ENG4940 Capstone I

Concept Generation, Conceptual Design

Prototype Report

Environment Sensing Occupancy Integration System

Name:
Raven Castaneda
Eric Dube
Brady Ibanez
Taylor Somann

Student Number :
100559932
100555757
100367230
100502392

Contents

List of Tables

List of Figures

List of Abbreviations

BLE Bluetooth Low Energy. 22

GPIO general purpose input/output. 13

HA Hub Application. 5, 7, 9

HVAC heating, ventilation, and air conditioning. 6, 8

i2c inter-integrated circuit. 13

IoT Internet of Things. 8, 21

ML Machine Learning. 5, 21

PIR passive infrared. 7, 19

1 Concept Generation and Analysis

Following the perspective of the problem definition, design process, and project plan outlined and considered in this implementation's R1 report, there are a variety of potential solutions defined to address the task to be satisfied. Of these perspective solutions, the most prominent goal to be maintained is that of facilitating an environment detection system capable of not only self-satisfying location detailing, connectivity, and environment analysis, but also to be capable of providing this environment analysis outside of the spectrum of internal local representation. This latter requirement is meant in order to satisfy the connectivity of the Machine Learning (ML) brother team for this project.

Of the requirements to consider, it was determined that overall the system configuration factors to most directly establish were power supply, node connectivity, node location recognition, data interaction from the auxiliary systems and user interaction perspectives, and system to user interfacing. In order to address these requirements, the following concepts were generated for each category:

1.1 Possible System Considerations

Power Supply

- AC line connected
- Self-provisioned node power supply

Node Connectivity

- Zigbee
- BLE
- WiFi
- i2c

Node Location Recognition and Organization

- Find3 localization API
- Manual user definition
- Individual node per device vs. clusters

Data Recognition/Control/Interaction

- Central hub/Hub Application (HA)
- Remote DB and application
- Per node designated control

System/User Interfacing

- User recognition/profiling/authentication vs. general users

The node location recognition definition was also assessed from the singular or clustered node approach, depending on whether or not environment room definition is required. This was assessed to not only address issues related to location mapping, but communication protocol as well.

1.2 Similar System Approaches and Assessment

In relation to similar systems referred to in report R1, it was determined that this much larger cohesive environment encompassing system should adhere to a more self-reliant mechanism base. Existing environment control systems like NEST[1] and Ecobee[2] allow for environment realization in terms of heating, ventilation, and air conditioning (HVAC) related parameter measurements facilitated by traditional and already installed residence environment measurement tools. Traditional thermostat thermometers are their bread and butter since these aforementioned systems are meant to be installed over traditional analog thermostats. They include no capacity for ulterior means of environment parameter measurement, and thus resolve to loose on guaranteeing accurate and area defined readings.

Occupancy detection systems at the moment work from either a very simplistic general detection approach, or a much more complex dedicated environment installed count mechanism in conjunction with wireless protocol recognition devices required. These consist of examples such as drop-arm turn-style "must pass" detectors in commercial implementations, or RF card tag identifiers, both of which would act as cumbersome infringements to a homestead.

As seen in existing residential occupancy detection, such as that of Lutron LRF series of sensors [3], occupancy detection comes down to the inclusion of general area detection with the potential for a non guaranteed occupant reference. While this stands for in room heuristic detection, the shortcoming still stands of being unable to pose a more refined and careful assessment of per-room detection. The inability to guarantee occupant count within a residence poses an issue for accurately allowing user inclusion validation.

Beyond these assessments of individual components, there also exists the issue that there is no inclusive system (beyond commercial and manually configured systems) which allow for inclusion of all these components. The details of these existing system shortcomings are outlined in the table below:

1.2.1 Existing System Shortcomings

Device Type	Occupancy Detection	Smart Thermostats
Shortcoming I	Cumbersome guarantee count mechanisms	Rely on existing environment measurement tools
Shortcoming II	No heuristic most/least definition	Do not allow for sensor error recovery
Shortcoming III		Do not allow for sensor locale designation

1.3 System Conceptual Configuration and Justification

The selected component installation exists as defined within this section to allow for inclusive inter-connectivity between system components that is vacant from existing implementations. In this way it is to be defined within this section how the residential environment analysis system will operate.

A hub interface and hub application will be included as a means of establishing overall collected representation of all active environment reading sensor nodes, as well as an entire comprehensive node status representation. This is the overall means of implementing a representative intuitive control interface.

In terms of how the HA and nodes are meant to interact, a wireless connection between the two facilitated via Zigbee. This is for the fact that a mesh node interfacing configuration of all nodes to the HA will be implemented in order to allow for appropriate user environment configuration (which will be defined below in how system installation will be outlined). Zigbee will allow for more node device inclusion, more easily facilitated mesh node programming, and overall easy of implementation. [4]

In terms of mesh node sensors, a mesh of sensor nodes will interact directly to the hub. From these sensor nodes, room related actuator nodes (servo vent controllers) will be interacted, where status data will be collected by the sensor nodes and provided to the HA, and status alteration data will be collected by the sensor nodes from the HA and transmitted to the actuator nodes. In all, node clusters including sensor and actuator nodes will be maintained in a collective determined by room parameter designations attributed to the nodes. This will more easily facilitate error recovery of faulty nodes through the residence by allowing nearest capable room nodes to replace faulty nodes within another.

