

CtrlFlow

An IoT Modular Thermo-System



University of Central Florida

Department of Electrical Engineering and Computer Science

Dr. Wei & Dr. Richie

Senior Design II

Group 17

Muhamad Elassar, CpE

Yoseph Hassan, CpE

Joseph Mansy, CpE

Vinh Tran, EE

Table of Contents

1. Executive Summary	1
2. Project Description	3
2.1 Project Plan	3
2.2 Objectives & Constraints	4
2.2.1 Motivation	5
2.2.2 Constraints & Restrictions	6
2.3 Requirements Specifications	7
2.4 Design Options	8
2.4.1 Atmel MCU with wireless peripherals	9
2.4.2 Thunderboard development platform by Silicon lab	9
2.4.3 Launchxl-CC26X2R1 by Texas Instrument	9
2.4.4 NodeMCU by Espressif Systems	9
2.5 Block Diagrams and Illustrations	10
2.5.1 Hardware Software Integration	10
2.5.2 Hardware Communication	10
2.5.3 Application Control Flow	11
2.6 Role Breakdown	12
2.7 Marketing and Engineering Requirements	12
3. Research and Part Selection	14
3.1 Existing Designs and Products	14
3.2 Relevant Technologies	18
3.2.1 Peltier Effect & Thermoelectric Cooling	18
3.2.2 Voltage Regulators	22
3.2.2.1 Linear voltage regulator	22
3.2.2.2 Switching voltage regulator	23
3.2.2.3 Comparison	24
3.2.3 Actuator	25
3.2.3.1 Servo motors	25
3.2.3.2 Stepper motors	26
3.2.3.3 Comparison	28
3.2.4 H-Bridge	28
3.2.5 Internet of Things (IoT)	32
3.2.6 Mobile Application Development	35

3.2.6.1 Android & iOS	35
3.2.6.1.1 Java vs Swift	36
3.2.6.2 React Native	39
3.2.6.3 Flutter	41
3.2.7 Network topology	47
3.2.7.1 Star topology	48
3.2.7.2 Mesh topology	50
3.2.7.3 Comparison	51
3.2.8 Networking Standards	51
3.2.8.1 Bluetooth Mesh	52
3.2.8.2 Zigbee	53
3.2.8.3 Open Thread	57
3.2.8.4 Mesh protocol comparison	58
3.2.8.5 LoRa	60
3.2.9 Embedded development	63
3.2.9.1 Arduino	63
3.2.9.2 TI Launchpad & Simplelink platform	65
3.2.9.3 SiliconLab Bluetooth mesh	66
3.2.9.4 Comparison and conclusion	67
3.3 Part Selection	67
3.3.1 MCU	67
3.3.1.1 Microchip megaAVR Family - Arduino - ATmega4809	68
3.3.1.2 Microchip SAMD Family - Arduino - SAMD21	68
3.3.1.3 TI SimpleLink Family - Launchpad - CC2652RB	68
3.3.1.4 SiliconLabs Gecko Family - ThunderBoard - EFR32BG22	69
3.3.1.5 Espressif ESP Family - NodeMCU - ESP8266	70
3.3.2 Wi-Fi Transceiver	71
3.3.2.1 ESP32-WROOM	72
3.3.2.2 NINA-W102	72
3.3.3 ISM Band Transceiver	73
3.3.3.1 HC-12	73
3.3.3.2 LoRa - RFM95W	74
3.3.3.3 RFM69HCW	74
3.3.4 IoT Cloud Services	75
3.3.4.1 Amazon Web Services	75
3.3.4.2 Microsoft Azure	77

3.3.4.3 Google Cloud	78
3.3.5 Display	80
3.3.5.1 Character LCD - ACM1602K	82
3.3.5.2 Color LCD - Waveshare 2.4" LCD	82
3.3.5.3 E-Paper Display - E2154FS091	82
3.3.5.4 No Display	82
3.3.6 Temperature Sensor	84
3.3.6.1 Thermocouple	84
3.3.6.2 Low Voltage Analog Temperature Sensor - TMP235A4DCKR	84
3.3.6.3 Low Voltage Digital Temperature Sensor - MAX31820	84
3.3.7 Motor	85
3.3.7.1 Stepper Motor - 28BYJ-48	86
3.3.7.2 Linear Actuator	86
3.3.7.3 Servo Motor - SG90	86
3.3.8 Power	87
Wired Power	87
Alkaline battery	87
Li-ion Battery	88
3.3.9 DC Power Regulators	89
3.3.9.1 12v to 3.3v Buck Regulator	89
3.3.9.1.1 TPS62160DGKR	89
3.3.9.1.2 TPS82150SILR	90
3.3.9.1.3 TLV1117-33CDCYR	90
3.3.9.2 LiPo/5v to 3.3v Regulator	91
3.3.9.2.1 TPS63031DSKR	91
3.3.9.2.2 TPS63001DRCR	92
3.3.9.2.3 LTC3553	92
3.3.9.2.4 TLV75533PDBVR	93
3.3.9.3 LiPo to 5v Boost Regulator	94
3.3.9.3.1 TPS81256SIPR	95
3.3.9.3.2 TPS61256YFFR	95
3.3.9.3.3 LMR62014XMF	95
3.3.10 Battery Charger	96
3.3.10.1 MCP73831	97
3.3.10.2 BQ24230RGTR	97
3.3.10.3 BQ24195LRGER	97

3.3.10.4 LTC3553	97
3.3.11 Transistors	98
3.3.11.1 2N4401	98
3.3.11.2 ZXTP5240F	98
3.3.11.3 NSS40201LT1G	99
4. Design	100
4.1 Relevant Circuits	100
4.1.1 Voltage Ladder	100
4.2 Prototyping	101
4.2.1 Mock HVAC System	102
4.2.2 Embedded Hardware Testing	104
4.2.2.1 Radio Communication	104
4.2.2.2 WiFi Communication	105
4.3 3D printing	105
4.4 Valves	106
4.5 Schematics	107
4.5.1 Central Hub	108
4.5.2 IoT Sensor	111
4.5.3 IoT Vent Damper	113
4.6 PCB	114
4.6.1 Central Hub	115
4.6.2 IoT Sensor	116
4.6.3 IoT Vent Damper	117
4.7 Application Design	117
6. Budget and Financing	122
7. Project Milestones	123
Appendix A: Copyright Permission	124
Permission to use images from Amazon Web Services	124
Permission to use images from Microsoft Azure	124
Permission to use images from Google Cloud	125
Permission to use images from DigiKey	125
Permission to use images from Waveshare	126
Permission to use images from Texas Instruments	126
Permission to use images from Analog Devices	126

Permission to use images from Flair	127
Permission to use images from Keen	127
Appendix B: References	128
Similar Projects	128
Internet of Things	128
Mobile Application Development	128
Appendix C: Datasheets	130
Microcontroller Units	130
Wi-Fi Transceivers	130
ISM Band Transceivers	130
Display Units	131
Temperature Sensors	131
Motors	131
Power	131
DC Power Regulators	132
Battery Chargers	133
Transistors	134

1. Executive Summary

Living in Florida has its difficulties, one of them being temperature/climate control in the home. The motivation of this project is the trouble which arises when a home does not distribute cooling between rooms evenly and this results in imbalances in the temperature between rooms. Oftentimes you try to set the thermostat to a certain temperature and what occurs is not what you were expecting. Certain rooms will simply not reach the temperature desired by the user as a result of the current limitations of modern HVAC systems. Many rooms have different temperatures and this creates a variability in climate from what the user may have been expecting. Modern AC systems are quite basic in their design. They typically cool on full blast until the desired temperature is reached and then turn off. The AC system continually performs this routine throughout the day in order to maintain the desired temperature. However this method of cooling and heating isn't comprehensive and typically doesn't heat every room to the desired temperature. In addition to this, certain users may have different temperature needs than others and one of our goals is to allow for different temperature zones between different rooms.

The issue of imbalance of temperature among different zones in the home is a very common issue which is running rampant, especially in areas such as Florida with extreme heat and humidity. This fundamental issue associated with these modern air conditioning systems must be addressed in order to result in a more comfortable experience for homeowners. The very way in which air conditioning systems are operated is fundamentally flawed and doesn't provide an adequate performance for the end users. When setting the temperature of the house, it is done with respect to the location of the thermostat and of course this results in colder temperatures near the location of the thermostat. By eliminating the centralized nature of a thermostat in the home, it will allow for better balance of cooling across all of the regions in the home. The disadvantages of these old designs results in an end product which suffers from too many flaws and inconsistencies.

The proposed system would need to eliminate the centralization of one temperature controlling unit in order to keep the blower running to the rooms that need it. For the rooms and regions which don't require cooling/heating, the vents will close and prevent unnecessary further changes in temperature. Once the desired temperature is reached the vent will close and the system will tend to other regions via opening/closing vents and blowing hot/cold air. This constant maintenance of desired temperatures for different zones requires a system which can periodically check on these areas and address changes in temperature which may be unfavorable. This requires the use of a more distributed design which can scale across several rooms and avoid the issue associated with centralized heating/cooling. Achieving such a design will allow for a more comfortable and consistent experience for users attempting to reach desired temperatures in their respective buildings.

Another concern which is often overlooked when it comes to temperature control in the home is the process of making your way to the thermostat and changing the temperature with a button or unnecessarily irritating mechanism. Oftentimes these devices break and aren't very reliable. This creates many points of failure and inability to change the temperature when it matters most. In addition to the unreliability of this physical medium are the times when users are simply unable to make their way to the location of the thermostat. Perhaps the user has a disability and would like the option of remotely changing the temperature to their desired settings. And everybody has had that night where they felt cold and had to practically drag themselves out of bed in order to make the house a bit warmer. What if this process could be simplified and made much more convenient for users? What if users could simply change the temperature from the comfort of their own bed? The system we are envisioning would alleviate all of these unnecessary trips to the thermostat just to change the temperature of your house and help you save time in the process.

There are other products which provide similar benefits such as the Keen and Flair smart vents however these products are extremely expensive and typically poorly reviewed. Each smart-vent from these manufacturers also costs anywhere from \$80-\$135 per vent depending on the dimensions of the vent in question. This isn't including the price of the "Flair Puck" which is an additional \$119 and must be placed in every room requiring a smart vent. This results in an astronomical cost in order to integrate the smart vent system in an average home. Our product will also require the installation of specially made vents and thermostats but targeting a much lower price point. Additionally neither of the current competitors provide a product to actually control the central AC unit.

2. Project Description

Living in regions with high humidity and extreme temperatures requires fine tuned air conditioning systems in order to create a positive and comfortable experience for the end users. The issue of inconsistency of temperature among different regions in the home is often underappreciated as users don't realize the severity of the issue. Temperature sensitive environments can be compromised as a result of improper heating/cooling to the respective area. In addition to this, the unnecessary activity required to simply alter the temperature is something which can be improved and made significantly more convenient.

We are looking to alleviate the many problems of modern air conditioning systems by creating an IoT system which controls the heating and cooling of independent rooms. One of the goals associated with this project is to develop an accurate independent temperature controlling technology which is relatively easy to use and can be controlled via a smartphone or computer application (Fig. 1.2, 1.4) & (Table 1.1, Specification 6). Certain rooms are too hot for their respective individuals while others are much too cold. Our product will make it easy to control the temperature in an individual's respective room without the need for physically going to the thermostat to alter the temperature and affect others in the household. Our product will also make it more comfortable for each individual as they can tailor the temperature to their needs.

A typical thermostat is also only controlled from one spot in the home, our approach will automatically control the central AC to reach the desired temperatures set by the user for each room. This will allow for the individuals to control temperatures to their liking assuming they are connected to the internet. The application interface will allow for the convenience of changing the temperature of any room remotely, resulting in improved portability as opposed to manually changing the temperature via the central thermostat (Table 1.1, Specifications 5 & 7). Our product will also be battery powered and relatively small in size in order to avoid routing many cables through walls. The specific details associated with these features are displayed in (Table 1.1, Specifications 1-4).

The Smart Air Flow project is not meant to be a device targeted towards the high-end market but rather a system which can be provided to the average household. This product will be easily attached to vents with IoT system integration and allow for wireless connectivity between devices and an online database (Fig. 1.3). One of the goals of our project is to show that achieving a level of independent climate control shouldn't be extremely expensive.

2.1 Project Plan

Each room or zone will have a temperature sensor along with a microcontroller that connects back to a main mcu which determines whether or not a room is hot or cold. This process will be automated via the microcontrollers communicating and sending the

information to the main microcontroller (Fig. 1.3 - 1.4). Essentially each room will have its own climate based on the input specifications of the user.

Since this project is relatively complex and incorporates many different technologies from different engineering disciplines the project will have to be set up such that each group member can be as efficient as possible. This requires a high level of coordination between group members and a significant amount of flexibility between group members and roles. The project roles will be assigned based on the strengths of each team member. The strengths and capabilities of each group member will be assessed based on their previous coursework, previous experience and demonstrated competency in certain technologies.

Muhamad and Yoseph will be responsible for the mobile application and software development associated with the project. The central hub and embedded design of the main microcontroller will be under the control of Joseph and Muhamad. Finally the Valves, Sensors, and PCB design will be managed by Vinh and Joseph. These roles are consistent with strengths associated with each team member. The breakdown of project roles and assignments can be found in (Fig. 1.5). We have also weighed our options in terms of cost, usage, and features in a House of Quality diagram (Fig. 1.1). In addition to this we've researched the costs associated with the required components and have mapped out planned milestones (Tables 1.2 & 1.3).

Our group will be demonstrating and testing the project in a small-scale representation in place of a full-scale implementation in a real home as it is simply not feasible given the expected presentation environment. A small-scale representation will simply provide a proof of concept of what could possibly be achieved in a full-scale model. Our model will consist of two separate rooms divided by a wall and will be interconnected with ducts leading to each room. Each duct will be fitted with a custom vent which will open and close depending on the temperature specified by the user.

