CSCI441web/>

Email: j_hartley@mail.fhsu.edu

Abstract

This project is mostly about creating small independent sensors and controllers that can be leveraged to make a house's ventilation system work with its heating and air conditioning, and provide vastly improved indoor air quality and energy efficiency for its occupants. These sensors need not be expensive, or difficult to setup and use, and can be used to bring dramatic improvement to indoor air quality. By combining many datapoints collected by the sensors, data driven improvements and diagnostics can be added to existing equipment, and proper sizing of new equipment can be created.

Table Of Contents

Abstract	2
Table Of Contents	3
1. Customer Statement of Requirements	4
2. Glossary of Terms	7
3. System Requirements	8
4. Functional Requirements Specification	8
4.d. Use Cases	11
4.d.5 Traceability Matrix	14
4.d.7 System Sequence Diagram	18
5. Effort Estimation using Use Case Points	18
5.f. UCP result	23
6. Domain Analysis	23
6.a.4 Traceability Matrix	30
7. Interaction Diagrams	30
8. Class Diagram and Interface Specification	31
9. System Architecture and System Design	40
10. Algorithms and Data Structures	43
11. User Interface Design and Implementation	43
12. Design of Tests	44
13. History of Work, Current Status, and Future Work	45
14. References	47

Whole House Combined HVAC System

The acronym HVAC stands for Heating, Ventilation, and Air Conditioning. While almost all houses in the U.S have a "HVAC" system, usually there is minimal coordination between the "V" and the "H" and "AC" parts.

1. Customer Statement of Requirements

1.a. Problem Statement

In most houses, the HVAC system consists of a central heating and air conditioning system, and the ventilation consists of a few bathroom fans, and if you are lucky a kitchen range vent fan that actually blows outside. The average house's thermostat measures dry bulb air temperature only, which does not take humidity into account, and usually only at a single location. Does your house feel the same in all the rooms? What about actual air quality? When you run your bathroom vent fan, or your clothes dryer, it takes air from inside your house, and blows it outside. Your house doesn't deflate like a balloon, and clearly you don't run out of air inside your house, so that means your house is replacing the air that you blew out with air from outside. Where does it come in at? Is it filtered or otherwise conditioned? Does hot humid outside air in the summer suck in through your wall cavities, where the humidity condenses on the back side of your cool sheetrock, growing mold in your walls? As most houses do a fairly good job of containing the air they have inside, ever thought about your indoor air quality? Ever had a VOC (volatile organic compounds), CO2 or radon gas sensor running in your bedroom? Is there any way that modern inexpensive sensors could be used to make my house an environment that my family would be happy to live in and not be worried that they would be better off outside?

In human terms, our [thermal comfort](#) is in large part a consequence of not just the temperature of the surrounding air, but (because we cool ourselves via perspiration) the extent to which that air is

saturated with water vapor. (<https://en.wikipedia.org/wiki/Psychrometrics>) Most current HVAC systems do not measure for nor take into account psychrometric data that more closely models how you feel inside. How often do you feel cold or hot in your house even though the air conditioning or heat is at the same temperature? If your system used psychrometric values derived from simple air measurements then it would move the temperature set point automatically so that the air feels the same all the time. Additional input data like room stratification, where the temperature at the floor is different from the ceiling, can be measured and used for intelligent ceiling fan control or zoned circulation while heating or cooling modes are off or there is minimal heating or cooling load as experienced in the fall or spring. All of the system functions described above do not require additional HVAC equipment over what is installed in the majority of U.S. houses, only additional sensors and better control strategies.

There are no systems that I am aware of that try to account for outgoing air from all sources and proactively bring in fresh air that has been filtered and conditioned. Some high-performance house builds implement ventilation via Energy Recovery Ventilators (ERV) or Heat Recovery Ventilators (HRV), but often operate at a steady low volume all the time, with no considerations of room usage, wasting energy. Most gas sensors can be had inexpensively and can be used to monitor air quality at a room-by-room level. There is no need to ventilate an empty bedroom for months on end, and it would not be very difficult to have room occupancy taken into effect. By having the data available to view, even something as simple as leaving the door open at night can be tried to see if an actual difference is achieved.

The goal is to use relatively inexpensive sensors to create a home HVAC monitoring and control system, that enables reduced energy consumption overall, vastly improved user comfort, and providing a means for proper equipment sizing during future system upgrades when HVAC equipment wears out. Additional features like system diagnostics and performance can be measured and evaluated, along with room-by-room data as a user grows their system.

1.b. Decomposition into Sub-problems

This project is made up of a few subprojects, listed below:

1. Sensor Boards. These boards will send measured data to be collected and used by the system.
This system will be implemented with STM32 (ethernet) or ESP32 (WIFI) microcontrollers, with firmware written in C++.
2. Control Boards. These boards will allow the system to control of the shelf HVAC equipment, via commands sent over the network, and return command acknowledgements and system status data for collection. This system will be implemented by the same types of microcontrollers as above.
3. System Network. This system needs a network over which collected data, processed data, control instructions, and user interfaces will be sent. This system will be implemented with of the shelf ethernet/WIFI networking, and messaging via MQTT (<https://mqtt.org/>) messages.
4. Data Processing. Although most HVAC systems collect basic measurements, and often only one, this system will be designed to collect enough measurements to allow for extended data points and mathematical models to be calculated for use in system decision making. This system will be implemented with NodeJS and JavaScript.
5. Data Storage. To enable review and system optimizations, either by human review or machine learning, it is very helpful to have recorded data to work with. This system will retain data for use later. This system will be implemented using InfluxDB (<https://www.influxdata.com/>).
6. User Interface. This system is the main way that users will interact with the overall system. I plan on using Home Assistant (<https://www.home-assistant.io/>) for this system, which provides both short term history viewing and many different built in ways to control the system.

7. Long Term Data Review. This system lets a user review and compare data across all system data points, over long periods of time. This system will be implemented with Grafana (<https://grafana.com/>).

2. Glossary of Terms

- Psychrometrics (psychrometry) - the field of engineering concerned with the physical and thermodynamic properties of gas-vapor mixtures. In an HVAC application we are mainly interested in the effects of water vapor (humidity) on system operation. ('Psychrometrics', 2021)
- MQTT (<https://mqtt.org/>) - A lightweight publish/subscribe messaging transport protocol ideal for connecting remote devices with a small code footprint and minimal network bandwidth.
- Energy Recovery Ventilators (ERV) – A system of ductwork, fans, and a heat exchanger that uses outgoing air from the house to condition incoming replacement air by transferring heat energy and humidity without mixing the incoming and outgoing airstreams.
- Heat Recovery Ventilators (HRV) - A system of ductwork, fans, and a heat exchanger that uses outgoing air from the house to condition incoming replacement air by transferring heat energy ONLY without mixing the incoming and outgoing airstreams.
- STM32 – A family of very inexpensive 32-bit microcontrollers by STMicroelectronics (<http://www.st.com>) all based on versions of ARM's Cortex M processor cores.
- ESP32 – A family of inexpensive 32-bit microcontrollers by Espressif Systems (<https://www.espressif.com/en/products/socs/esp32>) that include built-in WIFI and Bluetooth radios and TCP/IP Stacks for wireless ethernet connectivity.

3. System Requirements

3.a. Business Goals

Overall: Create an ecosystem of user-friendly sensors that create a home environmental monitoring system that evolves into a complete system over time, and at the customer's pace.

1. Create a collection of accurate but inexpensive small sensor boards to interface with the data collection system over the system network.
2. Create a collection of control boards to interface off the shelf HVAC equipment with the system network.
3. Create a set of off the shelf recommendations for small networking equipment to connect all the sensor boards together.
4. Create a software stack that is very easy to deploy on a wide array of hardware, preferably using open-source software.

4. Functional Requirements Specification

4.a. Functional Requirements

Initial work will be in sub-problems 1. Sensor Boards, 3. System Network, 4. Data Processing, 5. Data Storage, and 6. User Interface.

Requirement priorities are denoted by:

- E – essential, needed for base system operation.
- D – desirable, highly desirable but not mandatory
- O – optional, may be realized if time and resources permit
- F – future, will not be realized in this version, but should be considered in future versions.

4.a.1. Enumerated Functional Requirements for Sensor Boards.

Identifier	Priority	Requirement
------------	----------	-------------

REQ-1.1	E	Create sensors that can measure an array of environmental variables.
REQ-1.2	E	Have sensor data sent to MQTT broker.
REQ-1.3	E	Connect via ethernet
REQ-1.4	F	Connect via WIFI

4.a.3. Enumerated Functional Requirements for System Network

Identifier	Priority	Requirement
REQ-3.1	E	Use off the shelf network equipment.
REQ-3.2	E	System needs one server for software stack (any small computer will be capable).

4.a.4. Enumerated Functional Requirements for Data Processing

Identifier	Priority	Requirement
REQ-4.1	E	Subscribe to sensors over MQTT.
REQ-4.2	E	Take air data and create psychrometric data, and send back to MQTT.
REQ-4.3	F	Take data from many sources and create aggregated values (i.e. take computed feels like temperature from all rooms with sensors, and combine with room size weighting to make a single overall temperature value for the control system to act upon).
REQ-4.4	F	Take air data and power usage data from a heat pump/air conditioning source and generate Coefficient Of Performance (COP - efficiency) data.
REQ-4.5	F	Take air handler pressure data and generate air filter load and change requirements.
REQ-4.6	F	Take room air data and outside air data to generate ventilation needs.
REQ-4.7	F	Take room air data and create room-by-room heat and cooling loads.
REQ-4.8	D	Take sensor data and create human readable MQTT packets.

4.a.5. Enumerated Functional Requirements for Data Storage

Identifier	Priority	Requirement
REQ-4.1	E	Receive data from Home Assistant.
REQ-4.2	E	Keep data for extended periods of time.