Sensor data and actuator data readings defined as being attributed to a room conveyed to the HA will include temperature, humidity and occupancy readings. Overall residence occupancy will be a guaranteed value and provided to the hub from dedicated occupancy nodes at entry and exit points of the residence. The difference between guaranteed overall occupancy will be maintained separately and analyzed appropriately as to allow for heuristic assessment of room occupancy based on per room occupancy sensors. This will be maintained through an at least/at most approach.

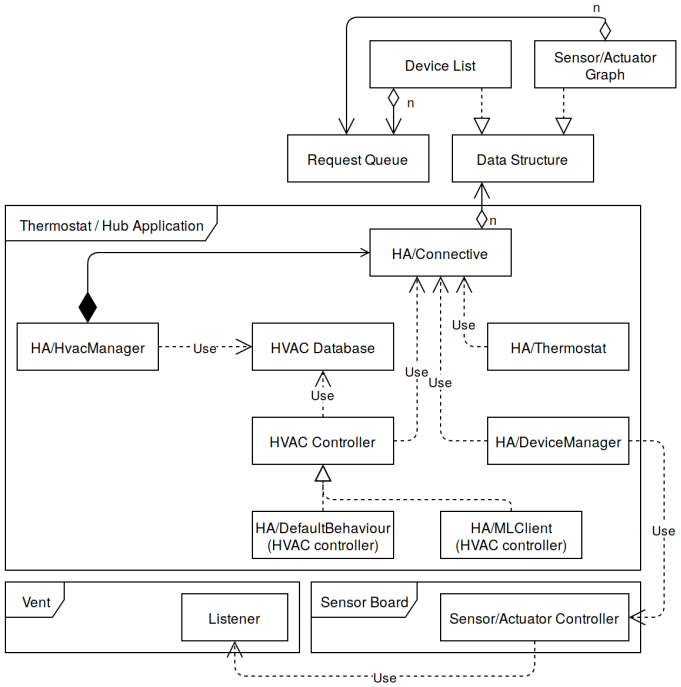
Occupancy detection was prototype tested from a purely passive infrared (PIR) sensor implementation, and allowed for the testing of guaranteed occupancy count. It was seen that the concept applied, however, that the accuracy of solely PIR sensors did not allow for the degree of accurate line of sight as needed. This is to be rectified in further implementation through the inclusion of ultrasonic PIR hybrid sensors to increase line-of-sight and accuracy.

2 Conceptual System Design

2.1 Overview

- Provide occupancy heuristic to broad range of Internet of Things (IoT) applications
- Provide sensor information to a broad range of IoT applications
- Create a framework for IoT software components to communicate
- Apply these features to service a third-party HVAC system controller

System Application Structure



The diagram on the left describes the logical structure of our entire system at the application layer. Components are grouped inside a frame representing their physical location, while objects inside HA/Connective are not inside any frame.

Our system is composed of a software suite for the thermostat (the Hub Application), the sensor board software, and very simple control logic on the actuators (vents).

Note that HA/Thermostat refers to a specific component inside the thermostat, and not the thermostat or Hub Application.

2.2 Hub Application

The Hub Application components are responsible for all computations performed on the thermostat, and includes the following functionality overall:

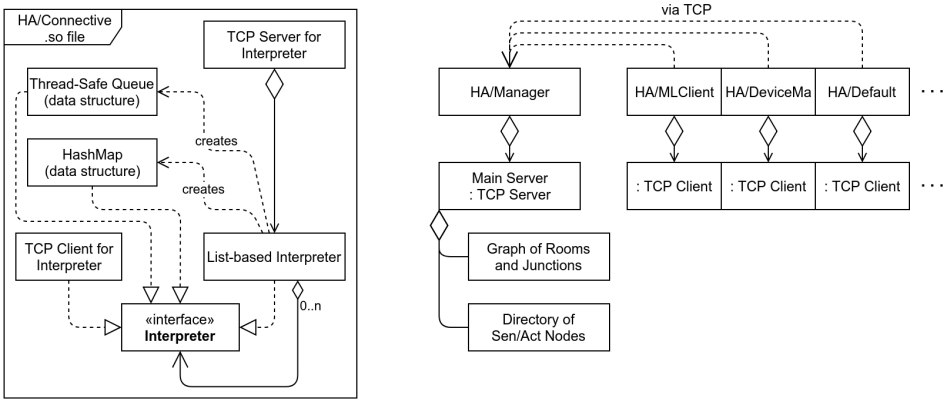
- Connection with sensor-actuator nodes (HA/DeviceManager)
- Collection of sensor data (HA/Manager)
- Controlling vents (HA/DeviceManager, HA/DefaultBehaviour, third-party HVAC controller)
- Communication with third-party controllers, ensuring valid system state, etc
- Maintaining a logical state representing known physical states (HA/Connective)

2.2.1 HA/Connective

The HA/Connective component maintains data structures (including request queues, the primary method of communication) which can be shared between components. This facilitates critical functions such as sending sensor data from HA/DeviceManager to HA/Manager, keeping the logical state of vents consistent across components, and allowing the HVAC controller to interact with the system via a single interface.

HA/Connective, rather than being a separate application, is a shared library with a server run inside HA/Manager and a client run inside other connected components. One advantage this has is that components have minimal connection logic, so switching communication protocols (for example, to use an operating system feature to optimize communication) would not require modification to other components. This also allows client components to use HA/Connective's data structures inside their own memory for convenience, or provide a remote application with debugging information.