2.2 Objectives & Constraints

This section will outline the main objectives associated with this project and what we hope to accomplish by the end of the design process. This section will also delve into the motivation behind this project and what led to conception of the idea and product proposed. In addition, this section also serves as a starting point to the project and provides a basis as to the reason in pursuing experience associated with this capstone design project. The other topics which will be covered throughout this section include the benefits which are driving the motivation behind this project as well.

In addition to this, this section will also cover the various constraints and restrictions associated with this project. These include but are not limited to power consumption, time constraints, size restrictions, economic, ease of use constraints, and many others which will be discovered through testing and further development of the final

implementation. This section will mainly delve into these restrictions in order to design an IoT system which is consistent with the plan for this project.

Finally this section will also cover the motivation of this project and what we as a group hope to gain from participating in this project. In addition to this the relevant experience we hope to attain through independent research and testing of different technologies and that of engineering methodologies. The other main experience which is to be acquired during the duration of this project is the ability to work cohesively as a team towards a common objective.

2.2.1 Motivation

The motivation of the overall project is to apply all of the engineering skills and techniques acquired while at University of Central Florida in order to solve a modern day problem. In addition to this, another motivation of this group is to gain valuable knowledge which may be useful or applicable to future jobs and careers in the Electrical & Computer engineering industry.

Our group, as engineers, also hope to gain experiences in new and emerging technologies and expand our skillset to certain areas of technology which haven't been covered throughout our curriculum. It is these technologies and innovations which can provide better solutions to common issues faced in this day and age.

In addition to this, the experience gained through researching and testing new technologies can be similar in scope to a technological elective which may haven't been available during our time at the University of Central Florida. In this way, we can incorporate our skills gained through our Electrical and Computer engineering curriculum and also gain new experience in more interesting newer technologies.

Furthermore, learning to work with other group members and learning proper team dynamics is something which must be experienced before graduating with an undergraduate degree in engineering. It will provide much needed integral experience which will be very much relevant in the engineering industry. This experience will allow for better integration into the future workplace or careers of our group members. In addition to this, when we face similar situations in the future in which a group must work together towards a common goal we will be ready to face the challenge.

The main objective of this project is to engineer a solution to imbalance of temperature in homes and provide heating/cooling which is consistent with what the user specifies. This solution is meant to mitigate this issue of imbalance of heating/cooling in addition to providing an easy to use platform for which to do so.

The product will eliminate the common issues associated with modern air conditioning systems and also incorporate new features in which consumers may find more convenient. By improving existing technologies and adding The aforementioned

reasoning provides a basis as to the motivation of this project and what we hope to accomplish through our design.

Our overall final goal for this project is to develop a proof of concept of what would be a functioning automated IoT heating and cooling controller via the use. By showing that this IoT design can be implemented in a small scale it can provide a basis on how to apply it to a modern home environment. The main emphasis is to show that this IoT design can be accomplished and easily scaled into a full home using minimally more effort and parts in order to achieve this.

2.2.2 Constraints & Restrictions

As with any design our implementation of the smart air flow system will require certain types of constraints in order to insure that the design is consistent with what was planned. These constraints may vary anywhere from power constraints, time constraints, scalability, size restrictions and economic restrictions associated with designing testing and validating the design. Like any successful implementation of any design there must specific restrictions and constraints set in place in order to fully realize the design which is being planned.

In terms of economic restraints, this stems mainly from the goal of creating a cheap and accessible product which doesn't require a hefty sum of capital in order to enjoy. This constraint is set to roughly \$400. This is a strict constraint and will not be violated as to the budget agreed upon by all members. This budget will exceed the price to manufacture a copy of the functioning IoT system as it accounts for experimentation and testing which requires extra costs.

There will also be a restriction placed on the equivalent market price of this product as one of the goals is for this product to be relatively cheap as compared to the current alternative products provided by companies such as Keen and Flair. The price associated with this restriction will be set to approximately \$120 for the entire working product for two rooms in a modern home. This is without the need of auxiliary products such as the "Puck" and "Bridge" provided by the competitive products.

In terms of time constraints, there will be a semester dedicated to building, assembling testing, and modifying the existing design in order to provide a working prototype. The planning and research phase is what will be completed by the end of the first semester allocated to the Senior Capstone Design project.

This report will be completed near the end of this allotted time when most of these tasks are completed. By this time, there should be a completed plan of design and all parts should have been selected and compared. In the second semester associated with the Senior Capstone Design project the actual testing and building phase will occur. This Of course, these time constraints are required to be strictly followed as to meet the deadlines which have been set for each project milestone.

The power constraints associated with this IoT implementation must also be strictly followed as to keep the system running for extended periods of time. That is, in order for this product to be easily scaled to full scale implementation the power draw must be low enough for the system to continue running without need for often recharging. It is for this reason that power consumption is a very important consideration when making selection in terms of parts and boards.

In order to keep power consumption low this must be complemented by choosing parts which consume very little power. Our total power draw from this IoT implementation should not exceed 10 watts. Regarding scalability, this implementation must be easily scalable to a full scale modern home in order to consider our project a success.

Another constraint/restriction associated with this project is that of ease of use. This design should take no longer than approximately 30 minutes in order to fully understand all of the features and capabilities of the product. This product is meant to be easily picked up and used by the majority of individuals to make this product as appealing as possible to all demographics.

The ease of use constraint is set due to one of the main goals associated with this design. One of the main benefits associated with this design is an overall more convenient product relative to the current alternative. It is for this reason that ease of use is one of the utmost priorities with this project. As previously stated the strict constraint of 30 minutes to fully understand all of the features and capabilities of the design will be strictly enforced and tested with individuals who have no prior knowledge of the IoT system's interface. This will be conducted during the testing and modifying phase of the design when all of the assembling and building of the model has been completed.

The final constraint is that of scalability. Scalability is a very important constraint considering the small scale nature of the design. This implementation is to be designed such that when applied to a full scale modern home it will scale correctly with little effort or modification. This is to be accomplished via using many wireless systems and long lasting power supply solutions. This is also why the power draw of the system must be little enough to continue running for a long amount of time without the need to recharge or replace batteries. In this way consumers will not be required to route cables through their homes or frequently recharge or replace batteries in order to keep a functional system in place.

2.3 Requirements Specifications

“The System” refers to the central hub and all peripheral devices.

1. General System Requirements

- 1.1 The System shall be able to independently control the temperature of each room in a typical home.
- 1.2 The System shall allow a user to update temperature targets from a web application or mobile app.
- 1.3 The System shall be able to be integrated into an existing HVAC system
- 1.4 The System shall be easily scalable.
- 1.5 The System shall require maintenance no more than once a month.
- 1.6 The components of The System shall easily communicate wirelessly with each other
- 1.7 The System shall not cost more than \$120 for a small home.

2. Central Hub Requirements

- 2.1 The central hub shall interface with a home HVAC system.
- 2.2 The central hub shall send commands to the vent dampeners.
- 2.3 The central hub shall track the reported temperatures from each room and determine which vent to open and close.
- 2.4 The central hub shall connect to the internet to receive commands.
- 2.5 The central hub shall be compatible with IoT technology providers (eg AWS)
- 2.6 The central hub shall receive power that the HVAC wiring provides
- 2.7 The total cost of the central hub shall not exceed \$40

3. Distributed Thermostat Requirements

- 3.1 The thermostats shall send temperature readings to the central hub periodically
- 3.2 The thermostat shall sense temperature readings to within $\pm 1^{\circ}\text{C}$
- 3.3 The total cost of the thermostat shall not exceed \$20

4. Distributed Vent Dampers Requirements

- 4.1 The vent damper shall fit inside of existing vents.
- 4.2 The vent damper shall be able to receive open and close commands from the central hub and react accordingly.
- 4.3 The total cost of the vent damper thermostat shall not exceed \$20

5. Software Requirements

- 5.1 The mobile application shall communicate with the central hub via IoT to receive temperature details frequently.
- 5.2 The mobile application shall communicate with the central hub via IoT to send temperature change requests.
- 5.3 The mobile application shall allow the user to control the temperature in each room which contains a vent.
- 5.4 The mobile application shall provide a way to register the central hub device with a user.

2.4 Design Options

Given the above requirements of the system there are several designs that we considered for the hardware side. These options must all be able to communicate with any IoT service that we may choose. The primary deciding factors for our solution is cost and ease of development. The primary motivator for the product is to provide equal or greater

functionality in our product while undercutting cost. These design options will aid in the part selection and design process.

2.4.1 Atmel MCU with wireless peripherals

The first platform we decided on were Atmel chips. Primarily because of the Arduino development platform. The primary benefit of this hardware environment is that it is one of the cheaper options and is very accessible. Arduino has become the primary choice especially for hobbyists because of the community support. For our application what is most important is that the vast majority of peripherals already have open source drivers and libraries written for them.

The primary drawback with choosing to develop our product with an Atmel CPU in the Arduino development environment is that the chips tend to be much simpler. This means that we cannot program the system at a very low level. Additionally any sort of device in the system that has a display will need to store graphics. This means we need to choose processors that have a large enough ROM to support this graphical data. Finally these chips do not have any kind of integrated Wi-Fi or radio controllers.

2.4.2 Thunderboard development platform by Silicon lab

The thunderboard development platform utilizes the gecko family of processors from silicon labs. These processors are more fully featured than the Atmel chips. These gecko chips have integrated Wi-Fi and other wireless protocols embedded inside of the chip. This can allow us to more tightly integrate our solution and reduce the part count of our device.

The largest issue of this platform is the ease and accessibility of development. Their development boards are 100-300\$ per piece which is outside of our development budget. The other issue is that because this platform is not as popular there are less open source drivers written for these boards. This means that it will be more difficult for us to program and develop for this platform as we may need to write drivers from scratch.

2.4.3 Launchxl-CC26X2R1 by Texas Instrument

Another Option is to utilize the LaunchPad development environment. Texas Instruments have microprocessor options all the way up and own the stack. However we are more interested in the more fully featured Micro processors. Much like the gecko family of processors Texas instruments offers microprocessors that are feature rich and offer built in Wi-Fi radios. One of the primary drawbacks is that the Texas Instruments microprocessors are also more expensive options.

2.4.4 NodeMCU by Espressif Systems

The final option is to utilize the NodeMCU development environment. NodeMCU is primarily based on the ESP32 series of micro processors. The primary use case for NodeMCU is internet connected IoT systems as all boards have integrated WiFi

controllers and antennas. The NodeMCU project is similar to the arduino project in that it is very popular in the hobbyist field. It is so similar to arduino that the arduino IDE is often used to program these chips. However, not all libraries developed for Arduino are compatible with NodeMCU.

2.5 Block Diagrams and Illustrations

In order to best understand the organization of the system a visual representation is best. In this manner it is easier to make decisions on the system as well as to make it easier to explain the structure of the project.

2.5.1 Hardware Software Integration

At a high level the flow of control follows as such. The user is able to utilize the mobile application which will communicate with the IoT cloud services. The central hub will be in charge of interconnecting between the auxiliary boards which include temperature sensors and also control the servos for opening and closing valves.

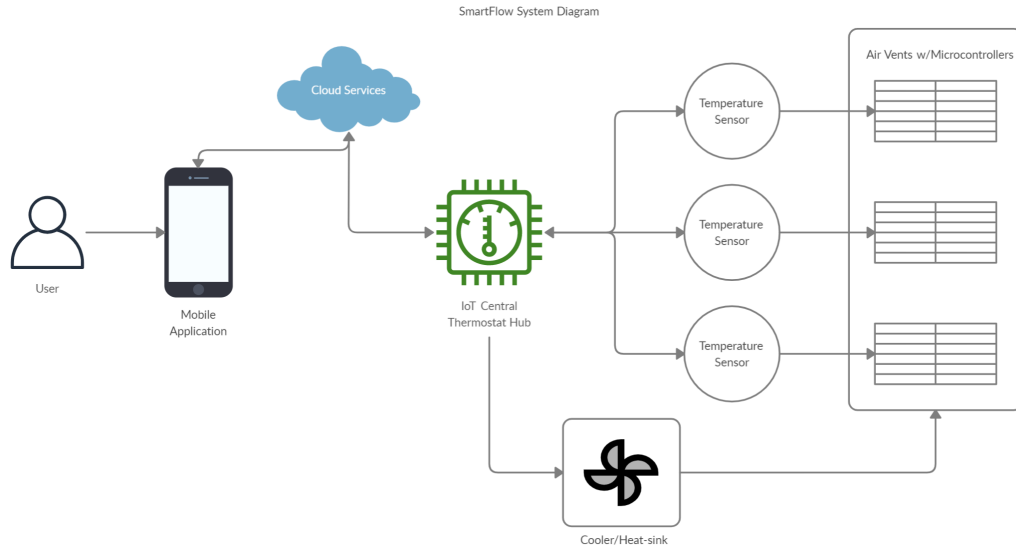


Figure 1: Hardware and Software Control Flow

2.5.2 Hardware Communication

In our system there are three types of devices - the central hub and the two types of peripheral devices. Given the temperature targets received from the web server over WiFi the central hub will keep track of temperature recordings from each room and send control signals to the AC system and the Vents so that the air can be directed towards where it's needed most.

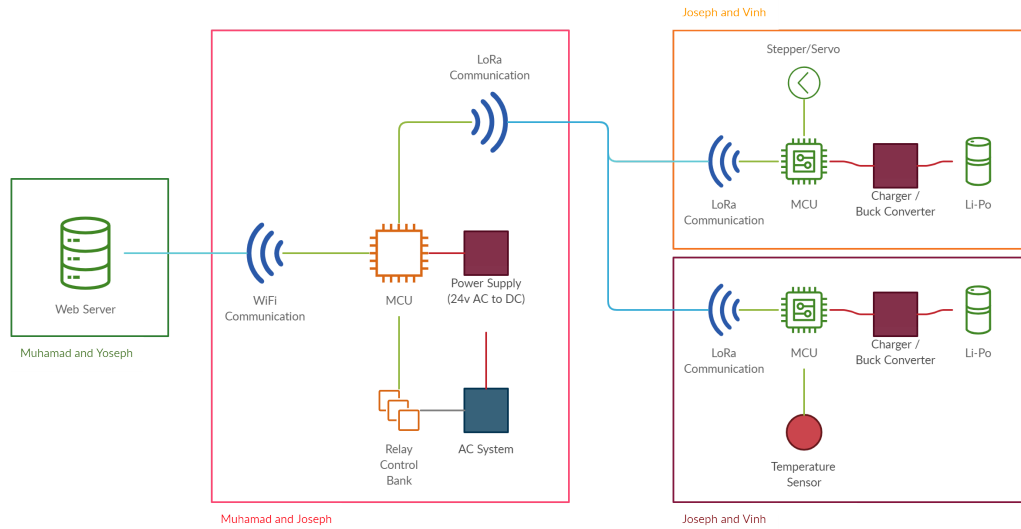


Figure 2: Hardware Block Diagram

2.5.3 Application Control Flow

We want to design our application with functionality in mind first. The primary purpose of the application is to control the AC system in a home via the user's smartphone. As such the simplest design possible for our application is best in order to provide the best user experience.