4.a.6. Enumerated Functional Requirements for User Interface

Identifier	Priority	Requirement
------------	----------	-------------

REQ-6.1	E	Subscribe to MQTT data.
REQ-6.2	E	Send data to InfluxDB.
REQ-6.3	E	Save a short timeframe of data locally (one month).
REQ-6.4	E	Setup dashboards to view real time data with logical organization.
REQ-6.5	F	Setup control dashboards for setting user input, i.e. threshold temperature and system operation modes.

4.b. Enumerated Nonfunctional Requirements

Nonfunctional requirements are grouped by the same numbering convention as above, for sub-problems: 1. Sensor Boards, 3. System Network, 4. Data Processing, 5. Data Storage, and 6. User Interface.

4.b.1. Enumerated Nonfunctional Requirements for Sensor Boards

Identifier	Priority	Requirement
REQ-1.5	D	Use the same firmware across same board types (i.e. all STM32 use firmware with most sensor types available, and set by configuration file)
REQ-1.6	D	Use SD Card to set device configuration (i.e. IP address, MQTT address, id number, what sensors to use.)
REQ-1.7	D	Keep device configuration in plain text to ease configuration
REQ-1.8	D	Measure air temperature using PT100/MAX31865 sensors
REQ-1.9	F	Measure air temperature using DS18B20 sensors
REQ-1.10	D	Measure air temperature/humidity using SHT40 sensors
REQ-1.11	D	Measure air pressure difference using MXP10DP/AD620 sensor
REQ-1.12	F	Measure air pressure/humidity using BMP280 sensor
REQ-1.13	F	Measure air pressure/humidity/quality using BME680 sensor
REQ-1.14	F	Measure air flow using automotive hot film mass airflow sensor
REQ-1.15	F	Measure refrigerant temperature using PT100/MAX31865 sensors
REQ-1.16	F	Measure refrigerant pressure using automotive refrigerant pressure transducer
REQ-1.17	F	Measure HVAC equipment power consumption
REQ-1.18	F	Measure CO2 levels
REQ-1.19	F	Measure VOC levels
REQ-1.20	F	Measure NH3 levels
REQ-1.21	F	Measure H2S levels
REQ-1.22	O	Blink error codes via onboard LED

4.b.3. Enumerated Nonfunctional Requirements for System Network

Identifier	Priority	Requirement
REQ-3.3	D	Use power over ethernet POE to power sensor boards
REQ-3.4	F	Use system server to bridge ethernet and WIFI networks.

4.b.4. Enumerated Nonfunctional Requirements for Data Processing

Identifier	Priority	Requirement
REQ-4.9	F	Use Data Processing to host a webpage for system setup
REQ-4.10	F	Use Data Processing to generate sensor board config files for the User.

4.b.5. Enumerated Nonfunctional Requirements for Data Storage

Identifier	Priority	Requirement
N/A		

4.b.6. Enumerated Nonfunctional Requirements for User Interface

Identifier	Priority	Requirement
REQ-6.6	F	Create dashboards for Kiosk devices (i.e. using an amazon fire tablet in a wall mount to view live data and make control choices)

4.c. User Interface Requirements

Identifier	Priority	Requirement
REQ-6.7	F	Easy to operate customizable controls
REQ-6.8	F	Easy to operate system thresholds (air temperature setting)
REQ-6.9	D	Easy to view system history

4.d. Use Cases

4.d.1. Stakeholders

- Home Users
- System Upgrade Designers or Monitoring Business

4.d.2. Actors and Goals

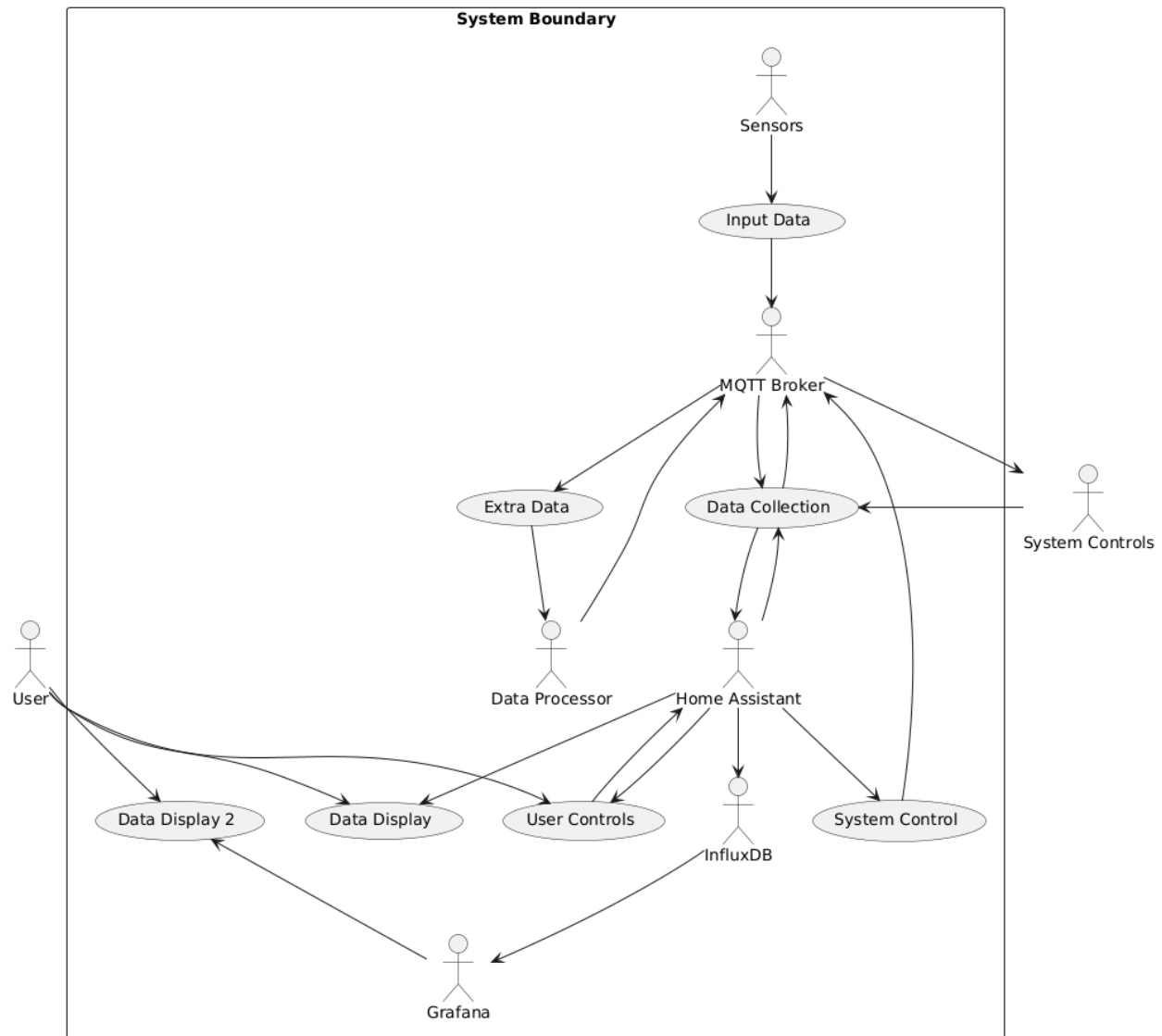
- Sensors (sub-problem 1.) Sensors create the inputs that the system acts upon, and records. They are an Initiating actor in this system.
- MQTT broker (sub-problem 3.) Receives messages and retransmits to all subscribers for that message.
- Data processors (sub-problem 4.) Data processors respond to sensors or other inputs in an event driven fashion, and process new output data from the input sources. A participating actor.
- System Controls (sub-problem 2.). System controls take data, and make system decisions for equipment modes and operating conditions. System controls should also be able to function at a minimal level without any other part of the system operating, and are therefore Initiating actors
- Users. Users can set system modes and thresholds, that the system controls use in their decision process. Users/Administrators also configure and install all system equipment, Initiating actors.

4.d.3. Casual Description

Actor	Actor's Goal	Use Case Name
Sensor board	Provide measured environmental variables for use by all other systems.	Input Data UC-1
MQTT broker	Distributes messages across the system	Acts in UC-1, UC-2, UC-3, UC-4
Data Processor	Provide additional derived data from the sensor input needed for better system operation decisions.	Extra Data UC-2
System Controls	Get filtered data, system modes, and compare against thresholds to make system operating decisions.	System Control UC-3
User	Makes changes to thresholds and modes to control system and get the desired environment.	User Controls UC-4
InfluxDB	To collect and retain all data for review and future system upgrade decision making.	Data Collection UC-5
User	View collected data short term history.	Data Display UC-6

User	View long term history	Data Display 2 UC-7
Home Assistant	Collects data for display, retransmits to InfluxDB for long term collection, contains the User Controls and short term History.	Acts in UC-4, UC-5, UC-6
Grafana	Displays long term data in any user specified manner	Acts in UC-7

4.d.4. Use Case Diagram



4.d.5. Traceability Matrix

For the traceability matrix, the priority codes are translated as follows:

- E – essential, Weight 5
- D – desirable, Weight 4
- O – optional, Weight 2

- F – future, Weight 0

REQ	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7
REQ-1.1	5	X						
REQ-1.2	5	X						
REQ-1.3	5	X						
REQ-1.4	0	X						
REQ-1.5	4	X						
REQ-1.6	4	X						
REQ-1.7	4	X						
REQ-1.8	4	X	X					
REQ-1.9	0	X	X					
REQ-1.10	4	X	X					
REQ-1.11	4	X	X					
REQ-1.12	0	X	X					
REQ-1.13	0	X	X					
REQ-1.14	0	X						
REQ-1.15	0	X						
REQ-1.16	0	X						
REQ-1.17	0	X						
REQ-1.18	0	X						
REQ-1.19	0	X						
REQ-1.20	0	X						
REQ-1.21	0	X						
REQ-1.22	2	X						
REQ-3.1	5	X	X	X				
REQ-3.2	5	X	X	X				
REQ-3.3	4	X	X					
REQ-3.4	0	X	X					
REQ-4.1	5		X					