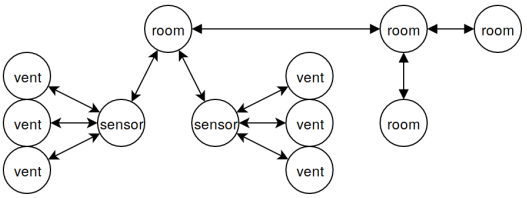
HA/Connective Class Diagram and Integration



The class diagram above shows the internal structure of HA/Connective (left) and a higher-level representation of how this integrates with the other HA components. Interaction between other components and HA/Connective's data structures happens through a common interface called "Interpreter", and a function interacting with this interface does not need to know if the data structure is local (on the component's own instance of HA/Connective's memory space) or remote (on HA/Manager's instance of HA/Connective's memory space).

The HA/Connective interpreter interface accepts a list of items, where an item can be another list or a string. The first item is compared to a hashmap of key words, where each key word is associated with another "interpreter". The subsequent items are then passed to the matching interpreter as a new list. At the end of a chain, the last implementer of "interpreter" will use the remaining data to perform a function and/or return some data. It is possible to add new keywords to the hashmap, and chain hashmaps together, or with other data structures.

Sensor/Actuator Graph Representation



The diagram above shows the data structure in HA/Connective's memory space where information about sensors and actuators is stored. Sensor nodes also aggregate a request queue which the HA/DeviceManager component will respond to. The structure of this graph is not defined by HA/Connective; instead it is defined by other components on the system.

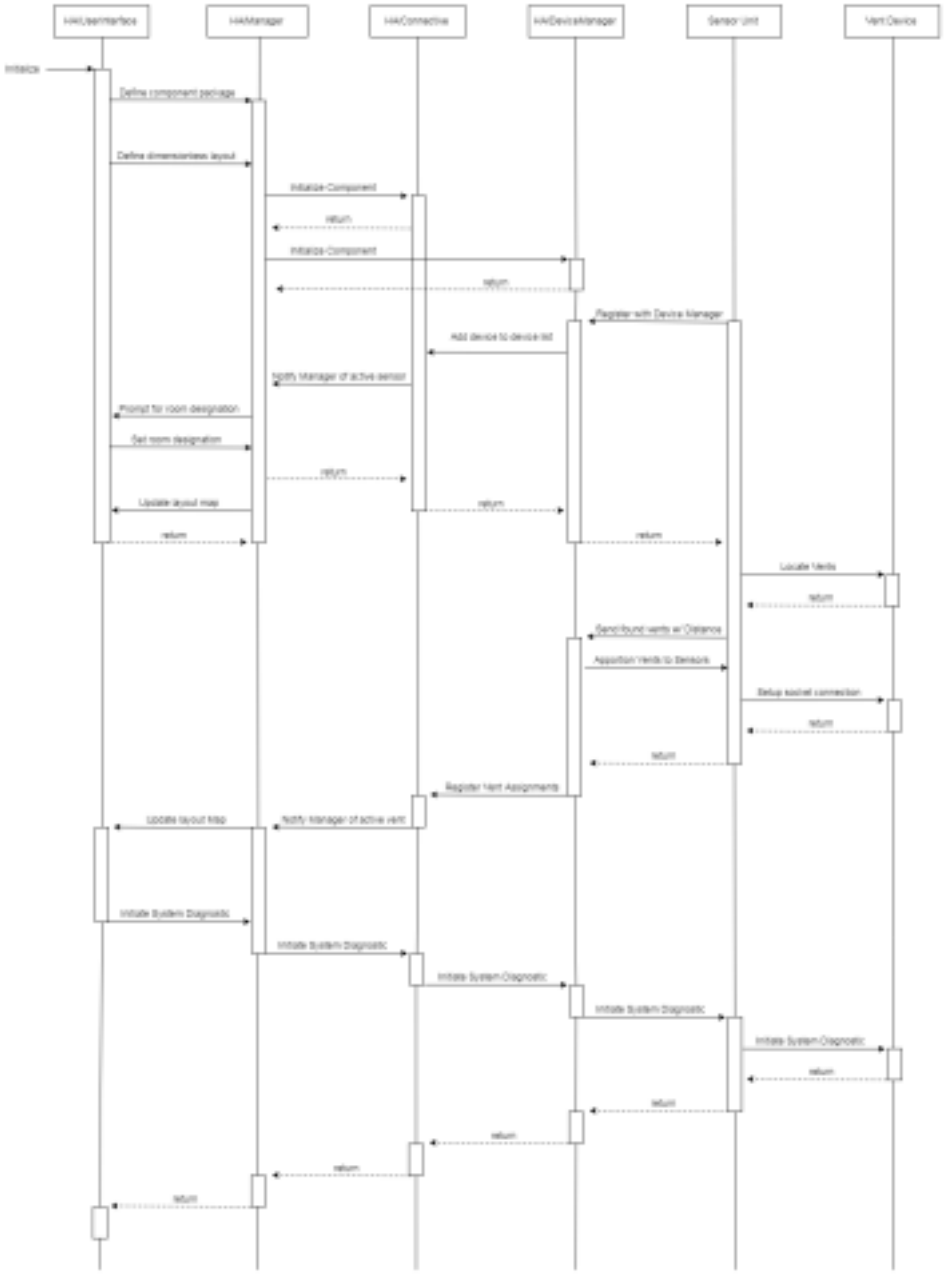
2.2.2 HA/Manager

The HA/Manager component is responsible for initializing the Hub Application and its components, selecting the correct components based on system configuration, monitoring components in case of total component failure, and storing sensor data.