Application Flow

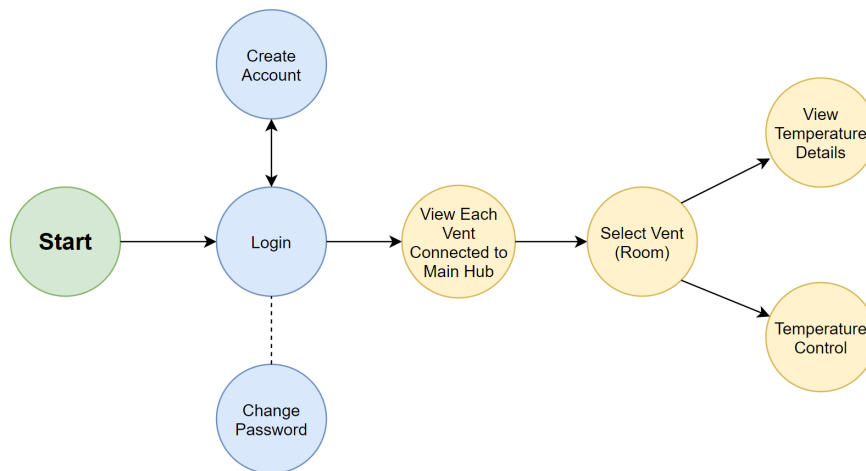


Figure 3: Application Control State Machine

2.5.4 Role Breakdown

It is important to assign roles when undergoing a large development project. It allows for efficient and organized assignments of jobs. It also allows for individuals to specialize on certain portions of the project in order for tasks to be completed more efficiently and effectively.

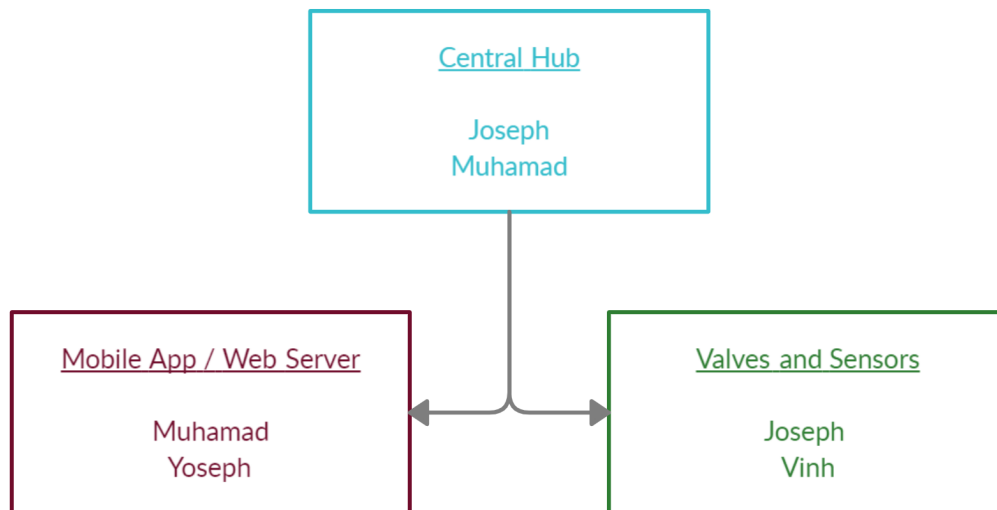


Figure 4: Group Role Breakdown

2.5.5 Marketing and Engineering Requirements

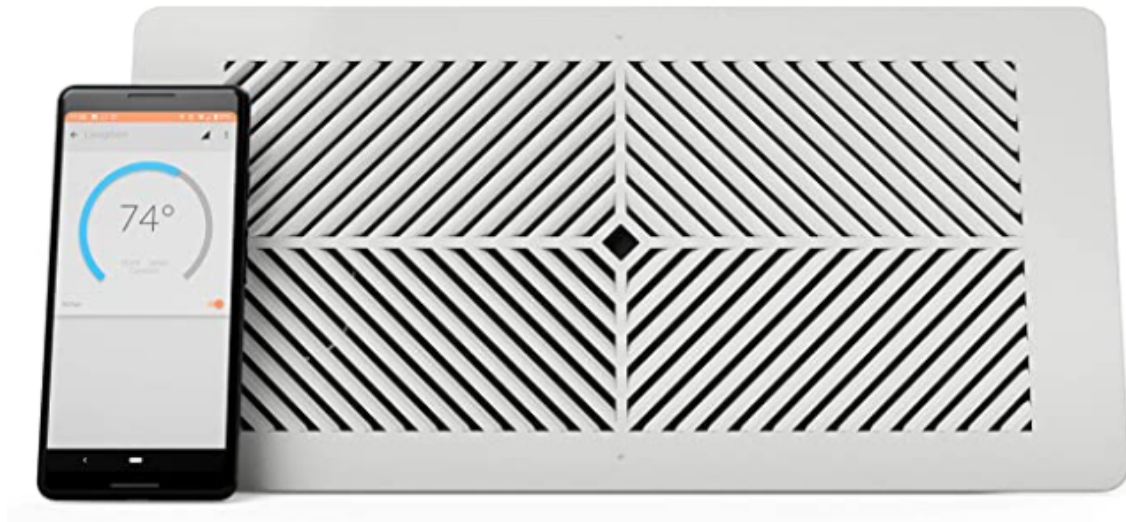
The below diagram visualizes the most important engineering and marketing requirements and their relationships inside of a house of quality. Visualizing this data is important in order to make decisions on part selection as it allows us to better decide on

3. Research and Part Selection

In this section we will discuss the research revolving around the idea of a fully automated air conditioning system that allows you to maintain different temperatures in each room or zone of the building from the convenience of your mobile device. The research that will be detailed below involves a range of topics including existing relevant designs and products in the market, an overview of relevant underlying technologies, technological specification comparisons, hardware selections, and etc. Although the idea of separately controlled vents that allow you to change temperatures in different rooms does currently exist in the market, this section includes useful additional features we plan on implementing that those designs do not have. This requires heavy research to be made on the technologies and techniques that make up an automated system of this sort, like the Internet of Things, networking standards, etc. There are many notable sources to get information on these technologies from like some of the top tech companies in Silicon Valley, higher education institutions, research articles, and any other supporting materials. Having reliable references on the technologies and techniques we plan to implement will make it easier to plan and design our project with the best standards taken into account. This section includes an introduction to each area of design that was considered for our project, which will assist in laying out the options we have and narrowing them down based on our objectives.

3.1 Existing Designs and Products

The products developed by Keen and Flair are aimed at the higher end market and don't provide an accessible solution to the problem many are facing as it pertains to improper heating/cooling. Keen in particular results in an extremely high price of entry due to the amount of products which must be purchased in order to create a functioning system. For instance, in order to accomplish the proposed model a smart vent and smart puck must be placed in each and every room which requires smart air flow capabilities. With each vent costing up to just under \$90 and each puck costing \$119 it becomes very inaccessible to the average user as this simply is far too expensive. In addition to the high cost associated with these products, the proprietary nature of them forces end users to only buy these products further exacerbating the issue. Figures 1 and 2 show the main products required in order to achieve the smart vent system of Flair. These issues will be further emphasized in the later sections.



*Figure 6: Existing Smart Vent Technology: Flair Smart Vent
(Reprinted with permission from Flair)*

Figure 1 shows the standard Flair Smart vent. This design is realized such that the vents open and close in order to maintain air temperatures in rooms. In order to constantly maintain this process a “Puck” must be placed in every room in order to constantly measure the temperature. However, if one room requires heating while the other requires cooling one of these processes will have to take priority.

This issue unfortunately is associated with the design of Air Conditioning units and can’t be resolved with IoT distributed systems. Simply put, installing smart vents and “Pucks” in each and every room will not fully resolve the issue unless rooms have small temperature differences between them.

The main issue with these vents is the price as these devices are not cheap and must be placed in every room in order to achieve the desired effect. In addition to this as stated before each room must also have a Flair “Puck” placed in them. This effectively is a smart thermostat in every room which is simply an unnecessary example of over-engineering. Multiple pucks can result in a very expensive system to assemble and just isn’t a realistic option for the average consumer. Rooms with multiple vents suffer the most from the overarching issue of cost as each vent costs up to just under 90 USD not factoring the price of the “Puck” as well.



*Figure 7: Flair Puck
(Reprinted with permission from Flair)*

Figure 2 shows the Flair “Puck” which acts as a smart thermostat to be placed in each and every room requiring smart air flow technology. The “Puck” device Flair sells is aimed to help the smart air flow system they design work in each and every room which requires a smart vent.

The overarching issues associated with the “Puck” is that they must be placed in each and every room containing a smart vent and they act as smart thermostats. The fact that these devices must be installed in every room coupled with the nature of these devices as being smart thermostats directly interferes with two of the goals of our project.

The stipulation in which “Pucks” must be placed in every room requiring smart vents is what really makes this a premium product which may be unattainable to the average person looking to experience this technology. With each Flair “Puck” coming in at about just under \$120 this can easily become a system which exceeds our price goals and the budgets of many individuals.

We are looking to eliminate the need for central hubs such as thermostats which require users to physically locate and operate them in order to achieve desired temperatures. In addition to this we intend for this system to be attainable by more average users and want to avoid designing a product that is aimed towards the higher end market. The need to purchase many “Pucks” in order to service several rooms can result in a quickly expensive project depending on the amount of “Pucks” needed around the home.

Also, due to the proprietary nature of these devices users are simply forced to purchase these devices and technologies from Flair which results in very expensive products.

Without having alternatives this creates a large monetary barrier of entry for individuals who want to gain access to this technology.



*Figure 8: Keen Home Smart Bridge
(Reprinted with permission from Keen)*

Figure 3 showcases the Keen “Home smart Bridge” which is an additional device required in order to properly configure the Keen smart vent system. This device is required in order to connect the smart vents to the cloud and furthermore to other partner devices. As opposed to Flair’s competing “Puck” this product isn’t very expensive at only \$49.99 versus the nearly \$120 of the Flair “Puck”. However, each smart vent depending on the dimensions may cost anywhere from \$111.99 to \$133.81.

This configuration by Keen is extremely expensive as multiple smart vents could cost individuals a hefty sum of capital. The price per vent is significantly more expensive than that of Flair. This company retains the same issues as Flair in terms of being inaccessible to the average consumer due to the high cost associated with these products.

Even with requiring extra devices in order to have a functioning system these products are still poorly reviewed on popular websites such as Amazon.com. The average review score for the Keen smart vents is approximately 2.5 stars out of 5 possible. The average review score for the Keen “Home Smart Bridge” is approximately 2.7 stars out of a possible 5 stars. In addition to the expensive prices of these products, it is very clear that consumers are obviously not impressed with the offerings of Keen’s solutions to temperature imbalance in modern homes.

Some of the complaints resulting in these negative reviews include “The vent keeps flashing green “ready to pair” but stays undetected “Using iPhone 8 with the latest iOS”. I also noticed that the bridge loses connection even though it’s hard wired to a switch or to the router!” and “Total waste of money. Vents do not stay connected at all”. These issues seem to stem from poor communication between devices and a lack of consumer ease of use.

Our product is aimed to be an easy technology to learn and use. All of the basic functionality and features associated with our product should be easily learned in a relatively short amount of time without extensive learning from the consumer. In addition to this, our system will not require that consumers buy several components in order for the system to work seamlessly with wifi compatible devices. These pre-existing products all require the use of extra devices which allow for the system to be connected to the internet or cloud which handles the temperature data.

3.2 Relevant Technologies

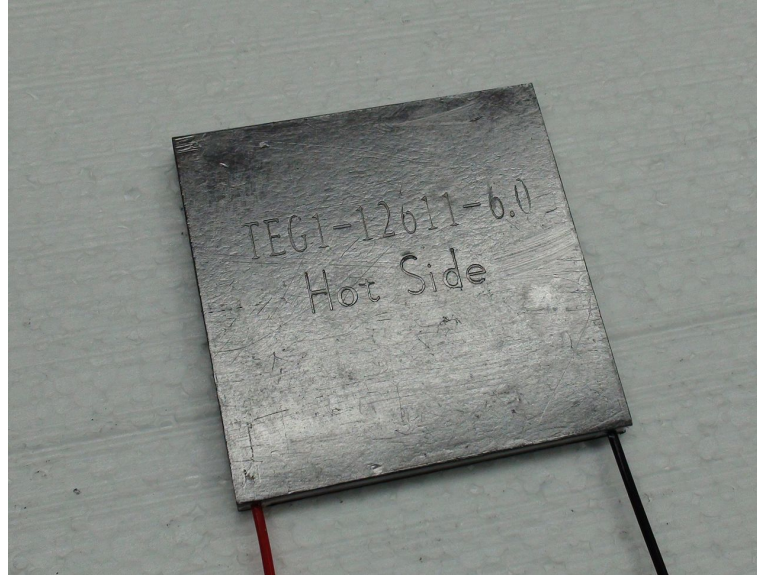
This section will cover many of the technologies which are relevant to the planned design and how they will be potentially incorporated into the final product. This includes but is not limited to electrical/electronic components, software technologies, hardware modules, communication techniques, specialized circuits/circuit designs, programming languages and many other devices which may hold relevance to the planned design.