REQ-4.2	5		X	X				
REQ-4.3	0		X					
REQ-4.4	0		X					
REQ-4.5	0		X					
REQ-4.6	0		X					
REQ-4.7	0		X					
REQ-4.8	4		X					
REQ-4.9	0		X					
REQ-4.10	0		X					
REQ-5.1	5					X		X
REQ-5.2	5					X		X
REQ-6.1	5	X	X	X			X	
REQ-6.2	5	X	X	X		X		X
REQ-6.3	5	X	X	X			X	
REQ-6.4	5	X	X	X			X	
REQ-6.5	0			X	X		X	
REQ-6.6	0	X	X	X	X		X	
REQ-6.7	0				X			
REQ-6.8	0				X			
REQ-6.9	4	X	X	X	X		X	
Max PW		5	5	5	4	5	5	5
Total PW		79	64	39	4	15	19	15

4.d.6. Fully-Dressed Description

Use Case UC-1:	Input Data
Related Requirements:	REQ-1.1-22, REQ-3.1-4, REQ-6.1-4, REQ-6.6, REQ-6.9
Initiating Actor:	Sensor Board
Actor's Goal:	Measure and send measurement data to the MQTT broker in the software stack.
Participating Actors:	MQTT Broker, Data Processor, Home Assistant, InfluxDB
Preconditions:	Correctly configured SD Card is installed, Board has correct sensor platform attached, Board is connected via POE Ethernet to a switch on the same network as the software actors.
Postconditions:	N/A
Flow of Events for Main Success Scenario: → 1. Sensor Board receives power and begins executing firmware → 2. Sensor Board reads configuration file from SD Card → 3. Sensor Board sets environment variables from config file	

- 4. Sensor Board initializes ethernet connection with config settings
- 5. Sensor Board connects to MQTT broker
- 6. Sensor Board sends heartbeat message to MQTT broker (all other systems subscribed to heartbeat messages will receive)
- 7. Sensor Board reads first configured sensor
- ← 8. Sensor Board sends first sensor data to MQTT broker (all other systems subscribed to the sensor topic will now receive a copy of this sensor data and perform their own tasks as required)
- 9. Sensor Board reads Nth configured sensor
- ← 10. Sensor Board sends Nth configured sensor data to MQTT broker (all other systems subscribed to the sensor topic will now receive a copy of this sensor data and perform their own tasks as required)
- 11. Sensor Board has reached end of configured sensor list
- 12. Sensor Board waits for configured delay time
- 13. Sensor Board repeats from step 6 to 12 until power is lost

Flow of Events for Extensions (Alternate Scenarios):

2a. Sensor Board cannot read SD Card

← 2a. Sensor Board blinks error code via onboard LED

- 1. Other systems heartbeat message timer expires and alerts users/administrators.

3a. SD Card has incorrect configuration file (same as 2a. above)

4a. Sensor Board cannot initialize ethernet connection (same as 2a. above)

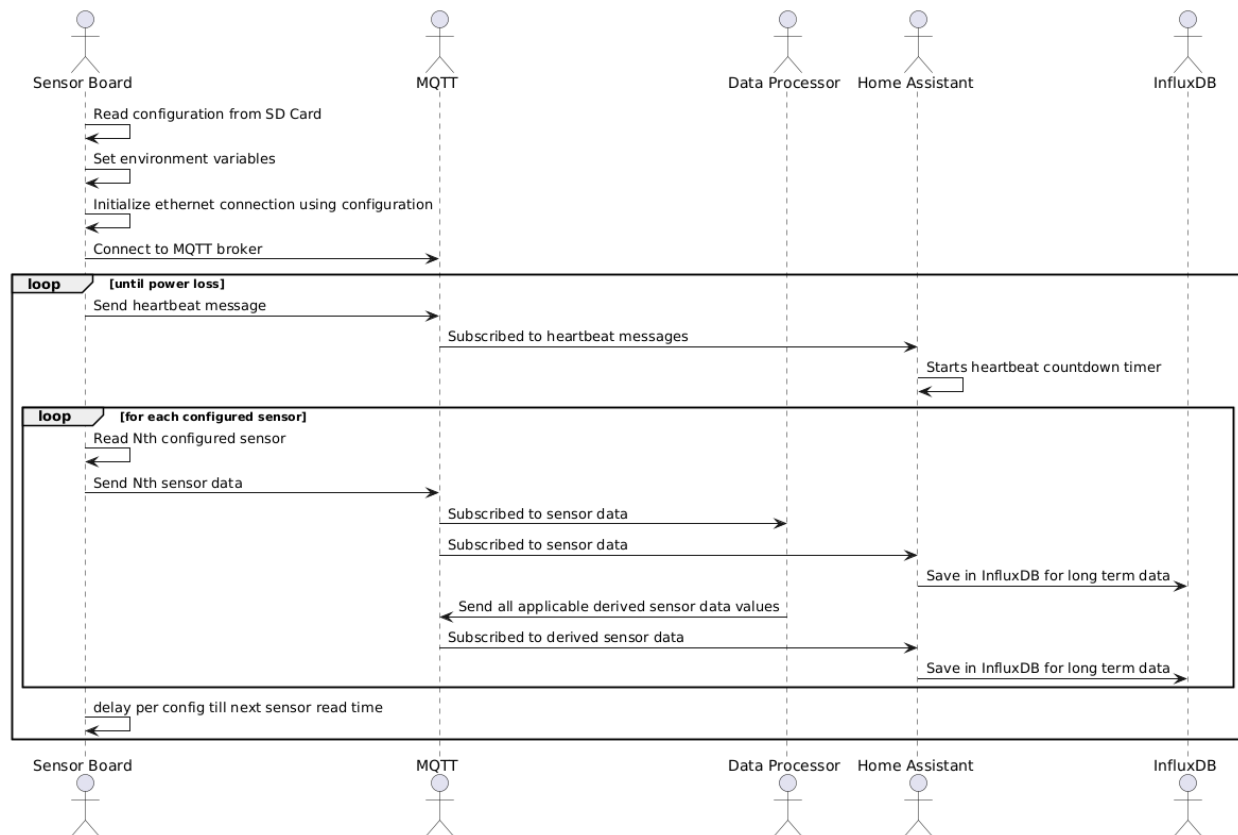
5a. Sensor Board cannot connect to MQTT broker (same as 2a. above)

7a./9a. Sensor Board cannot read expected sensor

← 7a./9a. Sensor Board sends error code via MQTT in heartbeat message.

- 1. Other systems respond to heartbeat message error code and alert users/administrators.

4.d.7. System Sequence Diagrams



4.e. User Interface Specifications

User interfaces for this system are handled through Home Assistant, which can have “Packages” of user interface added via YMAL configuration files, which this project will generate for the User, and via Grafana graphs for long term data.

5. Effort Estimation using Use Case Points

Use case points are calculated using the following equation:

$$UCP = UUCP * TCF * ECF$$

Where: UUCP is Unadjusted Use Case Points, the complexity of the functional requirements.

TCF is the Technical Complexity Factor, which measures the complexity of the nonfunctional requirements.

ECF is the Environment Complexity Factor, which assesses the development team's experience and their development environment.

The UUCP is the sum of two components:

UUCP = Unadjusted Actor Weight (UAW) + Unadjusted Use Case Weight (UUCW)

5.a. UAW calculation

Actor name	Description of relevant characteristics	Complexity	Weight
Sensor board	Provide measured environmental variables for use by all other systems.	Average	2
MQTT broker	Distributes messages across the system	Average	2
Data Processor	Provide additional derived data from the sensor input needed for better system operation decisions.	Average	2
System Controls	Get filtered data, system modes, and compare against thresholds to make system operating decisions.	Average	2
User	Makes changes to thresholds and modes to control system and get the desired environment, views collected data short term history, views long term history.	Complex	3
InfluxDB	To collect and retain all data for review and future system upgrade decision making.	Simple	1
Home Assistant	Collects data for display, retransmits to InfluxDB for long term collection, contains the User Controls and short term History.	Simple	1
Grafana	Displays long term data in any user specified manner	Simple	1

UAW(Combined HVAC) = 3 * Simple + 4 * Average + 1 * Complex = 3*1 + 4*2 + 1*3 = 14

5.b. UUCW calculation

Use Case	Description	Category	Weight
UC-1: Input Data	Sensor boards that send data to the system, 4 concepts	Simple	5

UC-2: Extra Data	System that takes sensor data and makes human readable values, filters, and generates derived data, 4 concepts	Simple	5
UC-3: System Controls	System controls use user set thresholds and user set modes, along with filtered sensor data or derived data to generate HVAC equipment commands, 9 concepts	Average	10
UC-4: User Controls	Control interfaces for user interaction, 6 concepts	Average	10
UC-5: Data Collection	Saves all data for future analysis, 3 concepts	Simple	5
UC-6: Data Display	Displays short term data, 2 concepts	Simple	5
UC-7: Data Display 2	Displays long term data, 2 concepts	Simple	5

$$\text{UUCW}(\text{combined HVAC}) = 5 * \text{Simple} + 2 * \text{Average} + 0 * \text{Complex} = 5*5 + 2*10 + 0*15 = 45$$

5.c. UUCP calculation

$$\text{UUCP} = \text{UAW} + \text{UUCW} = 14 + 45 = 59$$

5.d. TCF calculation

Perceived complexity has been selected on a 1 to 5 scale, with 3 being the averaged expected complexity, and 0 indicating that this project does not have this technical factor. The constants are:

$$\text{Constant-1 (C1)} = 0.6$$

$$\text{Constant-2 (C2)} = 0.01$$

The formula follows the table.