When the thermostat is turned on or restarted, the HA/Manager component will start. This component will create a server instance of HA/Connective for sharing data between components, and following that start other components including the Device Manager and HVAC Controller. Depending on configuration, it will start either the wired I2C-based Device Manager (for integration testing purposes), or the wireless Device Manager (using Zigbee). It will also start either the Default Behaviour HVAC controller (for integration testing, or if HA/MLClient fails), or the MLClient (HVAC controller which uses the other group's software package).

After initializing the system, the HA/Manager will continue to monitor connected components so that it may restart them upon failure. Detection of component failure is done using heartbeat timestamps stored within HA/Connective. If a component does not send a heartbeat after a substantial amount of time, HA/Manager will terminate the respective process and restart the component.

Initialization of System through HA/Manager Interaction



2.2.3 HA/DeviceManager

The HA/DeviceManager is responsible for all functions relating to the Sensor Devices and Vent Devices. Upon being initialized by the HA/Manager, it will be placed in an 'Await Connections' mode that will prevent it from attempting to perform it's standard functionality during system setup. When a Sensor Device registers with the system, HA/DeviceManager will notify HA/Manager so that the user can be prompted to enter the room designation. Once Sensor Device setup and Vent Device installation has been completed, HA/DeviceManager will notify Sensor Devices to start begin evaluating distance from Vent Devices. Once this information is collected, HA/DeviceManager will partition Vent Devices to Sensor Devices and register the assignment with HA/Manager through HA/Connective.

HA/DeviceManager can then enter it's 'Active' mode, where it will manage a mesh-node network of Sensor Devices and Vent Devices. It will receive sensor information from the Sensor Device and it's assigned Vent Devices on regular intervals and input this information into HA/Connective. If a Sensor Device fails to communicate or a Vent Device's pressure reading is not included in its assigned Sensor Devices data package, HA/DeviceManager will begin a failure recovery process.

If a Vent Device cannot be recovered, HA/DeviceManager will alert the user and track this information, recording the failure with each dataset, until the problem can be resolved. If a Sensor Device cannot be recovered, HA/DeviceManager will trigger a process that allows other Sensor Devices to claim unclaimed Vent Devices. The user will once again be notified and the failure will be tracked and recorded.

2.2.4 HA/Console

A console application will be created for integration testing and debugging. This will act like a component of the Hub Application but will be on another device (i.e. a computer on the LAN), and provide direct access to HA/Connective. A user of the console application will be able to display the contents of data structures, inject data into request queues, and manually send heartbeat messages in behalf of any component. Some high-level commands will be implemented for these features for convenience, but all of the functionality of HA/Connective will be available via the console.

2.2.5 HA/MLClient

HA/MLClient is an HVAC controller which interacts with the ML Group's software package to provide enhanced control using predictive machine learning algorithms. The MLClient will provide an API as specified by the ML Daemon (a component from the ML team), providing the daemon with any information that it requests and relaying commands to HA/DeviceManager via HA/Connective.

For the purposes of our group, this component serves as a proof-of-concept of our systems ability to integrate third-party components which require sensor information and control of smart devices (ex: vents, computerized thermostat) in the home.

2.3 Sensor Software

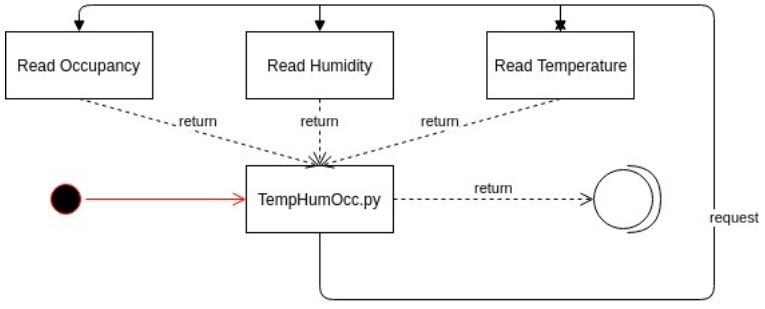
Sensor software was implemented through the means of including separate interaction mechanisms per sensor type for each environment metric required. It was found through implementation and testing that sensor protocol between sensors types in terms of data acquisition and transmission created race condition conflicts. Since an inter-integrated circuit (i2c) and general purpose input/output (GPIO) combination is required for inputting from all sensors, it was found that i2c was left queuing input while GPIO input was processed and rendered. This resulted in extremely delayed input of temperature alteration readings from the temperature sensors used (Honeywell ds1631). Furthermore, the temperature sensor input was only possible via a pre-configured C scripted library, while the occupancy sensor GPIO input was maintained by Python.

In order to solve the differing library usage requirements, it became essential to create a C to Python integration file, which allowed for Python code to call the C script which operated the temperature sensors and receive readings.

To solve the problem of race conditioning leading to lengthy queuing slowing down temperature variation readings, a multithreaded approach was created. In a Python script, calls to the temperature requesting function would thus be able to output from the ds1631 i2c sensors in real-time, and when the Adafruit DHT11 sensor used to define humidity was processed and ready to update humidity readings, along with valid input reading from the PIR sensor, they would be shown simultaneously with a new temperature reading. This allowed for continuous real-time updates from sensors.

Furthermore, the threaded approach allowed for a guarantee that should sensor accuracy need to be altered by the implementation of alternative sensors, that could be facilitated. This is good since decision to alter the humidity sensors to HIH7120 sensors, and the general purpose PIR sensors to ultrasonic PIR hybrids could pose the same types of conflict errors as previously seen. A diagram illustrating the calls to temperature, humidity, and occupancy sensor functions is provided below:

Sensor Data Reading



2.4 Sensor Board

In order to allow for prototype testing of sensors to be included within the tentative system, a Raspberry Pi 3+ was implemented. This was beneficial in allowing for the inclusion of alternative sensor types when testing as facilitated by the C to Python conversion and threading approach discussed above. While allowing for a much more available means of implementation, the Pi 3+

was deemed to be a totally inappropriate decision for final implementation due to it's excessive power consumption requirements as well as it's excessive processing requirements.