These products may find themselves useful for more hardware or software purposes depending on how they will be implemented. In addition, this section will cover why certain devices were chosen and how they may be useful to the final product. This section will also explain the inner workings of these technologies as to understand the fundamental concepts which enable these systems to function.

These devices and technologies may be already found in other IoT systems or designs. In addition to this, we may not use all of the technologies listed in this section in the final implementation of the project. Rather, we will evaluate which will provide the most benefit for our use cases and incorporate those technologies into our final design. In addition to simply covering the types of technologies and devices which may be relevant to our final implementation, this section also aims to delve into certain paradigms and concepts associated with the respective technologies in order to provide a basis as to their relevancy. Each section will cover a technology which may be useful in the final implementation of the model however may not necessarily be used in the final project design.

3.2.1 Peltier Effect & Thermoelectric Cooling

The first technology which is relevant to our design and will be integral to the main use of the product is the thermoelectric cooler. This device is the main component associated with heating and cooling the rooms with the microcontrollers. This component will be the driving force behind heating and cooling the system. The reasoning behind this device's usefulness is the ease at which it can change modes from heating to cooling. By simply switching the polarity either side of the device can become hotter or cooler while running a current through the device.



*Figure 9: Thermoelectric Module
(Creative Commons: Gerardtv)*

https://commons.wikimedia.org/wiki/File:Thermoelectric_Seebeck_power_module.jpg

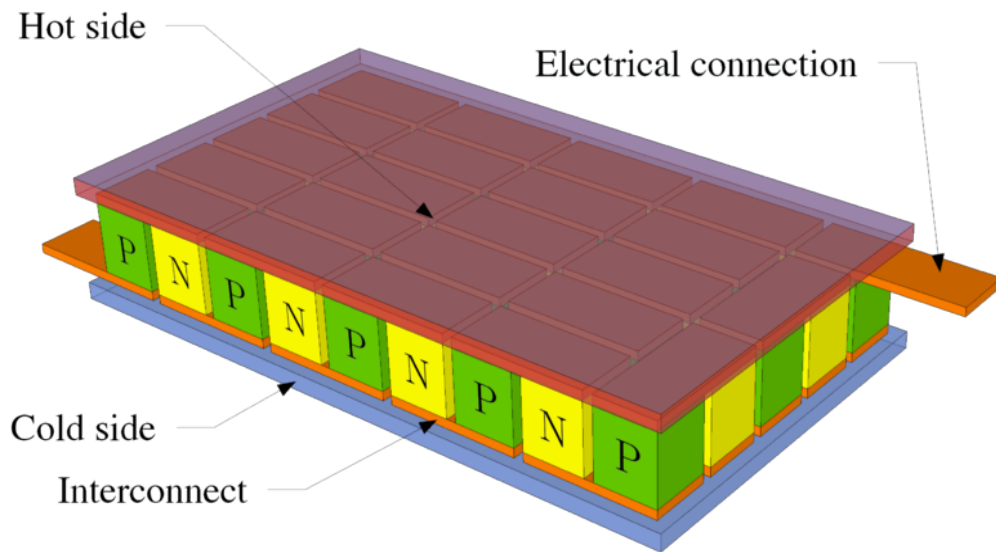
Figure 4 showcases a thermoelectric module or often referred to as a thermoelectric cooler (TEC) and Peltier coolers. These devices can be used for either heating or cooling but are most often used for cooling applications using some type of controller. These devices use a phenomenon known as the Peltier effect. The Peltier effect is used to create a heat flux at the link of two separate materials. These devices are then able to transfer heat from one side to the other via the expenditure of electrical energy. It is for this reason that this device will be ideal for the application associated with our design in which it is modeled after the heating and cooling in a home environment.

It is simply unrealistic to test our design in a home environment and we plan on placing more emphasis on the IoT design rather than the air conditioning system itself. That is, it isn't a goal of this project to perfectly emulate the actions associated with modern air conditioning systems. The heating and cooling capabilities of these thermoelectric modules are enough for our application and will showcase the drastic change in temperatures which can be exhibited in a small scale system.

The intention is to use a similar device in the small-scale design in order to provide adequate heating and cooling to the rooms. This device will be used in conjunction with a fan in order to emulate the function of an air conditioning system's blower. This will allow the system to provide heating or cooling via blowing one of the sides in which temperature increase or decrease is needed.

In order to achieve the heating/cooling effect in the rooms the polarity of the module must be inverted. That is, if originally in heating mode switch the polarity in order to result in the other side of the module to become cold while the other becomes warm. The fan will then blow through the side which is associated with the correct temperature

associated with the room. Ideally this will be accomplished using an H-bridge in order to easily allow for inversion of the polarity.



*Figure 10: Thermoelectric illustration
(Creative Commons: michbich)*

<https://commons.wikimedia.org/wiki/File:Peltierelement.png>

Figure 5 further showcases the inner workings of the thermoelectric module and further illustrates its usefulness to our design. As it is shown in the figure, these devices are made up of many couples of n-type and p-type semiconductors. This configuration allows for the transfer of heat from one side to the other which results in extreme temperature differences on both sides.

The heat being absorbed from the side allows for that respective side to become very cold. However in the case of the side which is releasing heat, it results in a very hot surface as the heat from the other side is being transferred to it via the Peltier effect.

Essentially, this figure provides an inside view to the overall structure of a typical thermoelectric cooler. As it can be seen the electrical connection annotated in the image shows where the module would be connected to some power supply. In the figure the electrical connection is more specifically connected to a p-type semiconductor as all of the p-type semiconductors are green.

The thermoelectric cooling device will be connected via positive and negative terminals in the electrical connections. The two sides of the thermoelectric cooler are connected by many groupings of n-type and p-type semiconductors and this system is further illustrated in Figure 5.1. This figure which is shown below displays an instance in which there is one n-type semiconductor and one p-type semiconductor. The figure also shows the flow of electrons and provides a schematic type of view to how the system would function with a current being passed through. The system displayed in Figure 5.1 represents a

subsection of the overall system which powers the entire thermoelectric cooling device. Putting many of these systems in a structure similar to the one found in the aforementioned figure results in a similar design to Figure 5.

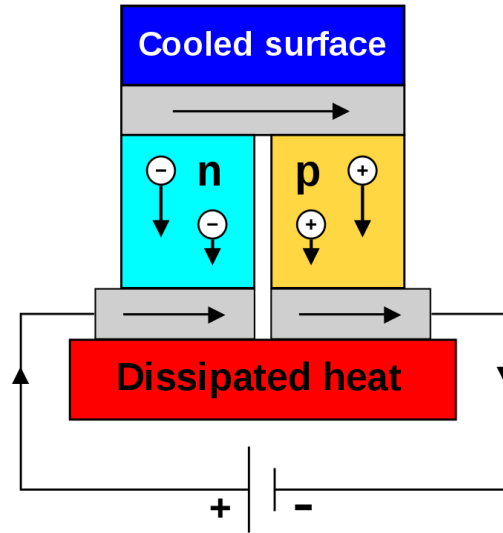


Figure 11: Schematic representation
(Creative Commons: Ken Brazier)

https://commons.wikimedia.org/wiki/File:Thermoelectric_Cooler_Diagram.svg

Figure 5.1 goes into detail on each of the n-type and p-type semiconductor groupings and shows the flow of electrons through this system. N-type semiconductors, as shown in the figure, are semiconductors which are made via doping the material with electron donor atoms. In n-type semiconductors the majority of their charger carriers consist of negative electrons. P-type semiconductors are semiconductors which are made via doping the material with electron acceptor atoms. In these types of semiconductors most of the charger carriers associated with them are positive holes which can be occupied by other subatomic particles.

The system displayed in Figure 5.1 shows the flow of electrons through the direction of current. This Peltier effect is what occurs when an electrical current flows through a thermocouple circuit. This results in the heat being released on one side of the junction and being absorbed at the other end of the junction. The Peltier effect was named after French physicist, Jean Charles Athanase Peltier, who made its discovery in 1834.

The Equation associated with Peltier heat is shown below:

$$Q = (\Pi_A - \Pi_B) I$$

such that A and B are conductors, Π_A and Π_B are the Peltier coefficients of their respective conductors, and I is the electrical current from conductors A to B . These Peltier coefficients are measurements in terms of the amount of heat which is carried per unit charge.

In addition to the Peltier effect, another phenomenon which can be observed with thermoelectric cooling/heating is the Seebeck Effect. This effect is what occurs when there exists an accumulation of electric potential throughout a temperature gradient. Thermocouples produce voltages which depend on certain temperatures due to the Peltier effect.

The voltage produced by the thermocouple can then be used to measure temperature as it is proportional to the temperature measurement. Due to this, thermocouples are very often used in electronic temperature sensing applications. Therefore, this is extremely relevant to the design of the smart air flow IoT system.

Arranging multiple thermocouples in series results in an electronic device known as a thermopile, which exhibits the Seebeck Effect. Also, the Seebeck effect results in an electromotive force and also induces significant changes in voltage or currents. Thermopiles are used to convert thermal energy into electrical energy which also overlaps with the Peltier effect discussed.

3.2.2 Voltage Regulators

Voltage regulators are circuits that's designed to regulate a voltage at a desired level. If the power source, such as a battery, varies from time to time, or have an inappropriate voltage level for our application, voltage regulators are required to provide a constant voltage for the device to operate. Because we're aiming to run our device on a battery, it should be highly efficient. There's two common circuits available, the linear voltage regulator, and the switching voltage regulator.

3.2.2.1 Linear voltage regulator

Linear voltage regulators utilized an operational amplifier with a NPN transistor. In a nutshell, the operational amplifier will control the NPN transistor to keep the output voltage at a desired voltage level. The desired voltage level can be programmed with R1 and R2: $V_{out} = V_{ref} * ((R1+R2)/R2)$; in the following figure, V_{ref} is $V_{cc} * 0.099$. The output voltage will always be below the input voltage. Additionally the desired output voltage is only achieved when the input is greater than the desired voltage, this difference is known as dropout voltage. There are regulators that have low dropout voltage, also known as LDO.

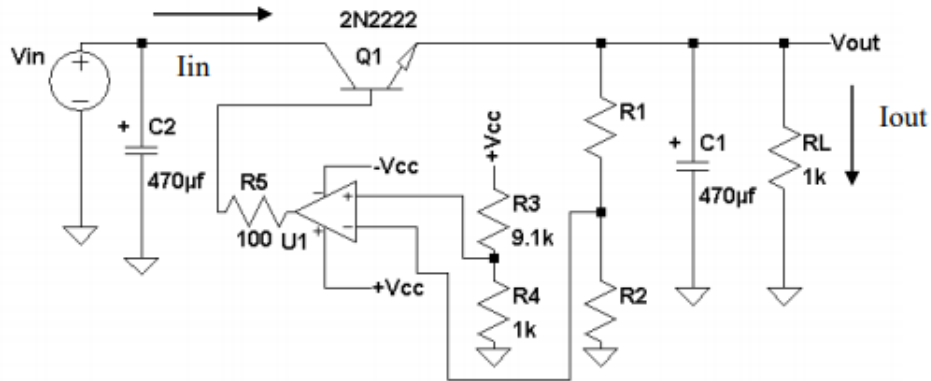


Figure 12: A functional schematic of an LDO regulator

The main disadvantage with linear voltage regulators is that their efficiency highly varies in applications. This is because the efficiency of the regulator heavily depends on the input and output voltages. In some applications, low dropout regulators can reach above 90% efficiency if the input voltage is very close to the desired output. On the other hand, if the input voltage is higher than the desired output, its efficiency can get as low as 30%. The smaller the difference between the input voltage and output voltage, the better the efficiency, and vice versa.

Additionally, most of the power is consumed by the transistor and converted into heat. Therefore, heatsinks might be required to dissipate the heat. Because we're using a battery that can change its voltage, it might not be effective to use a linear voltage regulator.

3.2.2.2 Switching voltage regulator

Switching voltage regulators are much more complex as more logic is required to control a gate (which is usually a MOSFET). Switching voltage regulators are usually available as highly integrated ICs. The circuit works by switching the MOSFET in a pre-programmed frequency, effectively charging and discharging the inductor. The desired voltage is programmed by a voltage divider, which will affect the charging period of the inductor. In practical applications, potentiometers are used, as resistors have tolerance and can't be accurate.

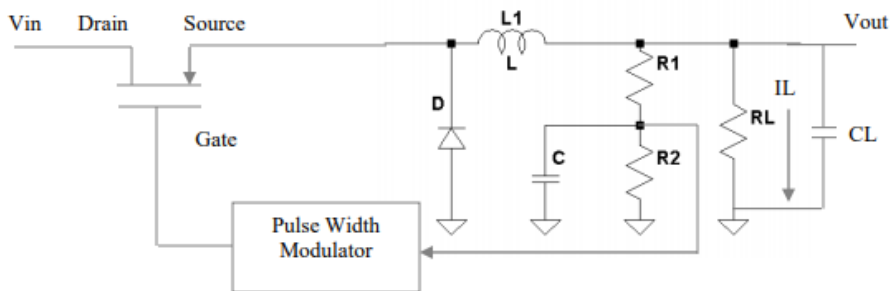


Figure 13: A functional schematic of a Switching Regulator

Switching regulator can be used to both increase (boost converter), and decrease the voltage (buck converter) to the desired. An IC is designed to perform one, or both functionality. The figure above is configured for a buck converter; to get a boost converter, the inductor and diode are replaced with each other.

Switching regulator has 2 modes of operation, continuous and discontinuous. In continuous mode, the current stored in the inductor is enough to last until the end of the discharging phase. In discontinuous mode, the current stored in the inductor is not enough to last until the next charge period. In this case, the IC itself will open the gate and produce a small sine wave, just enough so that the output still has the desired voltage.

Efficiency wise, switching regulators are very effective. A well designed IC can have efficiency as high as 90%. Similar to linear switching regulators, its efficiency can be affected by voltage, but the difference is much negligible. On the other hand, current can heavily affect the efficiency; as the current increases above 1A, its efficiency can be reduced.