Technical factor	Description	Weight	Perceived Complexity	Calculated Factor (Weight×Perceived Complexity)
T1	Distributed system (running on multiple machines)	2	4	$2*4=8$
T2	Performance objectives (are response time and throughput performance critical?)	1	2	$1*2=2$
T3	End-user efficiency	1	3	$1*3=3$
T4	Complex internal processing	1	1	$1*1=1$
T5	Reusable design or code	1	3	$1*3=3$

T6	Easy to install (are automated conversion and installation included in the system?)	0.5	3	$0.5 \times 3 = 1.5$
T7	Easy to use (including operations such as backup, startup, and recovery)	0.5	3	$0.5 \times 3 = 1.5$
T8	Portable	2	3	$2 \times 3 = 6$
T9	Easy to change (to add new features or modify existing ones)	1	3	$1 \times 3 = 3$
T10	Concurrent use (by multiple users)	1	1	$1 \times 1 = 1$
T11	Special security features	1	0	$1 \times 0 = 0$
T12	Provides direct access for third parties (the system will be used from multiple sites in different organizations)	1	0	$1 \times 0 = 0$
T13	Special user training facilities are required	1	0	$1 \times 0 = 0$
Technical Factor Total:				30

$$\text{TCF} = \text{Constant-1} + \text{Constant-2} * \text{Technical Factor Total} = 0.6 + (0.01 * 30) = 0.90$$

5.e. ECF calculation

Similar to TCF, the ECF involves a set of factors with predefined weights, along with a team generated perceived impact from 1 to 5, with 0 for N/A. From the Software Engineering, the following were used to guide the perceived impact values:

For factors E1-E4, 0 means no experience in the subject, 3 means average, and 5 means expert. For E5, 0 means no motivation for the project, 3 means average, and 5 means high motivation. For E6, 0 means unchanging requirements, 3 means average amount of change expected, and 5 means extremely unstable requirements. For E7, 0 means no part-time technical staff, 3 means on average half of the team is part-time, and 5 means all of the team is part-time. For E8, 0 means an easy-to-use programming language will be used, 3 means the language is of average difficulty, and 5 means a very difficult language is planned for the project.

The following constants are used for ECF:

Constant-1 (C1) = 1.4

Constant-2 (C2) = -0.03

Environmental factor	Description	Weight	Perceived Impact	Calculated Factor (Weight×Perceived Impact)
E1	Familiar with the development process	1.5	2	$1.5 \times 2 = 3$
E2	Application problem experience	0.5	4	$0.5 \times 4 = 2$
E3	Paradigm experience	1	3	$1 \times 3 = 3$
E4	Lead analyst capability	0.5	2	$0.5 \times 2 = 1$
E5	Motivation	1	4	$1 \times 4 = 4$
E6	Stable Requirements	2	1	$2 \times 1 = 2$
E7	Part-time staff	-1	5	$-1 \times 5 = -5$
E8	Difficult programming language	-1	3	$-1 \times 3 = -3$
Environmental factor total:				7

$$ECF = \text{Constant-1} + \text{Constant-2} \times \text{Environmental Factor Total} = 1.4 + (-0.03 \times 7) = 1.19$$

5.f. UCP calculation

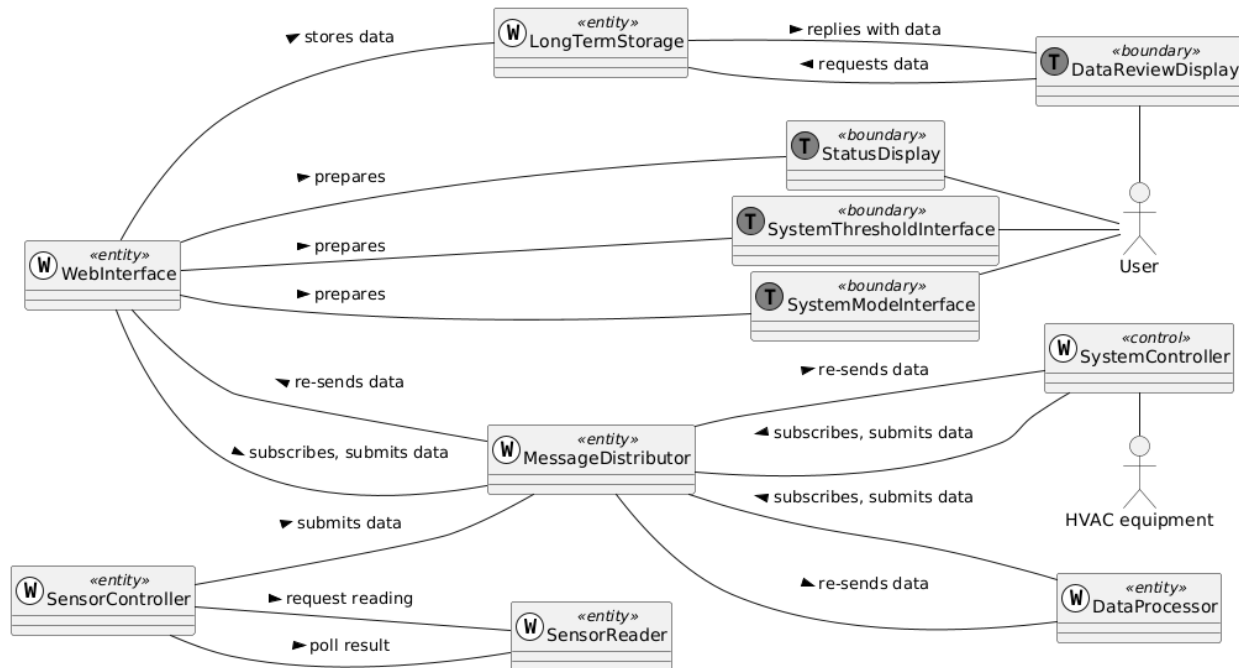
Per the UCP equation from 5. above:

$$UCP = UUCP \times TCF \times ECF = 59 \times 0.90 \times 1.19 = 63.189 = 63.$$

6. Domain Analysis

6.a. Conceptual Model

Below is the graphical representation of the domain model described in sections 6.a.i to iv.



6.a.1. Concept definitions

The following concept definition tables are derived from the Report 1 3.c.i. Casual Description(s) of the project's main use cases, one table per use case. Many of the cases are intended to be very flexible and configurable.

Many of the concepts in Use Cases 4 through 7 will be implemented via the choice of preexisting off the shelf open source software to reduce the need to reinvent the wheel for many of the user interface elements. The software selected is already very polished to a degree which I could not achieve in a short time scale, especially working by myself.

Responsibility Descriptions for UC-1: Input Data

Each system can have a large number of Sensor Boards which execute the SensorReader concept, each configured differently, with different sensors.

Responsibility Description	Type	Concept Name
Connect to and maintain connection to the MessageDistributor	D	SensorController

Periodically request readings from connected sensors per configuration	D	SensorController
Collect measurements as requested	D	SensorReader
Send to MQTT Broker (MessageDistributor) for distribution	D	SensorController

Responsibility Descriptions for UC-2: Extra Data

The Data Processor can be broken down into additional submodules, either running as independent NodeJS instances or as a larger single NodeJS instance. Because most Data Processor tasks are event driven, one Data Processing module may act upon the receipt of a subscribed topic, then send a topic, which a different part of the Data Processor is subscribed to, causing it to process the data, and generate additional MQTT messages.

Responsibility Description	Type	Concept Name
Subscribe to MQTT Broker (MessageDistributor) for raw sensor data topics	D	DataProcessor
Create Human Readable data points	D	DataProcessor
Generate derived data points upon receipt of required data points from a specific sensor location	D	DataProcessor
Send all generated and converted data out to MQTT Broker (MessageDistributor) for distribution	D	DataProcessor

Responsibility Descriptions for UC-3: System Controls

System Control boards are designed to be flexible in how they can be configured, and any given system setup may have more than one, and each may operate more than one piece of HVAC equipment. Each should be configured to allow for a minimal functionality if any board is found operating without any connectivity to the rest of the system.