A more appropriate controller is to be decided upon for later implementation. As seen in the implementation of the sensor components, C script utilization will be very much achievable. This is for the fact that Python is C based [5]. This will allow for selection of virtually any controller that may fit purposes of the project.

When selecting an alternative implementation controller, it will be necessary to utilize current prototyping implementation to allow for testing facility that incorporates reading of computation consumption. Once acquired, it will be possible to appropriate more accurately entail an appropriate controller decision based on specification. This will be combined with research indicating controller size and power consumption, since it will be necessary to select a controller that is compact and usage efficient. Most nodes will require battery power, since placement is going to require areas that may not necessarily allow for AC power line connection. Furthermore, aesthetic maintenance should be adhered to in order to achieve the pretense of sightly agreeable installation.

2.5 Actuators (vents)

Vent actuation was initially an issue trying to be decided upon for the fact that allowing for individual transmission and control of actuator controllers would be difficult considering control facets and implementation for a multi-room environment. To solve this, it was decided that sensor actuation would be facilitated through system node meshing, wherein vent actuators would be controlled and communicated to by sensor controllers. Once residence mapping is appropriate, a much more simplistic means of ensuring sensor readings render required actuation appropriately.

Actuator to be used per vents will depend upon vent sizing and composition material. Standardized vents will be the primary selection for implementation, since they will obviously be the most likely to be widely available in both from both initial configuration and production aspects.

Vent controllers will thus rely purely on control requests from sensor nodes affiliated. HA control will be conveyed from itself to the corresponding sensor node, and relayed from the sensor node to the actuator controller.

3 Definition of Integration Tests

This section outlines identified requirements for integration testing. Phase 2 testing will be outlined in detail. Phase 3 testing will primarily be dependent on performance results of the system following phase 2. For this reason, only a few of the integration tests for that phase will be outlined in this document. As Phase 2 and 3 of our project progresses, these test cases will be refined and then executed when the constituent components are prepared for integration.

3.1 Phase 2 Testing

Integration testing will be required for a number of components, including those located on the Hub device and IoT devices which will communicate with the Hub or one another. This section will outline the groups of components that will need to be integrated and tested. For each subsection, a number of integration test cases will be outlined. The following outlines integration testing required to deploy a full system test. This will include the following components:

- HA/Manager
- HA/Connective
- HA/Default
- HA/DeviceManager
- Sensor Devices
- Vent Devices

3.1.1 HA/Manager and HA/Connective

Integration testing between HA/Manager and the HA/Connective component which will facilitate inter-process communication is required to be completed first as it will enable integration testing between HA/Manager and all other processes.

HA/Connective will hold the sensor/actual graph, list of devices, and component heartbeats in its memory space. It is critical that HA/Manager is able to access these data in order to perform properly. Integration testing between HA/Connective and HA/Manager will check for the following:

- HA/Manager reports correct state for an arbitrary heartbeat situation
- HA/Manager's changes to the graph are reflected on another client of HA/Connective (i.e. the debugging console)

3.1.2 HA/Manager and HA/Default

Integration testing of HA/Manager and HA/Default will be performed to enable integration testing of components that will be governed by HA/Default through HA/Manager. The following will need to be confirmed through integration testing:

- HA/Manager is capable of initializing HA/Default

- HA/Manager is capable of monitoring HA/Default through HA/Manager's instance of HA/Connective
- HA/Default is capable of sending information through HA/Manager's instance of HA/Connective
- HA/Manager is capable of reading commands sent by HA/Default to other components through HA/Manager's instance of HA/Connective
- HA/Manager is capable of handling failure recovery for HA/Default

3.1.3 HA/Manager and HA/Thermostat

Integration testing between HA/Manager and HA/Thermostat will involve initializing HA/Default and HA/Thermostat and allowing HA/Manager to monitor both processes. The following will need to be confirmed through integration testing:

- HA/Manager is capable of initializing HA/Thermostat
- HA/Manager is capable of monitoring HA/Thermostat through HA/Manager's instance of HA/Connective
- HA/Thermostat is capable of polling for HVAC commands through HA/Manager's instance of HA/Connective
- HA/Manager is capable of handling failure recovery of HA/Default

3.1.4 HA/Manager and HA/DeviceManager

Integration testing between HA/Manager and HA/DeviceManager will be performed comparably to the integration testing of HA/Manager and HA/Thermostat. The following will need to be confirmed through integration testing:

- HA/Manager is capable of initializing HA/DeviceManager
- HA/Manager is capable of monitoring HA/DeviceManager through HA/Manager's instance of HA/Connective
- HA/DeviceManager is capable of registering sensors and vent assignments in the graph representation of the house layout through HA/Manager's instance of HA/Connective
- HA/DeviceManager is capable of sending sensor data through HA/Manager's instance of HA/connective
- HA/Manager is capable of committing sensor data sent by HA/DeviceManager to a database
- HA/DeviceManager is capable of receiving commands from an instance of HA/Default through HA/Manager's instance of HA/Connective
- HA/Manager is capable of handling failure recover of HA/Default