The disadvantage of this circuit is that due to the IC continuously switching, there will be more noise presented in the output. Therefore it's important to have a suitable capacitor, as well as ground planes to mitigate the noise. Additionally, as the IC is more integrated and requires more passive elements like inductors, the cost of a switching voltage regulator is higher than that of a linear regulator.

3.2.2.3 Comparison

Linear regulators are cheaper and simpler than switching voltage regulators, but switching regulators can provide more reliable efficiency. Therefore, the choice heavily depends on the application.

Table 1: Linear Regulators vs Switching Regulators

	Linear Regulator	Switching Regulator
Pros	<ul style="list-style-type: none"> ● Simple and cheap ● Doesn't create noise ● Efficiency doesn't decrease on different current (a linear device) 	<ul style="list-style-type: none"> ● Higher efficiency in most cases ● Can create voltage lower and higher than input voltage
Cons	<ul style="list-style-type: none"> ● Efficiency can vary drastically depending on input voltage ● Can only lower the input voltage to desired level 	<ul style="list-style-type: none"> ● More costly, due to IC and inductors ● Create noise ● Takes more footprint on PCB

Since we're using a LiPo battery, the voltage can vary from 4.2V (fully charged) to 3V (critically low). The desired voltage should be suitable for micro controllers, which can be below 3V. Since there's such a big gap between the voltage of a fully charged battery

and operational voltage of the MCU, a linear regulator wouldn't run at high efficiency. Since we're building IoT devices that are required to run for months, efficiency should be the priority. Therefore, switching regulators are better suited for the application, even though it's more expensive than linear regulators.

3.2.3 Actuator

This section will discuss actuators, components that create mechanical movements in a system. These actuators are required for application where physical movements are needed, which is very common in automations. Actuators in digital systems are usually powered with an external power supply. The MCU can then control actuators with low powered logic signals

There are 2 important factors to consider when picking an actuator, its power consumption when fully operational, and its power consumption when idle, which is known as quiescent current. This is because the device is battery powered, so options that require less power are more preferable. We'll look at servo motors and stepper motors, then compare the 2 options.

3.2.3.1 Servo motors

Servo motors are simple to operate actuators. These motors have 3 wires, power, control signal and ground. Servos are most commonly powered by 4.8-5.5V through its power wire. Meanwhile the signal wire is used to receive control signals from MCU, which will make the servo turn its axis and hold at the requested angle. The position of a servo is determined by square waves with different duty cycles. Most servos share the same control signal of 50Hz squarewave, where 5% duty cycle for default position, 10% duty cycle for max position (which is 180 degree relative to the default position). These wave signal can be created with ease by micro controllers

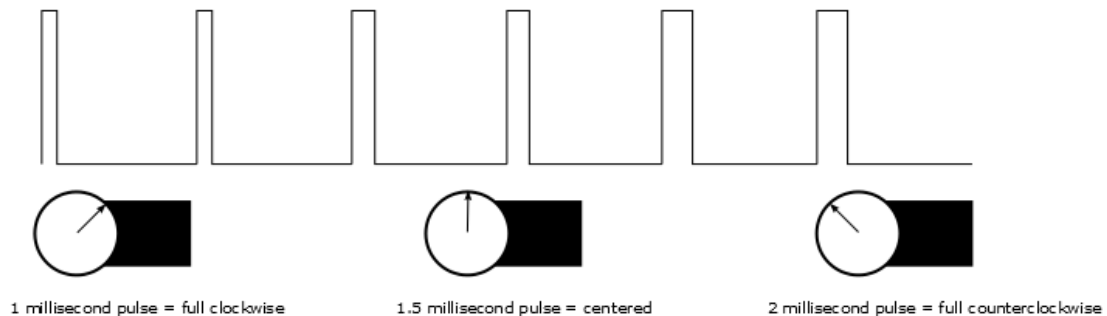


Figure 14: Visualization of a servo's control signal

Servo motor can differentiate the signals because it has a built-in microcontroller. The microcontroller performs 2 tasks, reading the current axis position and reading the requested angle. If there's a difference between the current axis angle and the requested angle, the microcontroller will spin a motor in the required direction until the two angles

match. Since the microcontroller constantly reads the input signal (every 20 ms), it'll react immediately if there's any subtle changes, making the servo extremely reactive and easy to use.

Even though servos can be easily controlled with just 1 signal wire, they have some drawbacks. The downside of servos is its power consumption, both operating and idling. Due to the fact that every 20ms its microcontroller will check the current position, servos can consume around 2.5-5 mA, just by standing still. In operation, it can draw anywhere from 200-600mA, heavily depending on the speed and torque required to spin its axis. If the servo draws more current than what the power supply can provide, the power supply can be damaged. Therefore, the power supply must be suitable for the servo.

3.2.3.2 Stepper motors

Stepper motors are motors that are designed to spin in increments of an angle, which is known as a step. The step is based entirely on a stepper motor design: the thinner the step, the more accurate the stepper motor can rotate. Stepper motors are useful in a lot of applications since they can be controlled to spin in specified amounts. Additionally, stepper motors can be combined with other mechanisms, such as a screw rail, to transfer rotation energy into linear movements, opening the door to all kinds of mechanical application.

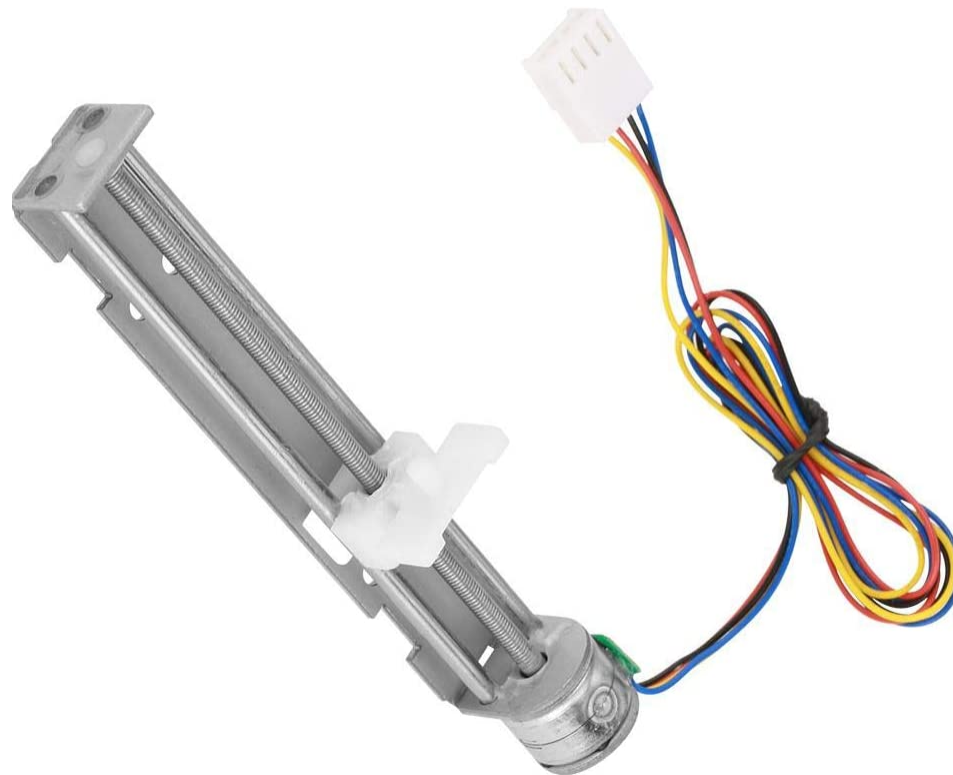


Figure 15: A stepper motor with a screw rail

However, implementing a stepper motor can be challenging. Firstly, a stepper motor requires more power to operate than what a logic from an MCU. Secondly, each of the coils needs to be controlled individually, which can take up precious processing power of a MCU. Therefore, a stepper motor driver is usually used. Drivers are ICs that interpret logic input and control the stepper driver's coils accordingly. Stepper motor drivers allow the MCU to control a stepper motor that's powered by a power supply much larger than its own.

Unfortunately, there's one very common issue with stepper motors, it can skip steps. This happens whether the motor fails to rotate, which can be caused by not sufficient power or not enough torque. When a stepper motor skips its steps, it's impossible for the MCU to know the exact location of the stepper motor. Therefore, more electronic components are required to sense the location of the stepper motor. This can be rotary encoders, optical sensors, or contact switches. The most common approach is using contact switches to detect if a linear rail reaches a certain position, known as calibrating.

In conclusion, stepper motors can provide accurate movement, but it requires a significant amount of hardware to be controlled by a MCU. It requires stepper drivers, which have built-in logic that will control the stepper motor with logic inputs, and sensor, which allows the stepper motors to be calibrated. The advantage of stepper motors is that they allow full customization, from hardware side, to power use, to movement speed. This freedom allows engineers to build mechanisms that's perfectly suited for their application.

3.2.3.3 Comparison

Both servos and stepper motors are used to create precise mechanical movements. Servos are ready to use components, but can have unpredictable power consumption. Meanwhile, stepper motors require more hardware, but they allow full customization for each application.

Table 2: Servo vs Stepper motor

	Servo	Stepper motor
Pros	<ul style="list-style-type: none"> • Easy to operate, only requires 1 IO pin from MCU • Absolute position, will always return to the correct angle • Automatically correct its position 	<ul style="list-style-type: none"> • Provide accurate and precise steps in any direction • Power consumption can be limited by its driver
Cons	<ul style="list-style-type: none"> • Only rotate half a circle • May pull large amount of current in a short time span and can damage its power source • High idle current 	<ul style="list-style-type: none"> • Relative position, need calibration • May skip steps and stray away from the required position. • Require multiple IO pins (at least 3 pins) • Require more hardware

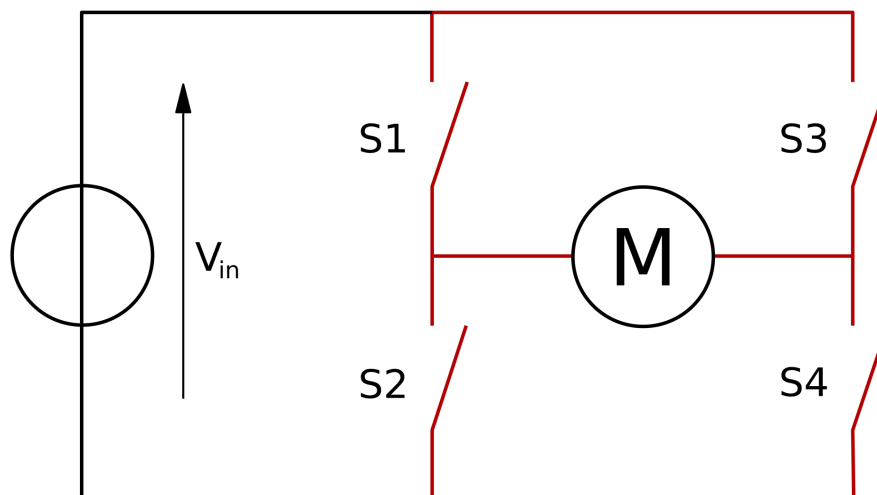
From the comparison, it's clear that servos are much more convenient to implement since they provide accurate position. However, while idle current can be eliminated by enabling or disabling the power source, its operating current can create a lot of power issues. Firstly, such a high power drain is not preferable in battery powered circuits. Secondly, it can draw more current than the source can provide, which will damage the source.

These disadvantages can be fixed if we can implement a stepper motor system. Theoretically, a customized stepper motor system can have controlled power consumption, so it can be modified to fit our IoT application. However, due to limited times, we can't test out this option. Therefore, we decided to use servo.

3.2.4 H-Bridge

This section will discuss the H-bridge design and its relevance to the implementation of the project. It will also cover how the H-bridge works and the different states in which the circuit operates in. The circuit's design in terms of connections, nodes, and input voltage will also be covered in this section.

This material is highly relevant to the Electrical and Computer Engineering curriculum and will be showcased in our final design. The main point of emphasis with this specific circuit is its ability to switch the polarity of voltages applied to a load. This will be integral when it comes to the switching of heating or cooling in the peltier coolers we plan on implementing. This is due to the thermoelectric coolers requiring a change in polarity in order to change from hot to cold on each side.



*Figure 16: H-Bridge Diagram
(Creative Commons: Cyril BUTTAY)*

https://en.wikipedia.org/wiki/H-bridge#/media/File:H_bridge.svg

In Figure 5.2 it illustrates the basic circuit design of an H-bridge. The fundamental concept behind an H-bridge is its ability to switch the polarity of the voltage being applied to a specific load. H-bridge circuits are typically used in robotics and are most often applied to DC motors in order to let them run in both directions. These circuits are also widely available as integrated circuits even though they can be built from scratch using different components.

The basic design of an H-bridge consists of four mechanical or electrical switches connected to a voltage source. The “M” indicated in the figure is representing a motor that would be connected to this circuit in between the switches. Depending on how the switches are configured and which nodes are completing the current flow the direction of the motor’s rotation will change.

This phenomenon is due to the voltage reversing and therefore resulting in an inverted polarity. This, in turn allows for the circuit to accomplish different tasks based on the polarity required by the specific application. If the device requires a positive polarity in order to accomplish a task this can be toggled using the switches via connecting them in certain configurations.

In the case of our implementation, instead of a motor the positive and negative nodes of the peltier thermoelectric cooling device will be placed into the H-bridge circuit. This will allow for easy polarity changes when the system requires heating or cooling to specific rooms.