Responsibility Description	Type	Concept Name
Subscribe to specific data topics for system operation decision making	D	SystemController
Subscribe to specific system mode topics to set system mode, for system operation decision making	D	System Controller

Periodically read onboard sensor for backup data point if there is any system or network issues	D	System Controller
Save last set mode for cold start operation	D	System Controller
Make system operation decisions, and verify equipment is operating as requested	D	SystemController
Report status of all devices controlled by each System Controller to MQTT Broker (MessageDistributor)	D	SystemController
Report value of onboard backup sensors to MQTT Broker (MessageDistributor)	D	SystemController
Report receipt of system mode messages to MQTT Broker (MessageDistributor)	D	SystemController
Distribute all received messages to subscribed actors	D	MessageDistributor

Responsibility Descriptions for UC-4: User Controls

Responsibility Description	Type	Concept Name
User sets system mode via web interface	K	SystemModelInterface
User sets system thresholds via web interface	K	SystemThresholdsInterface
User views system information via web interface	K	StatusDisplay
Web interface sends mode, threshold data to MQTT Broker (MessageDistributor)	D	WebInterface
Distribute all received mode and threshold change data to subscribed actors	D	MessageDistributor
Devices subscribed to mode or threshold topics act upon changes	D	SystemController

Responsibility Descriptions for UC-5: Data Collection

Responsibility Description	Type	Concept Name
Subscribe to all data topics, and retain for short term review	D	WebInterface
Convert to correct format and send all received data topics to long term storage	D	WebInterface
Receive and save all data into time series database	D	LongTermStorage

Responsibility Descriptions for UC-6: Data Display

Responsibility Description	Type	Concept Name
Retrieve and generate graphical representations of short term data	D	WebInterface

Display short term data for User	K	StatusDisplay
----------------------------------	---	---------------

Responsibility Descriptions for UC-7: Data Display 2

Responsibility Description	Type	Concept Name
Retrieve long term data queries for display via a web interface	D	LongTermStorage
Generate graphical views of long term data via a web interface	D	DataReviewDisplay

6.a.2. Association definitions

Association Descriptions for UC-1: Input Data

Concept Pair	Association Description	Association Name
SensorController <→> SensorReader	Controller requests sensor readings on a periodic schedule	requests reading
SensorController <→> SensorReader	Polls for reading result	requests result
SensorController <→> MessageDistributor	Sends data point to MessageDistributor	submits data
MessageDistributor <→> Subscribed entities	Distributes to all subscribed entities	re-sends data

Association Descriptions for UC-2: Extra Data

Concept Pair	Association Description	Association Name
DataProcessor <→> MessageDistributor	Requests subscription to “topic”	subscribes
MessageDistributor <→> DataProcessor	Receives data from subscribed topics	re-sends data
DataProcessor <→> MessageDistributor	Sends data point to MessageDistributor	submits data

MessageDistributor <-->	Distributes to all subscribed entities	re-sends data
Subscribed entities		

Association Descriptions for UC-3: System Control

Concept Pair	Association Description	Association Name
SystemController <--> MessageDistributor	Requests subscription to "topic"	subscribes
MessageDistributor <--> SystemController	Receives data from subscribed topics	re-sends data
SystemController <--> MessageDistributor	Sends data point to MessageDistributor	submits data
MessageDistributor <--> Subscribed entities	Distributes to all subscribed entities	re-sends data

Association Descriptions for UC-4: User Controls

Concept Pair	Association Description	Association Name
WebInterface <--> SystemModelInterface	WebInterface builds SystemModelInterface from config file	prepares
WebInterface <--> SystemThresholdsInterface	WebInterface builds SystemThresholdsInterface from config file	prepares

WebInterface <--> StatusDisplay	WebInterface builds StatusDisplay from config file	prepares
WebInterface <--> MessageDistributor	Requests subscription to "topic"	subscribes
MessageDistributor <--> WebInterface	Receives data from subscribed topics	re-sends data
WebInterface <--> MessageDistributor	Sends data point to MessageDistributor	submits data
MessageDistributor <--> Subscribed entities	Distributes to all subscribed entities	re-sends data

Association Descriptions for UC-5: Data Collection

Concept Pair	Association Description	Association Name
WebInterface <--> LongTermStorage	Sends generated and received data to long term storage	stores

Association Descriptions for UC-6: Data Display

Concept Pair	Association Description	Association Name
WebInterface <--> SystemModelInterface	WebInterface builds SystemModelInterface from local data store	prepares

WebInterface <--> SystemThresholdsInterface	WebInterface builds SystemThresholdsInterface from local data store	prepares
WebInterface <--> StatusDisplay	WebInterface builds StatusDisplay from local data store	prepares

Association Descriptions for UC-7: Data Display 2

Concept Pair	Association Description	Association Name
DataReviewDisplay <--> LongTermStorage	Sends query requests to long term storage	request data
LongTermStorage <--> DataReviewDisplay	Sends requested data	replies with data

6.a.3. Attribute definitions

As I am currently focusing on the Input Data UC-1 case per Report 1 3.c.iii Traceability Matrix, other than the obvious hardware board status and error attributes, and the data payload formats (raw data from Sensor Boards as the payload, with a sensor type/location topic, and JSON payloads for everything not operating on a microcontroller), most concepts are already built into their respective software.

6.a.4. Traceability Matrix

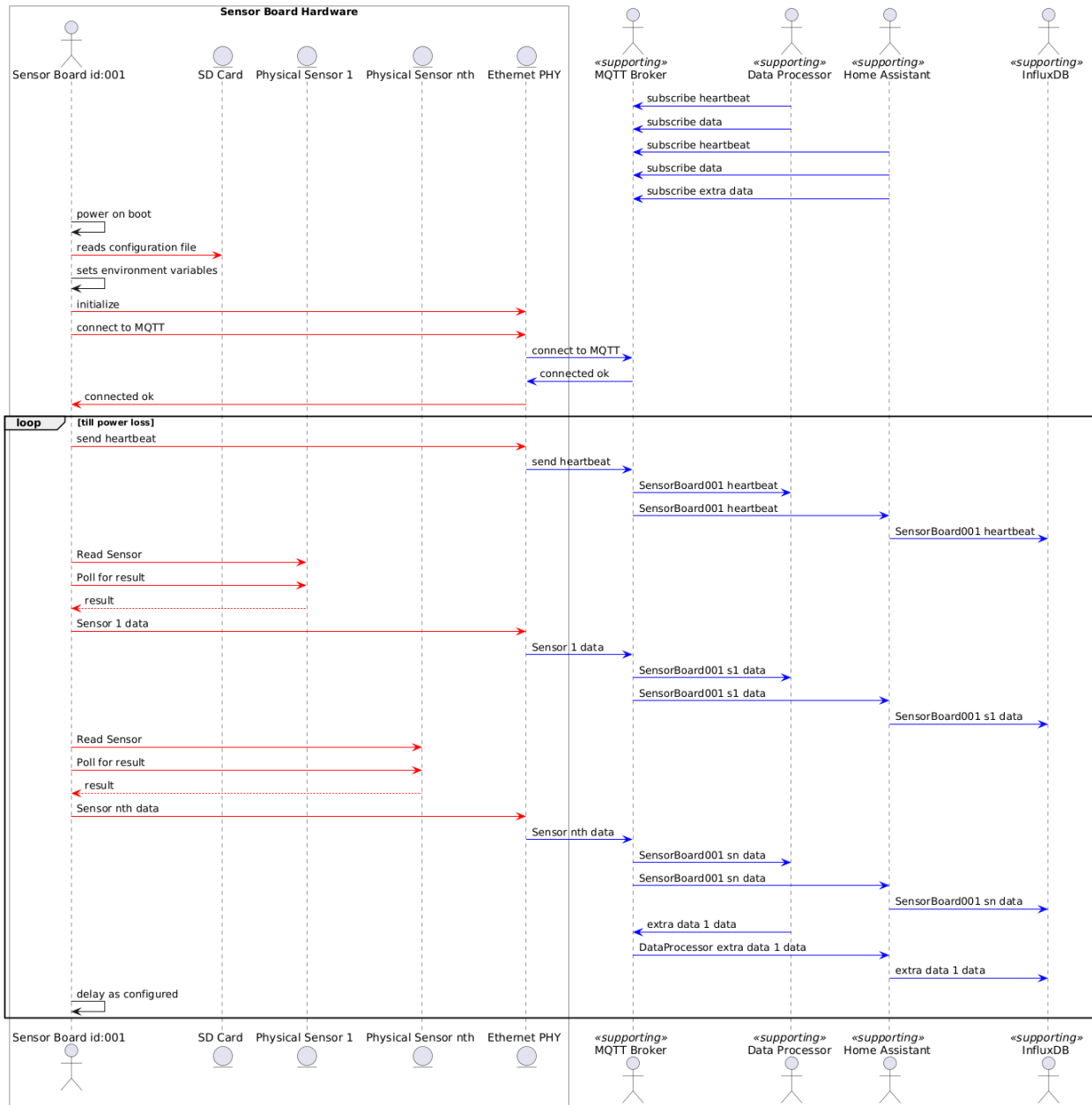
Use Case	PW	SensorReader	SensorController	MessageDistributor	DataProcessor	SystemController	SystemModelInterface	SystemThresholdInterface	StatusDisplay	WebInterface	LongTermStorage	DataReviewDisplay
UC1	79	X	X	X								
UC2	64			X	X							
UC3	39			X		X						
UC4	4			X		X	X	X	X	X		
UC5	15									X	X	
UC6	19								X	X		
UC7	15										X	X

7. Interaction Diagrams

The following diagram is for UC-1 Input Data from Report 1. The red messages are sent via Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I2C) buses which interconnect the various hardware breakout boards in the prototype sensor board microcontroller with the peripheral boards. All blue messaging is sent over off the shelf ethernet networking. Responsibilities are assigned mainly following the High Cohesion Principle, as the microcontroller is very good at reading sensor data from sensor specific hardware, and can pass that data to the network, where it is not particularly good at large sets of floating point math which is used to create the required extra data points. The responsibility for calculation of extra data has been delegated to the Data Processor actor, which will be implemented using NodeJS on normal computer hardware, for which floating point math is relatively trivial. Because

the Sensor Board is the first to know the sensor values, per Expert Doer Principle, it is the obvious candidate for sending the data to the network.

UC-1 Input Data sequence diagram.