3.1.5 HA/DeviceManager and Sensor Devices

Integration testing between HA/DeviceManager and a Sensor Device unit will be performed to ensure that registration and communication are functioning correctly. The following will need to be confirmed through integration testing:

- HA/DeviceManager is capable of handling registration of Sensor Devices while in 'Await Connections' mode
- Sensor Devices are capable of registering themselves with HA/DeviceManager
- HA/DeviceManager is capable of receiving Vent Device assignments
- Sensor Devices are capable of sending Vent Device assignments
- HA/DeviceManager is capable of receiving sensor information from Sensor Devices
- HA/DeviceManager is capable of monitoring Sensor Devices
- HA/DeviceManager is capable of initiating failure recover protocols for Sensor Devices

3.1.6 Sensor Devices and Vent Devices

Integration testing between Sensor Devices and Vent Devices may be performed before or after the previous section. The following will need to be confirmed through integration testing:

- Sensor Devices are capable of determining distance from Vent Devices
- Sensor Devices are capable of creating Zigbee connections with Vent Devices
- Vent Devices are capable of receiving commands to rotate the motor from Sensor Devices

3.1.7 HA/DeviceManager, Sensor Devices and Vent Devices

Integration testing between these three components will be performed after the two previous sections. In order to facilitate integration testing, at least two sensor devices and at least three vent devices will be required. The following will need to be confirmed through integration testing:

- Sensor Devices are capable of determining distance to vent devices and communicating distance with HA/DeviceManager
- HA/DeviceManager is capable of partitioning Vent Devices amongst Sensor Devices and communicating this assignment to Sensor Devices
- Sensor Devices are capable of receiving Vent Device assignment and generating connections with said Vent Devices
- HA/DeviceManager is capable of communicating with Vent Devices through the assigned Sensor Device

3.2 Phase 3 Testing

Phase 3 will revolve around adding additional utilities and developing the capability to integrate with third-party HVAC managers and other third-party applications. Other integration testing may be required if sufficient changes are necessitated by the results of our Phase 2 systems test.

3.2.1 HA/Manager and third-party HVAC Manager

This integration will be performed by developing an application called HA/MLClient which will manage a connection to a machine learning model created by our ML sister team. This integration will also cover all integration test requirements of other third-party applications. The following will need to be confirmed through integration testing:

- HA/DeviceManager is capable of including the third-party application in the initialization process
- HA/DeviceManager is capable of monitoring the third-party application
- HA/DeviceManager is capable of standing down HA/Default if a third-party application is included
- HA/DeviceManager is capable of performing recovery of the third-party application
- HA/DeviceManager is capable of restarting HA/Default if the third-party application can't be recovered.