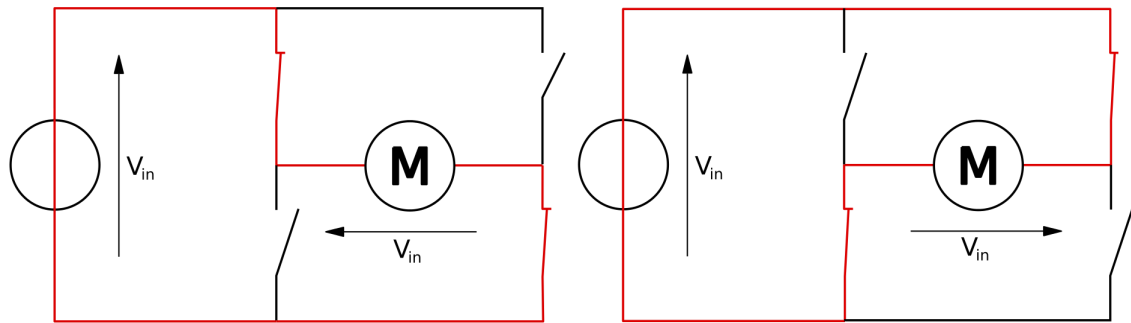
As it can be seen in the figure there are four switches. Those being, switches S1, S2, S3, and S4. Only two switches can be switched at once and the switches must be on different sides of the motor or device connected.

If all four switches are activated or two switches on the same side are activated this will clearly result in a short circuit to the motor or component being connected to the switch nodes. If switches S1 and S2 or S3 and S4 for both are activated, then either the positive node or the negative node of the component will be shorted. This is due to both of the S1 and S2 or S3 and S4 nodes merging into one node.

If all four switches are activated, then both sides of the motor will be connected to the same nodes resulting in a short to the entire device. If only one side is connected and the other isn’t such as S1 and S2 or S3 and S4 then one side of the motor will remain in an open circuit state and current will be unable to flow through the connectionless gap. It is for this reason that the switches must be activated on opposite sides of the device connected.

Also the switches that only are activated simultaneously are S1 and S4 or S2 and S3 because if both switches are connected are connected to the same node from one end it

will also result in a short circuit. For example, if S1 and S3 are connected, then both ends of the motor will be connected to the same node which results in a short circuit to the device. In addition to this the other of the voltage source will be open and the circuit will not be complete. The same issue is faced when the switches S2 and S4 are activated as the opposite end of the motor will be shorted once again.



*Figure 17: The two basic states of an H bridge
(Creative Commons: Cyril BUTTAY)*

https://en.wikipedia.org/wiki/H-bridge#/media/File:H_bridge_operating.svg

In this figure, Figure 5.3, the main point of emphasis is on the different states of the H-bridge circuit and what that looks like in terms of the circuit design. In this case it shows the nodes that are connected when the switches are toggle back and forth between different states.

In the first configuration, the nodes S1 and S4 are connected to either end of the motor or in our case what would be the peltier thermoelectric cooling device. As shown in the diagram the flow of current is towards the left.

In the second configuration the nodes S2 and S3 are connected to the overall circuit at either end of the motor. This completes the circuit and allows for current to flow through the component connected at the respective nodes. However the flow of current is inverted and the voltage is also inverted.

For instance, if the voltage difference of the two nodes of the motor was approximately 5V before switching the configuration of the switches after performing the switch the voltage would subsequently become -5V. Therefore this circuit accomplishes its intended task and inverts the polarity of the nodes connected to the device.

As previously stated the h-bridge will be mainly used in order to quickly and more easily reverse the polarity of the peltier thermoelectric cooling device. As to not invert wire traces which may cause complications and using the H-bridge will simply provide an easier convenience as this operation may occur frequently.

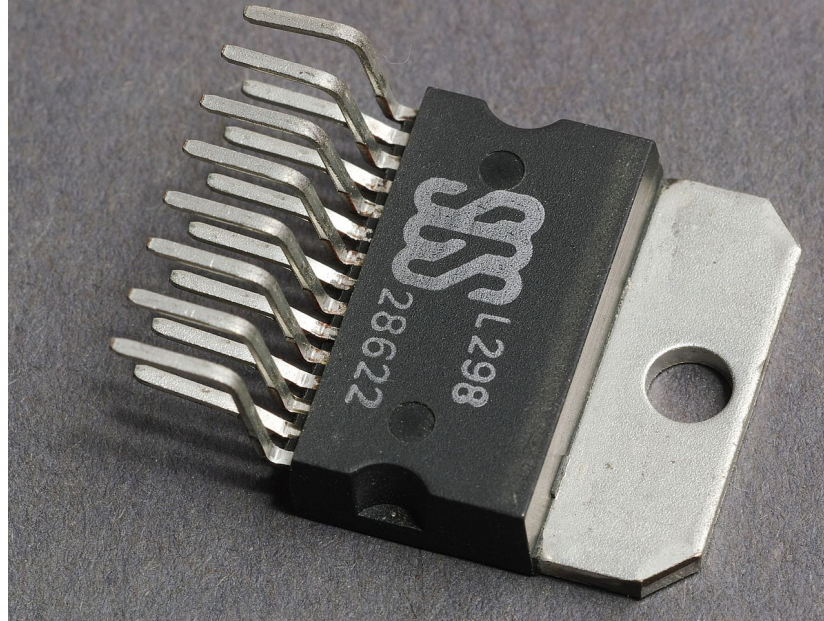


Figure 18: L298 H Bridge IC

https://commons.wikimedia.org/wiki/File:L298_IMGP4533_wp.jpg

The image found in Figure 5.4 is similar to the H-bridge which has been planned to be incorporated into the final implementation of the IoT design. This device will most likely be attached to a PCB which in turn is connected to the peltier thermoelectric cooling devices and by inverting the polarity will allow for either cooling or heating to be produced from the thermoelectric coolers.

The connection between the H-bridge and the thermoelectric cooling devices are the main driving force behind the planned implementation of how the small-scale model will be approached in terms of simulating a modern air conditioning system. As it can be seen in the image, the H-bridge depicted is an auxiliary device which can be “snapped” onto a board and then when signals are sent, switch the polarity of the device which is connected to the overall circuit.

3.2.5 Internet of Things (IoT)

The Internet of Things, or IoT, describes the network of billions of physical devices that are connected to the internet, all collecting and sharing data. These devices are known as “things” and are embedded with sensors, software, and other technologies. They also are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. Thanks to extremely cheap computer chips and the ubiquity of wireless networks, it’s possible to turn anything, from something as small as a pen to something as big as a cruise ship, into a part of the IoT.



Figure 19: Internet of Things (Creative Commons: JUST Creative)

The smart devices that are intermittently connected through IoT share the data that they collected to an IoT gateway or other edge device where the data is then sent to the cloud to be analyzed and managed. In our case, we only require one smart device to be connected through IoT, the central hub thermostat. However, in some cases these devices communicate and share information with other devices that are related and perform actions based on the data they receive from one another. Being within an IoT network allows the devices to do the heavy lifting without any human action required. However, people can interact with the devices. In our project for instance, the user will set up the central hub thermostat and control the temperatures in each room using it.



Figure 20: Major components of IoT (Creative Commons: rfpape)

The IoT network is important because it helps the consumer live and work smarter, giving them complete control over their lives. In addition to providing the capability to

automate homes with smart devices, IoT is a great supplement to businesses. With IoT incorporated into their systems, businesses can get a real-time look into how everything works. This provides them with all of the analytics they need from the performance of machines to supply chain and logistics operations. What exactly makes up an IoT system? The four main components of a complete IoT system is hardware, connectivity, software, and a user interface.

The hardware side of the system consists of any type of device like a sensor or a microphone. This hardware collects data from a physical environment and/or performs actions in the environment. An example of this, which can be performed for our particular project needs, is a temperature sensor retrieving the temperature details from a bedroom or a vent opening after receiving a command. The hardware then needs to either make use of this data after collecting it or to receive data with instructions on what actions to perform.

For the hardware to somehow send the data it has retrieved to the cloud or receive commands from the cloud, it needs connectivity. To accomplish this, forms of connectivity like WiFi, satellite, or cellular are available for the hardware use. There are also more IoT-focused connectivity options like LoRa. A possible solution for our project would be to utilize a combination of forms of communication: the sensors would communicate to a central parent device through LoRa, and the central device would communicate with the cloud over WiFi.

Once the hardware is communicating with the cloud, there needs to be software hosted in the cloud that is responsible for analyzing and managing the information that is retrieved from the hardware. With this data, the software makes decisions for what happens next and responds to the hardware if any action needs to be performed. For instance, a scenario in our project would be if the bedroom temperature sensor senses a temperature that is higher than the desired room temperature, the cloud will need to tell the vent in that bedroom to open and allow airflow so that the temperature can decrease to the desired temperature.

After having hardware that communicates with the cloud, and software that controls the information and instructions that are being communicated, there needs to be a way for users to interact with the IoT system. A user interface accomplishes this. This can be anything from physical buttons, to a mature web-based application. For practicality, we have gone with a mobile application for our project. More on this can be found in section [3.2.4 Mobile Application Development](#).

The simplest way to dive into the IoT network and get your devices set up to communicate with each other is to use an IoT platform. An IoT platform gives developers a head start in incorporating their smart devices into an IoT system by providing built-in tools, features, and capabilities to make the life cycle of IoT a lot easier and less costly for the consumer. There are many big tech companies today that have developed their own IoT platforms for developers to delve into, with lots of provided documentation and

big community support. More information on these platforms can be found in [3.3.4 IoT Cloud Services](#).

3.2.6 Mobile Application Development

Once we have an IoT cloud platform setup to receive and manage data from the central hub via WiFi, we would need a way for the user to interface with the devices so that they can view and send this data. The most convenient and user-friendly way to accomplish this would be a mobile application. With a mobile application, the user will be able to easily view and change the temperature in each room that contains a smart vent with a click of a button on their mobile device.

The mobile app platform that we utilize for the mobile application must be able to use the provided mobile SDKs from the IoT cloud platforms mentioned in section [3.3.3 IoT Cloud Services](#) later covered, like Amazon Web Services or Microsoft Azure. Requests will constantly be sent between the mobile app and the IoT service.

The first thing to take into consideration when developing a mobile application is choosing which operating system to develop for. The two notable smartphone operating systems are Android OS and iOS. Although they will both be sufficient for our overall needs, they have differences in global market share, compatibility, performance, and user experience. Both the operating systems are leading in their way. No matter whether you are a well-established enterprise or a startup, developing a Native Android or iOS App is always a costly deal.

However, with today's modern technologies, it does not need to be a split decision between the two operating systems. There are many cross-platform frameworks that allow development of an application that can run on multiple platforms, including iOS, Android, and Windows, with a single codebase set. Some of the top choice cross-platform frameworks are React Native and Flutter.

3.2.6.1 Android & iOS

One of the first things to think about when a developer is considering a platform is the smartphone operating system market share. Knowing the global market share of the smartphone OS will give the developers an idea of the amount of users their application can reach. This is something Android has been on top of for a while and still is today. The Android OS makes up 73.86% of the global market, while Apple's iOS takes up about 23.35% of the global market (as of 2019).

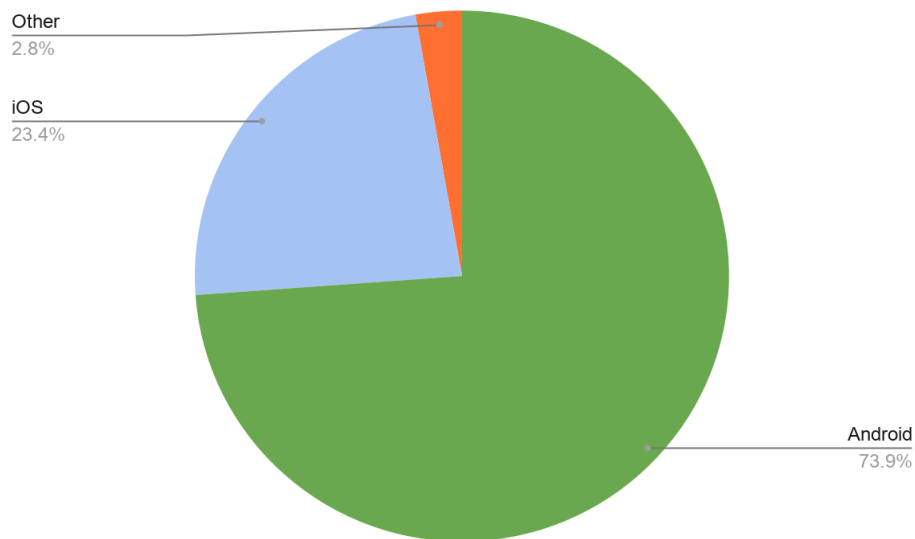


Figure 21: Global Market Share held by Major Smartphone OSes (2019)

Although Android OS is the leader in the global market and has been for a while, the margin between Android and iOS has been shrinking overtime, and for good reason. A big issue when developing an Android application is fragmentation. Due to the sheer number of smartphones that are built with the Android OS, it is difficult to tailor the user experience of your mobile application to each Android user. This is because of the vastly different specifications all of the Android devices have like screen size, screen resolution, camera quality, CPU performance, etc.

On the other hand, when building an iOS application, there is a much smaller family of devices you have to worry about - the iPhones and iPads. This makes it a lot easier to know what kind of device you need to develop your application for, and will lead to a better experience for the user. However, having a smaller set of devices to develop a mobile application for is not always a good thing. Apple sets many restrictions when it comes to application development, making it difficult to customize iOS applications.

3.2.6.2 Java vs Swift

When it comes to the specific programming languages the operating systems use for development, Java is the official language for Android while Swift is the official language for iOS. While they're both powerful and intuitive programming languages, they have different functionalities and serve in different ways.

Java is a class-based language that is object oriented, designed to have as much built-in functionality as possible so it doesn't have many dependencies. It is also designed to give the capability of running compiled Java code anywhere that supports Java without having to recompile the code. This makes it a general-purpose and platform-independent

programming language, intended to let developers write once, run anywhere (WORA). This includes both software and hardware systems, along with many platforms like Windows, Linux, MacOS, etc.

Java’s simplicity makes it easy to learn and understand. It is an object-oriented language just like C++ and is also based on C++’s syntax. It fully encapsulates the concepts of classes, inheritance, objects, polymorphism, and abstraction. This, along with its use of an automatic garbage collection, makes it a powerful and strong language that exceptionally handles memory and performance issues in the background. Finally, it is a distributed language which means most applications are free to use Java.