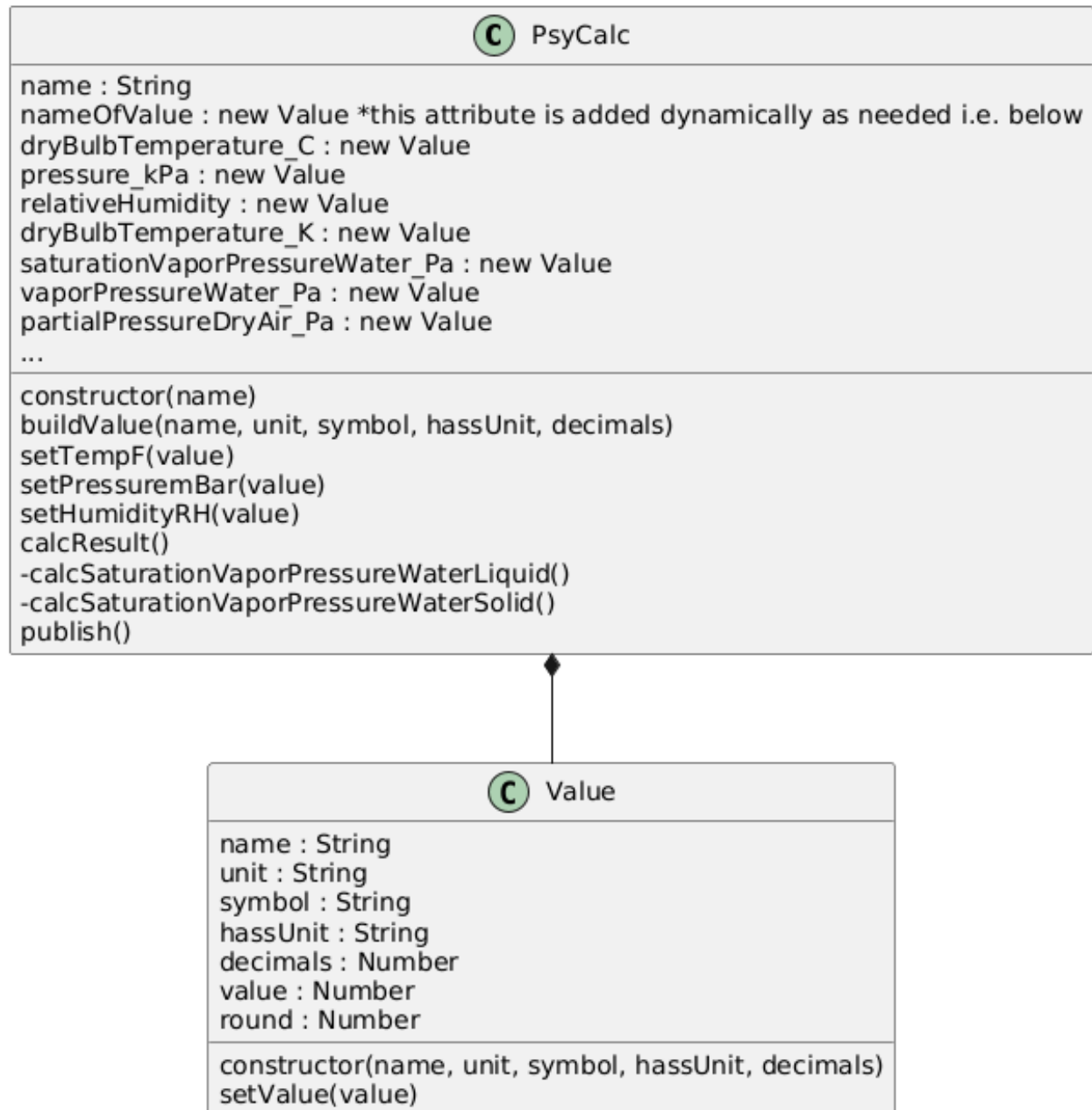


8. Class Diagram and Interface Specification

8.a. Class Diagram

As the firmware for the Sensor Board is not particularly conducive to object oriented programming due to minimal support for dynamic ram operations, I will diagram the classes used in the Data Processor's psychrometric extra data system. I have not found a particularly good way to diagram JavaScript, due to the fact that classes in JavaScript are just objects themselves, and all are dynamic as far as attribute types, or even the attributes themselves. My PsyCalc class's definition only has the name attribute, but as it calculates the psychrometric and thermal dynamic properties, it generates additional attributes that were not actually in the class definition. These attributes only exist in a specific instance of the PsyCalc class, and may be different between individual instances. Each newly used attribute for this class is created as an object whose key is the attribute name, and whose value is a new instance of the Value class object. The Value class creates all the attributes needed for a new value object, with attributes that make each property have the correct units and rounding for human readability in Home Assistant and InfluxDB.

In the Data Processor, there is an external config.json file that the Data Processor uses to create the appropriate instances of the PsyCalc class. This configuration object (JSON is the stringified version of a normal JavaScript object) holds an object of topic objects that are used by the MQTT client.on("message", ... event handler to call the correct method of the correct instance of PsyCalc when a MQTT message is received from a sensor board so that all new values are created and distributed via MQTT.



8.b. Data Types and Operation Signatures

1. PsyCalc Class

 PsyCalc
<pre> name : String nameOfValue : new Value *this attribute is added dynamically as needed i.e. below dryBulbTemperature_C : new Value pressure_kPa : new Value relativeHumidity : new Value dryBulbTemperature_K : new Value saturationVaporPressureWater_Pa : new Value vaporPressureWater_Pa : new Value partialPressureDryAir_Pa : new Value ... </pre>
<pre> constructor(name) buildValue(name, unit, symbol, hassUnit, decimals) setTempF(value) setPressureBar(value) setHumidityRH(value) calcResult() -calcSaturationVaporPressureWaterLiquid() -calcSaturationVaporPressureWaterSolid() publish() </pre>

This class is used to create instances of the PsyCalc object. Each PsyCalc object represents a datapoint that is either gathering or has the necessary three points (temperature, relative humidity, and air pressure) required to create a full set of psychrometric and thermodynamic properties for a specific location either in an HVAC system or for a specific space in a residence or outside. As the required data points may arrive in any order or at any time, this class tries to recalculate and publish to MQTT all the data points each time a new input value is received. If the instance does not yet have the other two points required, it will not publish any values.

- The only attribute created at instantiation is the PsyCalc instance's name, used when publishing values. This is a String value (as much as any JavaScript variable can have a type).

- Every datapoint either input or calculated becomes an additional attribute, created in the object's top level, whose key is the name of the datapoint, and whose value is a new instance of the Value class, setup with all of the attributes created by the constructor.
- The buildValue method is used to create a new attribute, and set its value as the object created by calling the constructor for the Value class with the parameters specified in the buildValue call.
- The setTempF method is called with the parameter being a temperature in degrees F, which creates or updates a dry bulb temperature in degrees C attribute, and then calls the object's calcResult method
- The setPressureBar method, as the setTempF method, creates or updates the required pressure attribute, then calls calcResult
- The setHumidityRH method, like above, creates or updates the required relative humidity attribute, then calls calcResult
- The calcResult method checks for the three required attributes, and if present, then creates or updates every other datapoint that can be made, by calling buildValue and the Value class's setValue methods with the required math operation for each data point. It also calls either calcSaturationVaporPressureWaterLiquid or calcSaturationVaporPressureWaterSolid based on the dry bulb temperature towards the beginning of the datapoint calculations, as other datapoints require the saturation vapor pressure value for their calculations. At the end, calcResult calls the publish method to publish all of the datapoints to MQTT for use elsewhere and storage.
- The calcSaturationVaporPressureWaterLiquid or calcSaturationVaporPressureWaterSolid methods build the appropriate attribute, call setValue with the appropriate calculation, and then return.

- The publish method publishes all of this object's datapoint attributes to a MQTT topic using the name attribute in the topic string, with a stringified JSON object as the message payload, for use by other systems.

2. Value Class

 Value
name : String unit : String symbol : String hassUnit : String decimals : Number value : Number round : Number
constructor(name, unit, symbol, hassUnit, decimals) setValue(value)

- The Value class is used to create datapoint attribute objects for the PsyCalc class. These attribute objects contain the datapoint value, along with rounded for human readability versions, as well as the type of datapoint, its units, the symbol for its units and how many decimal places are used for rounding. All of these keys and values are converted to JSON strings and published in the payload part of the MQTT messages sent by the Data Processor, which are then used by other actors like Home Assistant and saved to the InfluxDB database in human readable form.
- The name attribute is set to the name of the datapoint
- The unit attribute is set to the Human readable unit name.
- The symbol attribute is set to the symbol commonly used for this data point, if applicable. i.e. density uses ρ , lowercase Greek rho.
- The hassUnit attribute has the unit to display in Home Assistant

- The decimals attribute contains the number of decimal places to use when rounding for human readability
- The value attribute contains the current calculated value
- The round attribute contains the rounded value for human readability
- The constructor method generates a new object with the attributes set via the method's parameters.
- The setValue method sets the value attribute, and then rounds per the number of decimals and stores the result into the round attribute. This method is called each time the value needs to be updated.

8.c. Traceability Matrix

For the Class Diagram Traceability Matrix, I have listed the domain concepts, and will explain what classes are derived and why.

SensorReader – This concept involves the hardware required to generate a mathematical value representing a real world datapoint which is both human readable and useable for other calculations. There are no specific classes created as the programming related to this concept is written as firmware for a microcontroller, although I will be trying to tweak each sensor's software to conform to a more standard function interface.

SensorController – This concept involves the firmware and hardware on the microcontroller required to read sensor values from any of the sensors used in this application. There are no specific classes generated by myself, however some of the libraries used sometimes contain C++ classes, although most is written in C. I am forking and modifying some of the libraries so that a more common interface can be used between the different sensor types, and to support multiple copies of the same sensor hardware per Sensor Board.

MessageDistributor – This concept involves the off the shelf open source MQTT Broker software Mosquitto by the Eclipse Foundation (<https://www.mosquitto.org/>) which uses the [Eclipse Public License \(EPL\) 2.0](#), allowing for any commercial use. This software is part of the software stack, that would be deployed via a docker compose file onto the customers computer hardware.

DataProcessor – This concept is implemented using NodeJS to run a JavaScript program, in a docker container, via the docker compose file onto the customers computer hardware. The PsyCalc and Value classes are used in this JavaScript program. Additional future classes and processing will be executed by this program, for instance generating a single whole house virtual temperature created from many different temperature sensors throughout a residence, to simplify HVAC system operation decisions.