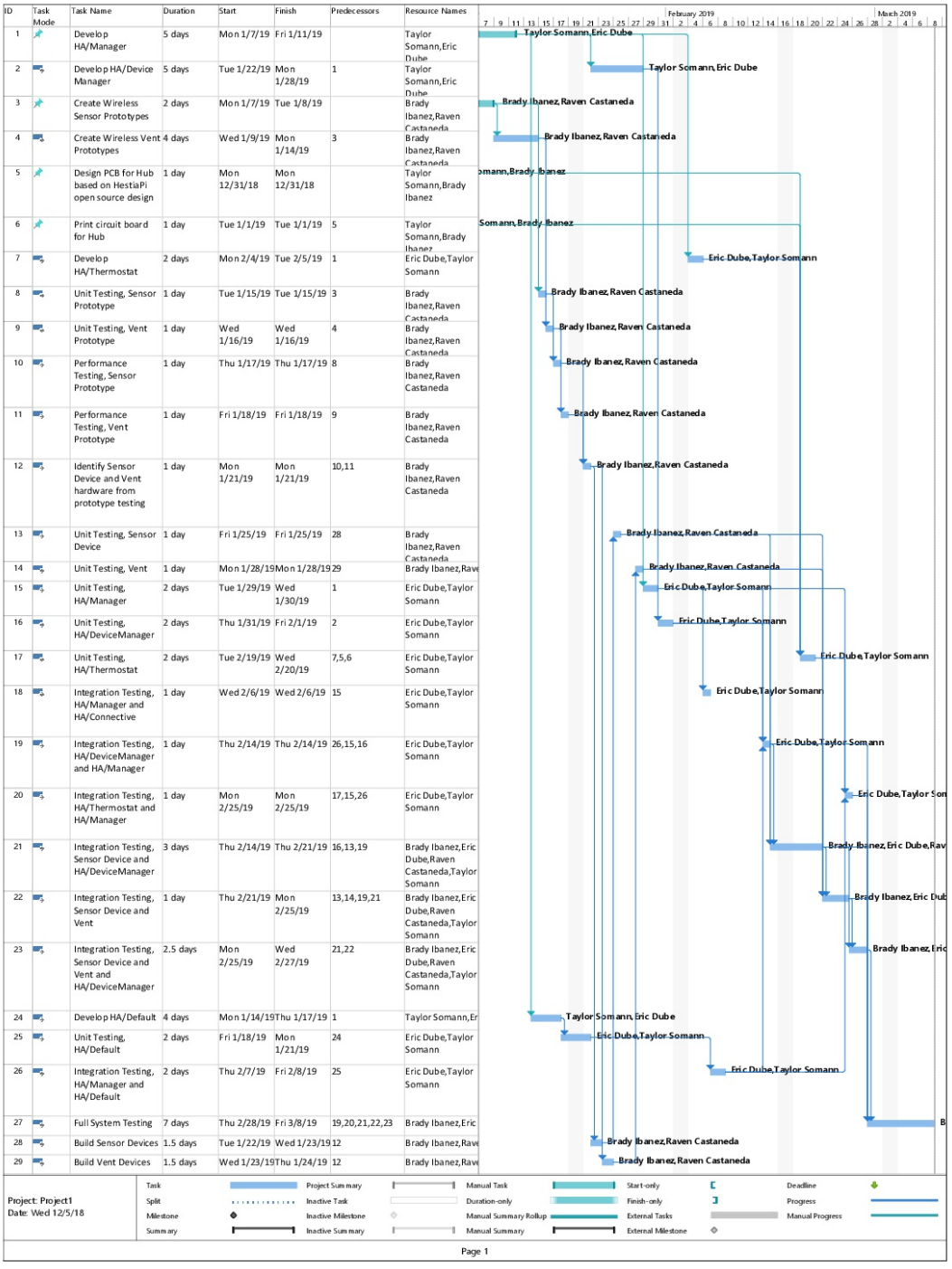
4 Estimated Cost of the Project

For the current implementation of Active Vents, the current prototype is designed around a by room basis with small specifications on a per floor basis as well. With the Active Vents prototype, it utilizes on a per room basis one PIR sensor to maintain the at least heuristic and one temperature sensor to monitor the current temperature of the room. At entry-points into the house, six PIR sensors will be set to track entry and exit. Lastly each sensor set currently connects and runs using a raspberry pi 3 and wired i2c communication using i2c busses. So the overall material list for running the Active Vents for a house is six PIRs for entryways into the house, one raspberry pi 3, one PIR and one temperature sensor, one humidity sensor per floor and wires with i2c busses and breadboards.

Details	Rate($/hr)	Hours	Amount
Prototype Implementation			
Sensor Configuration			
HC-SR501 PIR Motion Sensor			39.97
Breadboards			4.99
Wires			4.99
I^2C 16-Channel Input/Output Port Expander (MCP23017)			3.19
HIH7120 (Humidity Sensor)			7.19
Raspberry Pi 3B+			54.99
DS1631+-ND (Temperature Sensor)			5.27
Shipping Cost			21.99
Implementation Subtotal			142.58
System Software Development			
Fourth Year Software Engineering Students	35.00	1,280	44,800.00
System Software Development Subtotal			44,800.00
Total Projected Project Cost			44,942.58

For future implementations, the wiring and i2c communication network will be replaced with ZigBee communication networks and a more cost-effective micro-controller setup will be implemented to help reduce cost. Additionally the heavy cost incurred by multiple PIRs in the prototype will hopefully be replaced by other combination sensors in order to improve accuracy as well as value for that cost increase/maintenance. Lastly, for the manpower costs, besides light installation in the future which does not require much manpower to install, the cost will be low.

Phase II Gantt Chart



ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		Develop HA/Manager	5 days	Mon 1/7/19	Fri 1/11/19		Taylor Somann,Eric Dube
2		Develop HA/Device Manager	5 days	Tue 1/22/19	Mon 1/28/19	1	Taylor Somann,Eric Dube
3		Create Wireless Sensor Prototypes	2 days	Mon 1/7/19	Tue 1/8/19		Brady Ibanez,Raven Castaneda
4		Create Wireless Vent Prototypes	4 days	Wed 1/9/19	Mon 1/14/19	3	Brady Ibanez,Raven Castaneda
5		Design PCB for Hub based on HestiaPi open source design	1 day	Mon 12/31/18	Mon 12/31/18		Taylor Somann,Brady Ibanez
6		Print circuit board for Hub	1 day	Tue 1/1/19	Tue 1/1/19	5	Taylor Somann,Brady Ibanez
7		Develop HA/Thermostat	2 days	Mon 2/4/19	Tue 2/5/19	1	Eric Dube,Taylor Somann
8		Unit Testing, Sensor Prototype	1 day	Tue 1/15/19	Tue 1/15/19	3	Brady Ibanez,Raven Castaneda
9		Unit Testing, Vent Prototype	1 day	Wed 1/16/19	Wed 1/16/19	4	Brady Ibanez,Raven Castaneda
10		Performance Testing, Sensor Prototype	1 day	Thu 1/17/19	Thu 1/17/19	8	Brady Ibanez,Raven Castaneda
11		Performance Testing, Vent Prototype	1 day	Fri 1/18/19	Fri 1/18/19	9	Brady Ibanez,Raven Castaneda
12		Identify Sensor Device and Vent hardware from prototype testing	1 day	Mon 1/21/19	Mon 1/21/19	10,11	Brady Ibanez,Raven Castaneda
13		Unit Testing, Sensor Device	1 day	Fri 1/25/19	Fri 1/25/19	28	Brady Ibanez,Raven Castaneda
14		Unit Testing, Vent	1 day	Mon 1/28/19	Mon 1/28/19	29	Brady Ibanez,Rave
15		Unit Testing, HA/Manager	2 days	Tue 1/29/19	Wed 1/30/19	1	Eric Dube,Taylor Somann
16		Unit Testing, HA/DeviceManager	2 days	Thu 1/31/19	Fri 2/1/19	2	Eric Dube,Taylor Somann
17		Unit Testing, HA/Thermostat	2 days	Tue 2/19/19	Wed 2/20/19	7,5,6	Eric Dube,Taylor Somann
18		Integration Testing, HA/Manager and HA/Connective	1 day	Wed 2/6/19	Wed 2/6/19	15	Eric Dube,Taylor Somann
19		Integration Testing, HA/DeviceManager and HA/Manager	1 day	Thu 2/14/19	Thu 2/14/19	26,15,16	Eric Dube,Taylor Somann
20		Integration Testing, HA/Thermostat and HA/Manager	1 day	Mon 2/25/19	Mon 2/25/19	17,15,26	Eric Dube,Taylor Somann
21		Integration Testing, Sensor Device and HA/DeviceManager	3 days	Thu 2/14/19	Thu 2/21/19	16,13,19	Brady Ibanez,Eric Dube,Raven Castaneda,Taylor Somann
22		Integration Testing, Sensor Device and Vent	1 day	Thu 2/21/19	Mon 2/25/19	13,14,19,21	Brady Ibanez,Eric Dube,Raven Castaneda,Taylor Somann
23		Integration Testing, Sensor Device and Vent and HA/DeviceManager	2.5 days	Mon 2/25/19	Wed 2/27/19	21,22	Brady Ibanez,Eric Dube,Raven Castaneda,Taylor Somann
24		Develop HA/Default	4 days	Mon 1/14/19	Thu 1/17/19	1	Taylor Somann,Er
25		Unit Testing, HA/Default	2 days	Fri 1/18/19	Mon 1/21/19	24	Eric Dube,Taylor Somann
26		Integration Testing, HA/Manager and HA/Default	2 days	Thu 2/7/19	Fri 2/8/19	25	Eric Dube,Taylor Somann
27		Full System Testing	7 days	Thu 2/28/19	Fri 3/8/19	19,20,21,22,23	Brady Ibanez,Eric
28		Build Sensor Devices	1.5 days	Tue 1/22/19	Wed 1/23/19	12	Brady Ibanez,Rave
29		Build Vent Devices	1.5 days	Wed 1/23/19	Thu 1/24/19	12	Brady Ibanez,Rave