Swift on the other hand is a more modern language compared to Java, becoming available just in 2014. Although Apple designed it to be more optimized for application development for Apple products like the iPhones, iPads, and many of their other devices, it is a programming language that supports all devices. Since they have tailored this language for their products, they have made it exceptionally simple to understand how to write, compile, and run Swift code on their devices.

Table 3: Java vs Swift Overview

Java	Swift
<ul style="list-style-type: none"> ● Released by Sun Microsystem in 1995 ● Can be used for mobile applications. ● Known as one of the best programming languages. ● Defined as the first programming language in the industry with security. ● Complex readability for beginners. ● Currently not as popular as Swift. ● Low computer performance. ● Not predicted to become more popular in the future. 	<ul style="list-style-type: none"> ● Developed by Apple and released in 2014 ● Can be used for mobile applications ● Currently one of the more popular programming languages. ● Commonly used for platforms like Linux, macOS, iOS, Apple TV, Apple watch. ● Syntax is straightforward, easy to read and write. ● High computer performance. ● Predicted to become more popular in the future.

Furthermore, Swift is an open-source programming language which means it supports all platforms and anyone can fix bugs or issues, and share what they find to the community. Similar to Java, it is based on the syntax of C and C++. But unlike Java, you can run C++ or C code in Swift. This makes Swift easy to learn and use. It is a modern, high performance, and efficient language.

Today it seems like Apple and IBM (their Swift partners), are both betting on Swift becoming the language of the future. However, switching from Java could be a hassle for some companies, so why would they switch?

If we look strictly at performance, according to IBM, Swift is at the top right next to Java compared to other popular languages like JavaScript and Ruby.

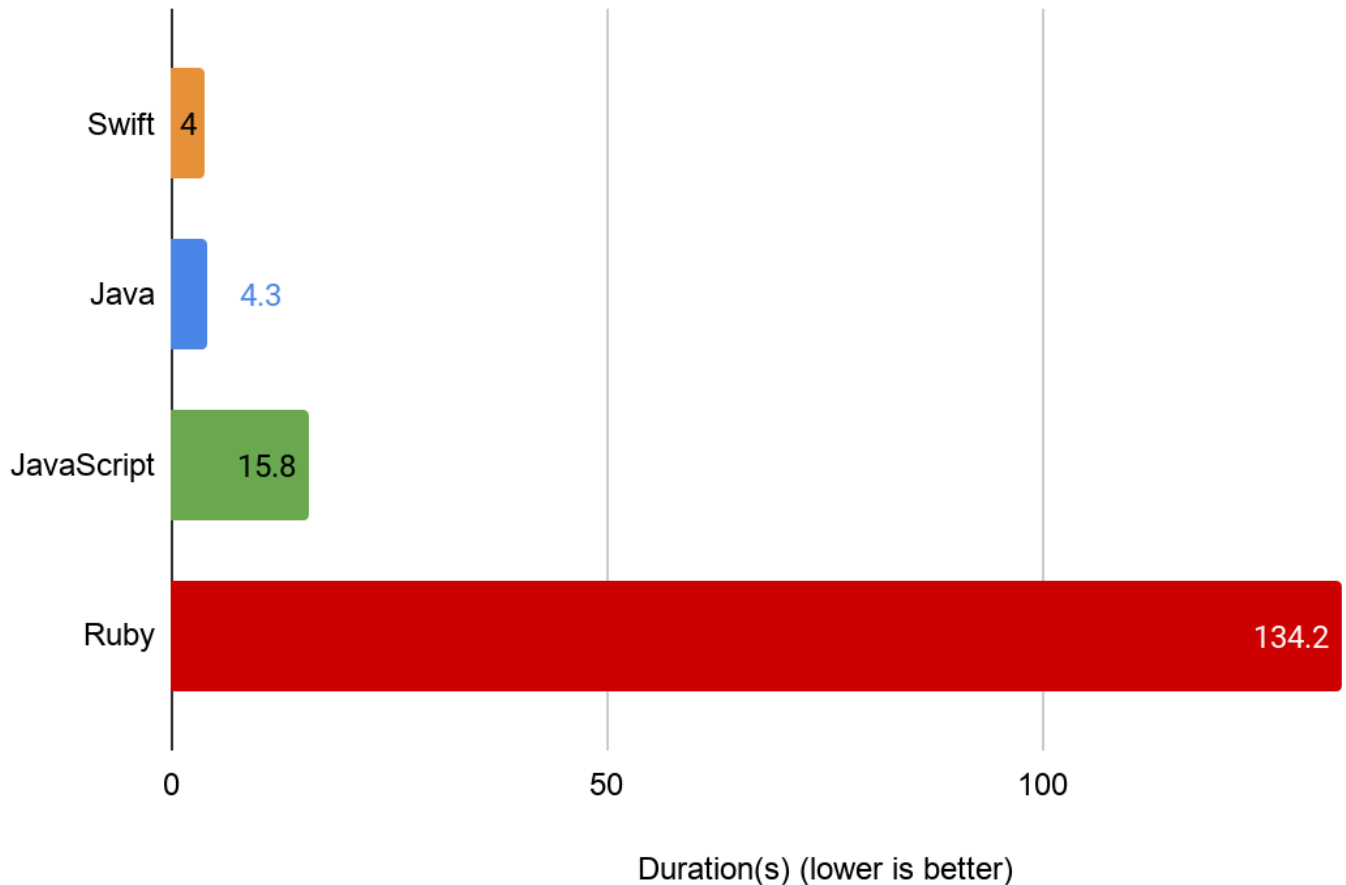


Figure 22: Application Performance of Popular Languages

However, having high application performance and speed isn't the only important thing. Swift's efficiency also gives it one of the lowest memory footprints compared to other languages. In the past, developers had to manually allocate and release Objective-C objects. But today, languages like Swift automatically prevent memory leaks for you using Automatic Reference Counting (ARC) which tracks and manages your application's memory usage for you. No work is required from the developer to manage memory themselves. In addition, they provide a method called lazy loading which allows you to load objects into memory only when you need them. Not only does this save memory, but it also reduces the loading time of the application and the screens.

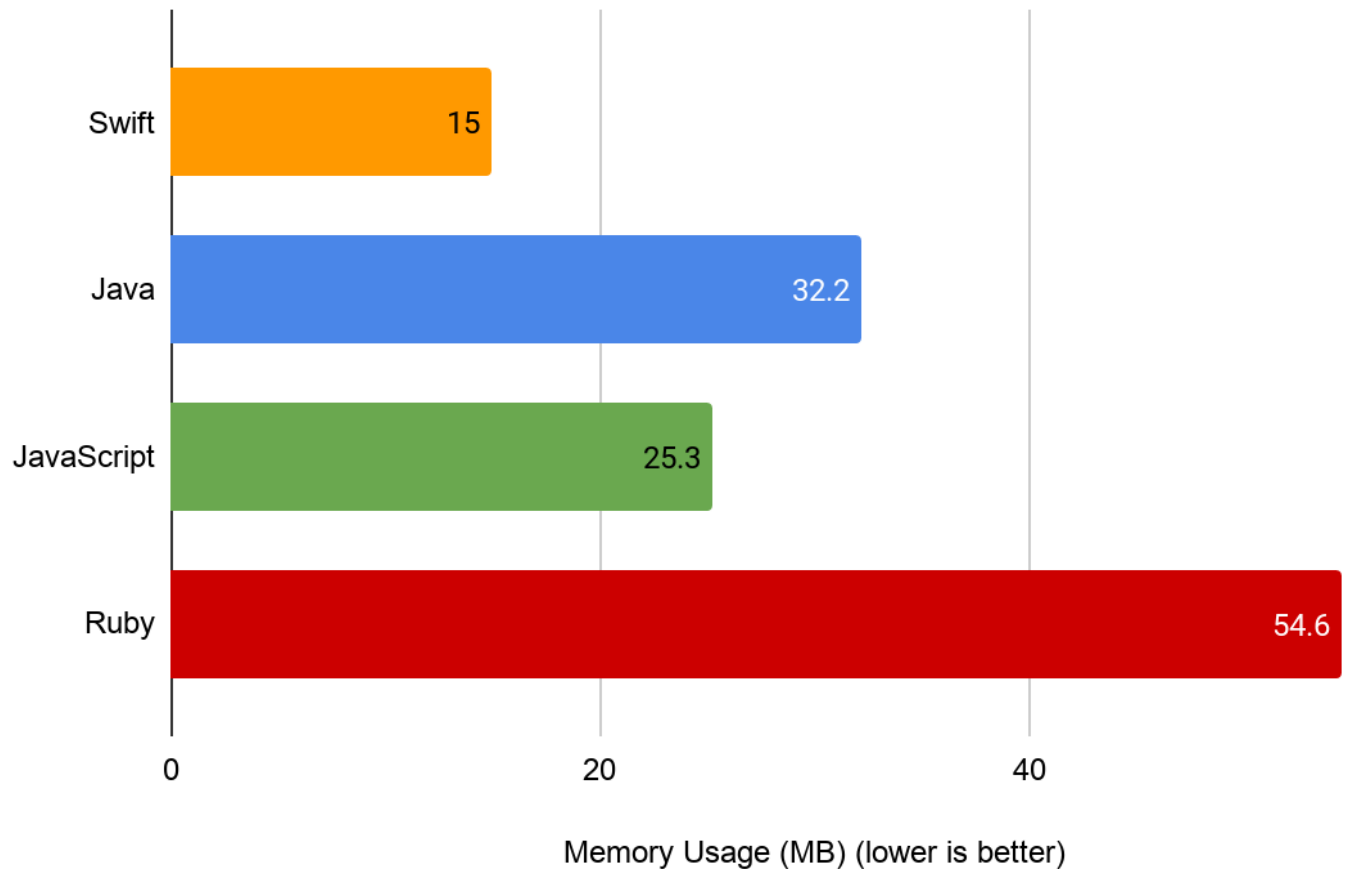


Figure 23: Memory Usage of Popular Languages

In conclusion, Swift and Java are different programming languages and have different methods, syntax, and functionalities. Although both are very powerful and simple languages, research is looking like Swift will be more useful in the future. However, Java is still one of the best languages in the tech world today.

3.2.6.3 React Native

Today in the modern tech industry, it is common for developers to choose a cross-platform framework that helps the development of an application for both Android and iOS. React Native is one of the most well known cross-platform frameworks. It is an open-source mobile application framework created by Facebook, which uses JavaScript. It is a framework branched off of Facebook's React framework, which is also an open-source, front end, JavaScript library for building user interfaces, but for web applications. React Native is used to develop applications across all platforms including iOS, macOS, tvOS, Android, Android TV, Windows, UWP, and Web by allowing developers to use the original React framework alongside native platform functionalities.

Developing a React Native application is simple, quick, and efficient. It is the perfect option for developers who have experience in JavaScript as there is no need to get familiar with Android specific Java or Apple iOS's Swift. It focuses on the UI of the application which makes the app load quickly and gives it a polished feel. Furthermore, using React Native allows for a wider global marketplace, which leads to making more profit. It is a solution to the dilemma of whether a developer should focus on making more profit by developing an iOS app or developing an Android app for user strength, as mentioned in [3.2.4.1 Android & iOS](#).

React Native being a library for JavaScript has its benefits. JavaScript gives the developer the ability to easily style and animate their UI with endless possibilities. This is because using JavaScript allows the use of HTML and CSS which supplements the JavaScript code. The HTML code is the structure and the apparent content of the pages. Then CSS is built on top of the HTML and is what styles the pages with the colors, shapes, fonts, sizes, and animations that the developer sets. The HTML and CSS are what make up the UI front-end. The JavaScript is then built on top HTML and CSS to give the pages functionality. This includes the behavior for everything that the user interacts with like the logic for what happens when a user clicks a button, the logic to take the input from a username and password field, or to send requests to servers.

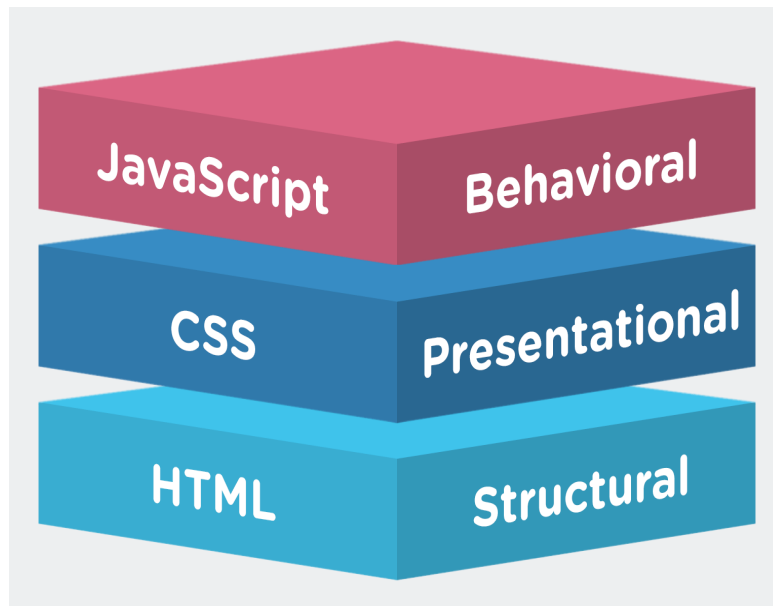


Figure 24: JS, HTML, & CSS Stack (Creative Commons: HackerNoon)

JavaScript is very useful and makes it easy to build an intuitive front-end. With that being said, React Native benefits greatly from using JavaScript. Usually, if developers wanted to create a mobile application using JavaScript, they would not have any capabilities that the native platform like Android or iOS provides. React Native gives the best of both worlds allowing a developer to use JavaScript, HTML, and CSS while still keeping the

native platform functionalities. While you do write JavaScript code with React Native, the components you define will end up rendering as native platform widgets. If a developer is familiar with the original React framework for the web, they will have no problems getting the hang of React Native. Similarly, if a developer has written applications in Java or Objective-C, they will recognize many of React Native's components. Fortunately for us, we have members with both React and Java experience.