SystemController – This concept involves slightly more powerful microcontroller hardware than the Sensor Board, due to its requirement to both send operating data and listen for specific data messages to base system operating decisions on, as well as to listen for messages that change system modes and thresholds. Specific base operating logic is to be implemented into these System Controller boards so that they can reliably function independently if disconnected from the rest of the system. As the Sensor Controller and Sensor Reader concepts, the firmware for these boards will be made in C, and is not normally made with OOP style classes.

SystemModelInterface – This concept is implemented with yaml format packages for the open source Home Assistant software, which uses the [Apache 2.0 license](#) that allows for commercial use.

SystemThresholdInterface – Same as SystemModelInterface

StatusDisplay – Same as SystemModelInterface

WebInterface – Same as SystemModelInterface

LongTermStorage – This concept is implemented using InfluxDB, off the shelf open source software using the [MIT license](#). InfluxDB is a time series database, ideally suited to taking many similar data points

and storing them in chronological order. It also will be part of the docker compose stack on customer hardware.

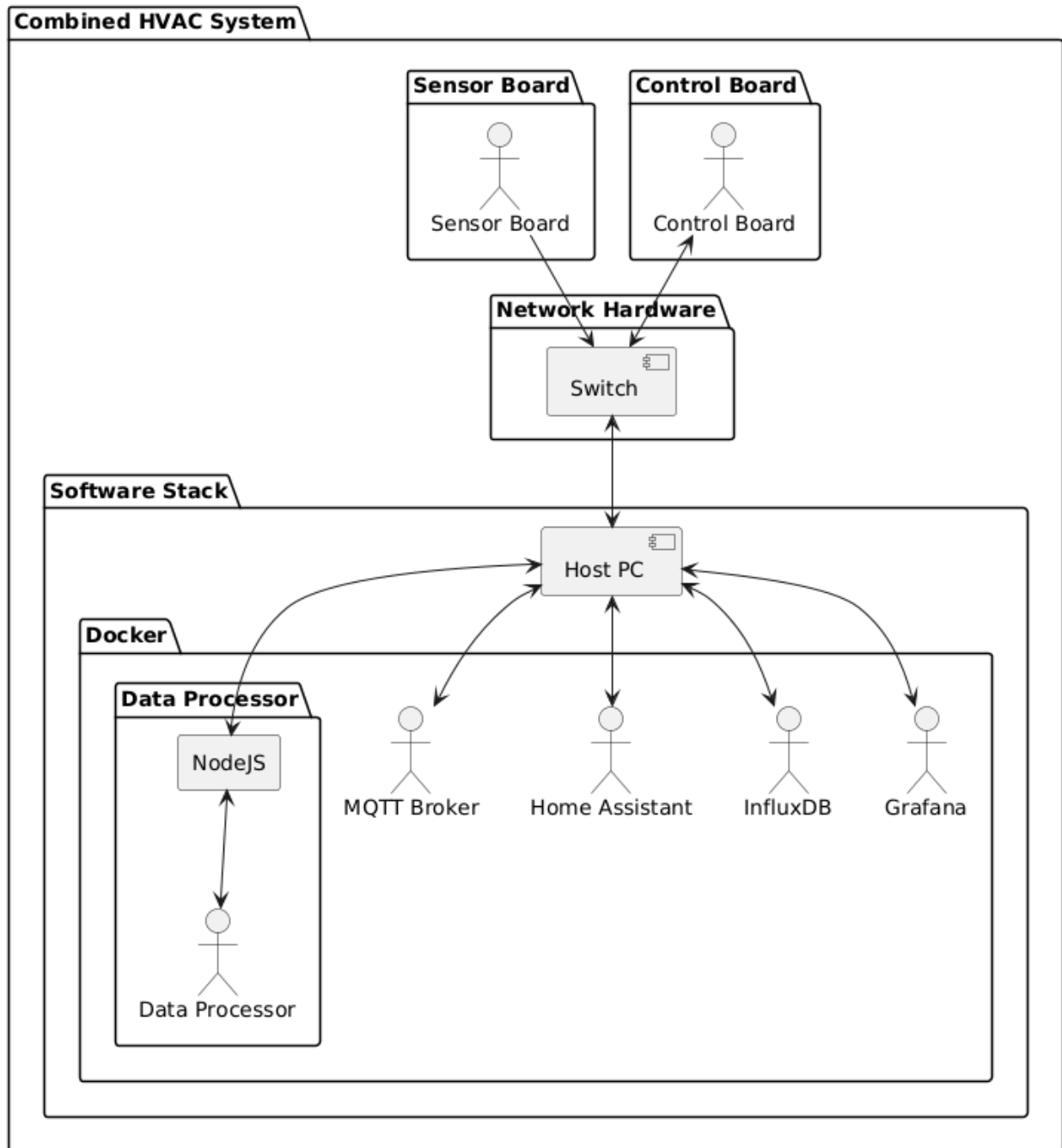
DataReviewDisplay – This concept is implemented using Grafana off the shelf open source software using the [GNU Affero General Public License v3.0](#). As above, to be installed via docker compose on customer hardware.

8.d. Design patterns

The classes detailed above are implemented in JavaScript, and JavaScript has built in methods for generating (publishing) events and for setting listeners (subscribers), and it is definitely one way of using the code to handle multiple sets of psychrometric data, as each new psychrometric data point can subscribe to the three required detailed input datapoints (temperature, humidity, pressure), and then recalculate and publish all the derived data points as the events are received. As already mentioned, the main mechanism for all the networked microcontrollers and other processes in this system all use MQTT, which is built on the publisher-subscriber paradigm.

9. System Architecture and System Design

9.a. Identifying Subsystems



9.b Architecture Styles

The sensors will have a client server architecture via MQTT. Derived data calculations processed by the Data Processor will be event driven by subscribing to MQTT topics, and will cause event driven reactions by posting the derived data back to MQTT for other subscribers to collect and save, and act upon. The sensors will operate on a fixed interval basis, but everything else is based on the event driven architecture.

9.c Mapping Subsystems to Hardware

9.c.1. Subsystem 1. Sensor Boards

Initial plans are to use STM32 based microcontrollers, W5500 SPI to cabled ethernet modules, SD Card to SPI modules, and various sensor breakouts using SPI, I2C or analog data. By using an SD Card for configuration, the Sensor Boards should only need to be flashed when produced, and all configuration will be on the SD Card.

9.c.2. Subsystem 2. Control Boards

Control boards will start with higher model STM32 controllers, which have more flash memory for program storage. They will follow the same basic design as the Sensor Boards, but will need more programming space to be able to both send and respond to MQTT messages, along with failsafe operation using an onboard temperature sensor so that an error or problem in the network stack or other system does not prevent it from controlling the HVAC equipment on a bare minimum basis, which is what the average thermostat does now.

9.c.3. Subsystem 3. Network

Standard POE network switches and ethernet wiring will be used for initial networking. POE Ethernet is much more reliable than WIFI, but I definitely understand homeowner's reluctance to run cables, therefore accommodations for future WIFI sensor and control boards will be made in future versions.

9.c.4. Subsystems 4-6. Software Stack

The software stack is to be deployed onto a pc, using docker to allow the system to work across many different user hardware environments. Used PC's or Raspberry Pi SBC's are both suitable candidates for the software required. Home Assistant has had a Raspberry Pi edition for quite some time. The Data Processing system will be developed in NodeJS in a docker container, and docker images will be pushed up to Docker Hub so that it can be downloaded by anyone running docker into the software stack with the other off the shelf open-source software.

9.d. Connectors and Network Protocols

Most traffic will be sent using the MQTT protocol, which uses TCP, between most of the hardware and software. MQTT was chosen because of its extremely small packet size, and support for JSON payloads, which are supported by both Home Assistant and NodeJS. MQTT's client/subscriber architecture also allows system operation in an event driven fashion. Home Assistant will push data to InfluxDB with its built-in package, which uses an HTTP API. TLS encryption for the packets is a bit too much for the smaller STM32 boards, which means all sensor and control data will not have encryption locally. If the software stack is deployed to a cloud environment like AWS, then a local MQTT broker can forward the packets to the MQTT broker in the cloud stack, with TLS being provided across that link.

9.e. Global Control Flow

The system operates on a real time basis, and timing is needed for some derived data. The sensor boards will collect data at a preset interval, but all other software will operate on an event-response system. Things like heartbeat messages that allow the system to be distributed across many components but still have error checking are key to system reliability.

9.f. Hardware Requirements

This system has a fairly light weight foot print, and can be run by Raspberry Pi 3's (Quad Core Broadcom System On a Chip (SOC) processor, 64-bits, 1.2 GHz, 1 GB of Ram, 100 BaseT Ethernet, and uses Micro SD cards for a hard drive). Most users will interact with Home Assistant via their cell phones using either Home Assistant's native Android or IOS apps or its web interface, which is also very good on full size screens. Grafana also has an excellent user interface via web page.

10. Algorithms and Data Structures

10.a. Algorithms

This project is not using any specific algorithms

10.b. Data Structures

Complex data structures do exist in this project, but they are all implemented in other software that is not written by myself.