5 IoT/ML Team Interaction

The interaction between the two groups of this conglomerate project, the IoT and ML teams, has been effective in facilitating interactive assuredness of implementation capability. In terms of the IoT team's ability to gain advantageous reference from efforts of the ML team, the ML team's contracted creation of the contrived house model for a confined means of guaranteed environment control allowed for reference of sensor capacity to react appropriately to environmental fluctuation. Temperature and humidity alterations were thus controllable to high degree of accuracy, and would allow for guaranteeing appropriate response of the sensors.

The ability to vary composition of the home layout also helps to understand how difference in internal construction will effect sensor reading deviations between adjacent points in a residence.

Team members actively collaborated on the configuration of these components by meeting on multiple occasions in configure and implement the mechanisms that allowed for the control and maintenance of the model. This included meeting to create a simulation HVAC system, using a blow dryer re-purposed with relay current control allows for temperature and humidity fluctuation. To control where in the model this might occur, vent in controls were implemented to prevent or allow blow dryer input to certain aspects of the model through intake inserts at certain points throughout the frame. This allows for actual HVAC simulation similar to that which would exit in a genuine residence.

Through the use of i2c wired implementation of sensor connectivity, designation of multiple sensors at different points allowed for the simulation of state aware room designations. The wired implementation allowed for an assured base on reconfiguration and sensor mobility. i2c bussing also allowed for system reconfiguration and designation to attempt a preliminary manifestation of the modularity to HA will be implementing in the event of node reconfiguration.

For Phase II it will be the intention to implement the same kind of system rendered in the model within an actual homestead environment. The phase I implementation within the model provided the necessary insight to preface the type of system interaction to be expected on a larger scale.

6 Contribution Matrix

While contribution detailing does not require strict adherence in the sense that any given contribution head may not apply effort outside of their given effort set, the following work concentration divide means to provide governance of developer focus within the project.

Raven C. – Head of Sensors Research, Research Assistant, Sensor Software Assistant, Report Assistant, and Sensor Implementation Assistant

- Raven will be responsible for assisting in sensor implementation, additional sensor research, and documentation for reports.

Taylor S. – Project Manager, Hub Designer, Head of Software

- Taylor will be responsible for heading the project, keeping the project on time, and designing the main hub for the system.

Brady I. – Head of Sensor Software, Head of Report Organization, Circuit Designer

- Brady will be responsible for the implementation and synchronization of Bluetooth Low Energy (BLE) sensor integration to allow for the capacity of the system to infer physical recognition of the environment governed.

Eric D. – Head of Middleware, Head of Research

- Eric has developed the HA/Connective component and tested its use from a program written in Python.
- Eric will be responsible for integration with HA/connective and other components as well integrating the ML Team's software package with the rest of our system.

References

[1] NEST. Nest temperature sensor. [Online]. Available: https://nest.com/ca/thermostats/nest-temperature-sensor/overview/

[2] EcoBee. Ecobee 4. [Online]. Available: https://shop-ca.ecobee.com/products/ecobee4

[3] Lutron Inc. Occupancy/vacancy sensor design and application guide. [Online]. Available: http://www.lutron.com/TechnicalDocumentLibrary/3683197.pdf

[4] I. F. C. M. Alexandru LAVRIC, Valentin POPA, "Performance evaluation of tree and mesh zigbee network topologies used in street lighting control systems," *Przeglad Elektrotechniczny*, vol. 89, 2013. [Online]. Available: https://pdfs.semanticscholar.org/9dab/c9404d883e13974944cd9335091fe4a44e19.pdf

[5] Titus Brown. Wrapping c/c++ for python. [Online]. Available: https://intermediate-and-advanced-software-carpentry.readthedocs.io/en/latest/c++-wrapping.html