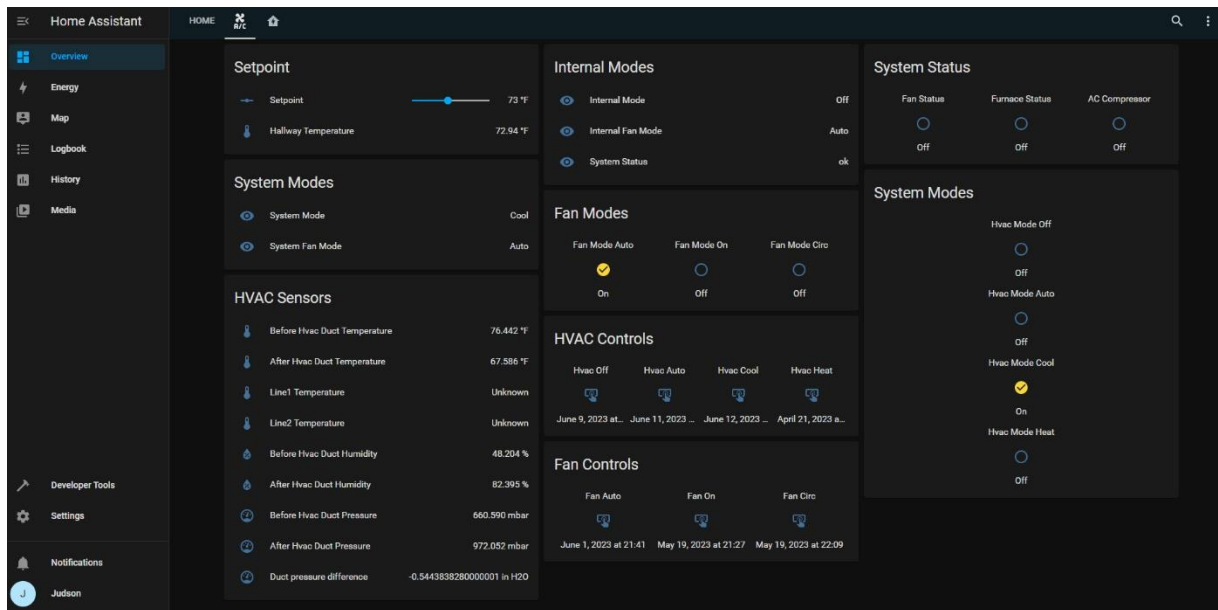
10.c. Concurrency

This system does not specifically use multithreading in any parts written by myself, other than there will be multiple sets of code running on separate processors and separate microcontrollers at the same time, all concurrency is handled via sending and processing MQTT messages (events).

11. User Interface Design and Implementation

The user interface will be leveraging the built in interfaces of Home Assistant, an example of which is show below. Home Assistant has built in sliders, buttons, status indicators for "binary sensors" which can only hold yes/no, on/off, true/false data, shown with either blank circles or yellow check circles, and sensor items, which when clicked bring up a short term graph of that sensor's readings. These Home Assistant controls natively support MQTT, for example when you slide the slider, it publishes a MQTT message, with the topic specified in the slider config, and the new value as the

payload. The good part is that the slider value does not change until Home Assistant receives an updated value in a separate response MQTT message. If the other actor tasked with listening to and changing because of a control message does not send back an update message with the new value, Home Assistant will still display the previous value. Similarly the Controls buttons send a MQTT message, and the System/Fan Modes do not update until the system changed and sent MQTT messages for the new states.



12. Design of Tests

The Data Processor is being designed with receiving very small MQTT messages which have the payload type specified in the topic string, and then resending them to a different topic with a full JSON payload that includes rounded and full values, proper units, and other relevant information. As I don't expect to have implemented sensors for all expected sensor data types, another NodeJS program will be made to send at least one message of each proposed data type to verify that the Data Processor is correctly generating the full data messages.

Additional integration testing to verify the correct operation of psychrometric extra data point creation will be done as well. Some sensors are bad at creating erroneous outliers for which I will implement a median value filter in the Data Processor for that specific data type, which will create additional integration testing. Median value filtering is better than moving average filters for any sensors that create occasional values that are wildly incorrect, as these will be completely ignored in a median filter, but will cause a large offset for a period of time in a moving average filter.

13. History of Work, Current Status, and Future Work

13.a. Merging the Contributions from Individual Team Members

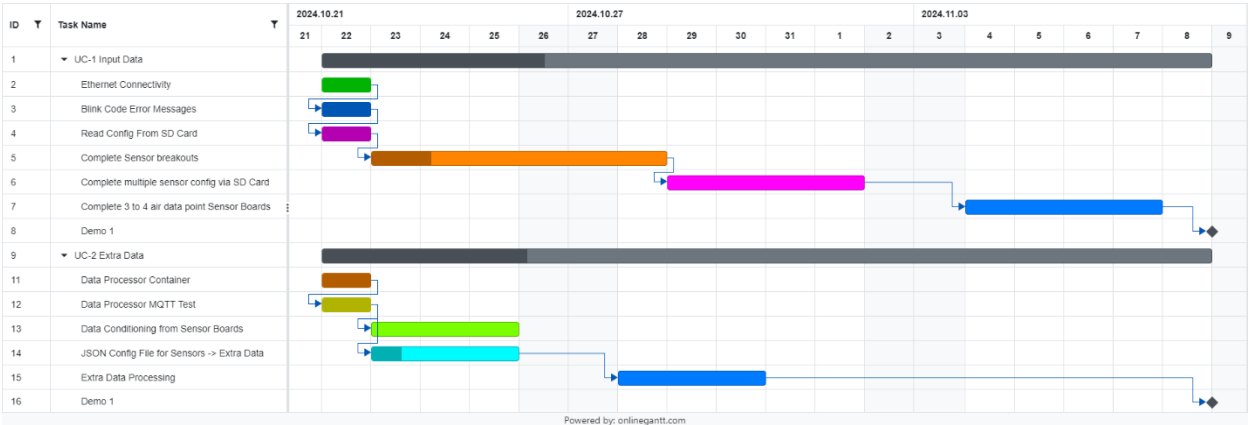
N/A

13.b. Project Coordination and Progress Report

Work on UC-1 is continuing, with Ethernet connectivity to the MQTT Broker and configuration via a text file on an SD card complete, along with an error blink code system to troubleshoot connectivity issues. Soldering sensor breakout boards so they can be connected to a few sensor boards with breadboard, and improving SPI and I2C noise immunity over dupont wire ribbon cable is progressing. If these ever reach the production stage, this sort of noise issue will be eliminated via short pcb traces and stacking pin header interconnects instead of overly long ribbon cables which act like noise antennas.

Code for the Data Processor is also being completed, so that integrated sensor testing with the Sensor Board can be completed. A list of specific sensor data types, units, and rounding is being created to offload data conditioning from the Sensor Boards to the Data Processor. As new sensor hardware is added, I will be conforming the sensor readings from hardware to one of the message types to simplify how data is sent from the Sensor Boards, and for code reusability in the Data Processor concept.

13.c. Plan of Work



13.d. Breakdown of Responsibilities

N/A

14. References

- [1] American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. [ASHRAE] (2021). *ANSI/ASHRAE 41.6-2021 Standard Methods for Humidity Measurements*. ASHRAE. Available at: <https://webstore.ansi.org/standards/ashrae/ansiashrae412021-2458418?srsId=AfmBOoqM6NXlnWfJP06lvtRAehRMJM1W9gaSTHLGm7tBu8cc94xuNZqR>
- [2] Docker inc. (2024). *Docker: Accelerated Container Application Development*. Available at: <https://www.docker.com> [Accessed: 25 November 2024].
- [3] Espressif Systems. (2024). *ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems*. Available at: <https://www.espressif.com/en/products/socs/esp32> [Accessed: 25 November 2024].
- [4] Grafana Labs. (2024). *Grafana: The open observability platform | Grafana Labs*. Available at: <https://grafana.com/> [Accessed: 25 November 2024].
- [5] Home Assistant. (no date). *Home Assistant*. Available at: <https://www.home-assistant.io/> [Accessed: 25 November 2024].
- [6] Wikipedia Contributors. (1 November 2024). *I²C*. Wikipedia, The Free Encyclopedia. Available at: <https://en.wikipedia.org/w/index.php?title=I%C2%B2C&oldid=1254687349> [Accessed: 25 November 2024].
- [7] InfluxData Inc. (2024). *InfluxDB Time Series Data Platform | InfluxData*. Available at: <https://www.influxdata.com/> [Accessed: 25 November 2024].
- [8] Eclipse Foundation AISBL. (no date). *Eclipse Mosquitto*. Available at: <https://www.mosquitto.org/> [Accessed: 25 November 2024].
- [9] MQTT.org. (2024). *MQTT – The standard for IoT Messaging*. Available at: <https://mqtt.org/> [Accessed: 25 November 2024].
- [10] OpenJS Foundation. (no date). *Node.js – Run JavaScript Everywhere*. Available at: <https://nodejs.org/en> [Accessed: 25 November 2024].

[11] Online Gantt. (no date). *Free Online Gantt Chart Software*. Available at:

<https://www.onlinegantt.com/#/gantt> [Accessed: 25 November 2024].

[12] PlantUML (no date). *Open-source tool that uses simple textual descriptions to draw beautiful UML*

diagrams. Available at: <https://plantuml.com> [Accessed: 25 November 2024].

PlantUML model links:

Class Diagram [model link](#)

Class Diagram for PsyCalc [model link](#)

Class Diagram for Value [model link](#)

Domain Model [model link](#)

Input Data Sequence Diagram [model link](#)

System Sequence Diagram [model link](#)

Use Case Diagram [model link](#)

[13] Wikipedia Contributors. (20 October 2024). *Psychrometrics*. Wikipedia, The Free Encyclopedia.

Available at: <https://en.wikipedia.org/w/index.php?title=Psychrometrics&oldid=1252197041> [Accessed: 25 November 2024].

[14] Raspberry Pi Ltd. (no date). *Raspberry Pi*. Available at: <https://www.raspberrypi.com/> [Accessed: 25 November 2024].

[15] Wikipedia Contributors. (23 November 2024). *Serial Peripheral Interface*. Wikipedia, The Free Encyclopedia. Available at:

https://en.wikipedia.org/w/index.php?title=Serial_Peripheral_Interface&oldid=1259201545 [Accessed: 25 November 2024].