React Native components look and behave just like React web components, but they get rendered as native UI widgets instead of HTML. For example, a “<View>” component would get rendered as a native “UIView” on iOS, and “android.view” on Android. When any data changes for your components, React Native automatically calculates what in the “<View>” needs to be changed and sends the changes to the native UI widgets.

By embracing the native platforms while still allowing the mobile application to share its codebase across all platforms, React Native gives developers the capability to create respectable apps that users will enjoy using, without having to increase their budgets to build two separate apps.

3.2.6.4 Flutter

Flutter is another well known cross-platform framework used for mobile application development. It is an open-source UI mobile application tool created by Google and used to develop applications across all platforms including iOS, macOS, Android, Linux, Windows, Google Fuchsia, and Web from a single codebase. Flutter apps are written in the Dart language, a client-optimized language developed by Google that is used for mobile, desktop, server, and web applications. However, Dart is not known by many developers, so why has Google choosed Dart over a more known language like JavaScript for Flutter?

Dart is a fairly new language for anyone not developing applications for Google. However, Google is a large and reputable company, so having a company like Google backing the programming language is a big advantage. Although JavaScript has a variety of libraries, there are many libraries that have little to no maintenance. Dart is also relatively faster in certain instances. But the fact of the matter is, JavaScript has been around way longer in the industry than Dart has, so it is a mature and stable language. This also means there is a lot more documentation out there for JavaScript. A lack of documentation for Dart means the learning curve for developers is much steeper.

The Flutter framework is rich with UI rendering components with Google's own libraries. This makes it so developers don't need to look for third party libraries for UI components. Everything you need comes bundled with Flutter.

In terms of performance, usually building an application with purely platform native languages is the most efficient and lightweight option. While the industry has seen cross-platform frameworks like React Native to be very useful, they have been significantly CPU-intensive. However, Flutter does a very good job to narrow that margin

between cross-platform performance and native performance. According to tests run by specialists at inVerita, Flutter impressively keeps up in performance against native languages like Swift and Objective-C, and even surpassing them in some cases. The tests ran utilized the Gauss-Legendre and Borwein algorithms on both iOS and Android, which tests for memory intensity and cpu intensity, respectively. This corresponds to the different types of performances on a mobile application like interacting with a phone API (accessing file system or retrieve GP location), rendering speed (frames per second with UI animations), and business logic (speed of memory calculations and alterations).

The first test conducted was on iOS which tests performance while stressing on memory (Gauss-Legendre algorithm). As we can see, Objective-C is the quickest for iOS app development, being 1.7 times quicker than Swift. React Native is 20 times slower than Objective-C. And finally, surprisingly Flutter came out to be 15% *faster* than Swift. That is exceptional for a cross-platform framework.

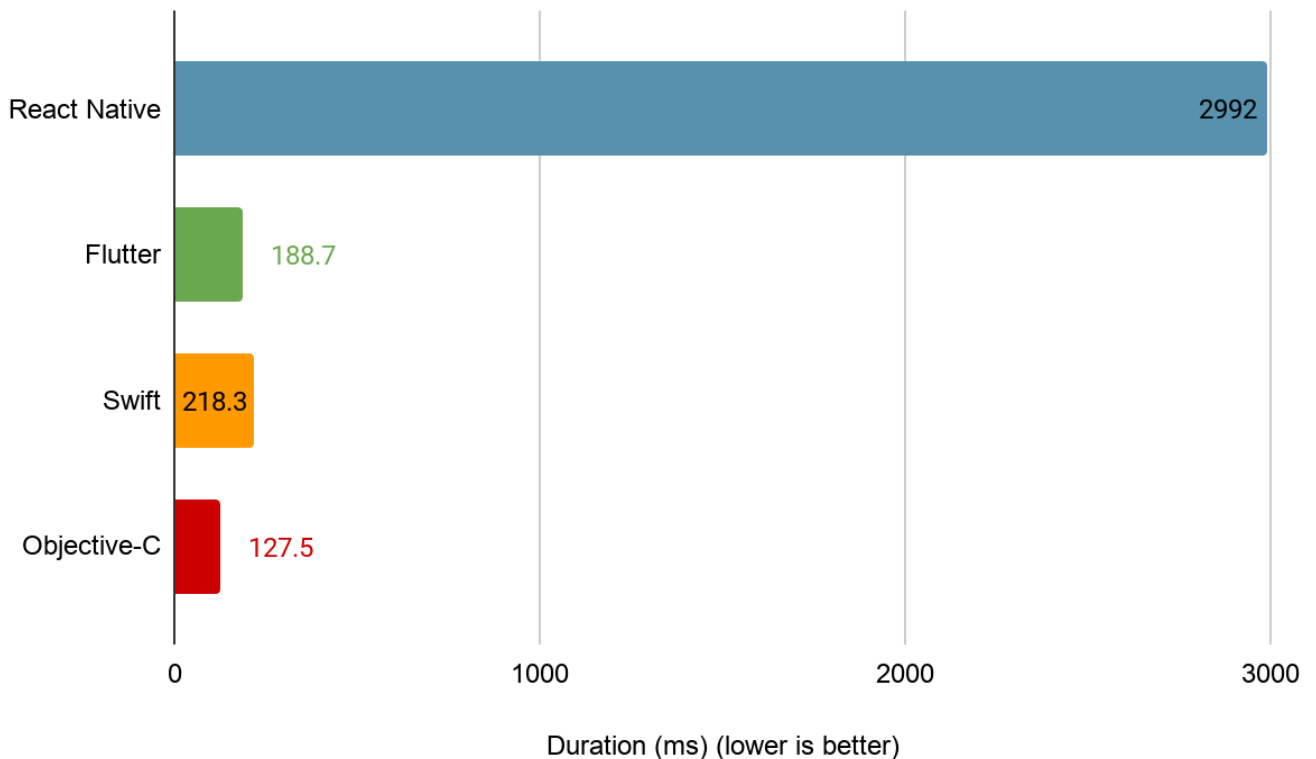


Figure 25: Gauss-Legendre algorithm test for iOS

The same test but conducted on Android with Java and Kotlin in replacement for Swift and Objective-C deemed similar results. Flutter competes very closely with Java and Kotlin, the native Android languages. On the other hand, React Native is around 15 times slower than the native languages.

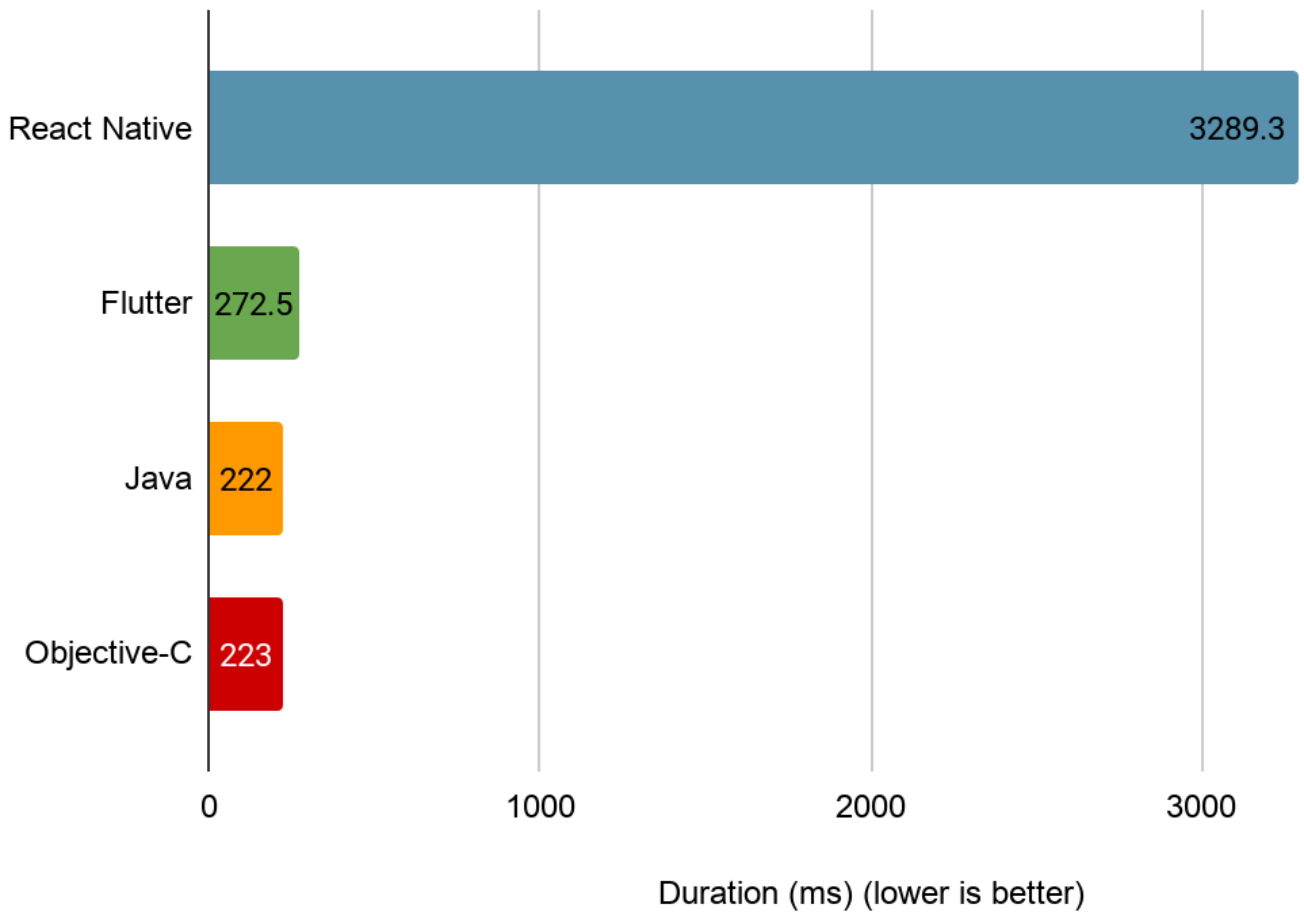


Figure 26: Gauss-Legendre algorithm test for Android

The second test performed on iOS was the Borwein algorithm which stressed on the CPU. Again, coming out on top was Objective-C which was 1.9 times quicker than Swift. React Native came out the slowest for a second time, being 15 times slower than Swift. And last but not least, Flutter was 5 times slower than Swift for this test. Although the gap was narrowed for the Gauss-Legendre algorithm between React Native and Flutter, the results were still in Flutter's favor by a lot.

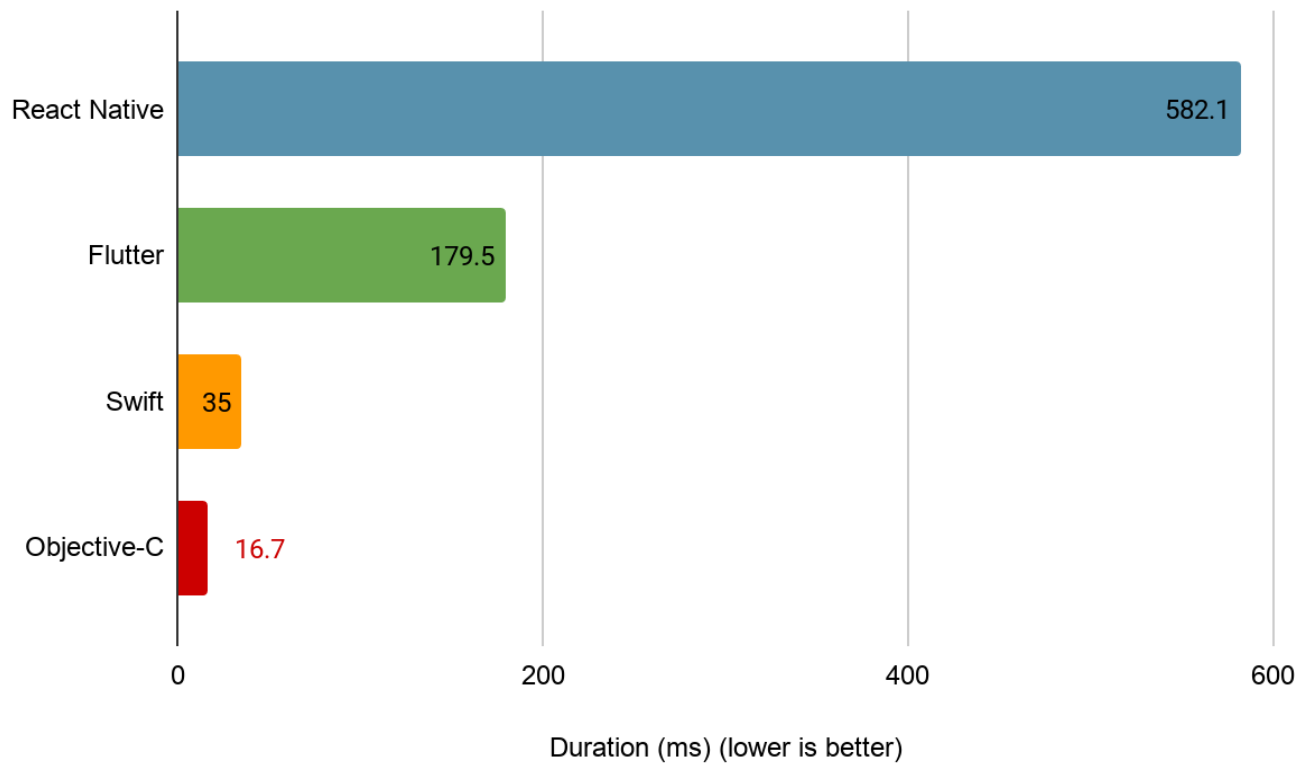


Figure 27: Borwein algorithm for iOS

Similarly, for the Borwein algorithm test for Android, the native languages came out on top which makes Java and Kotlin the best options for Android development. Flutter came after that, being 2 times slower than the native languages. And finally, React Native again being the slowest out of the four, being around 6 times slower than the native languages.

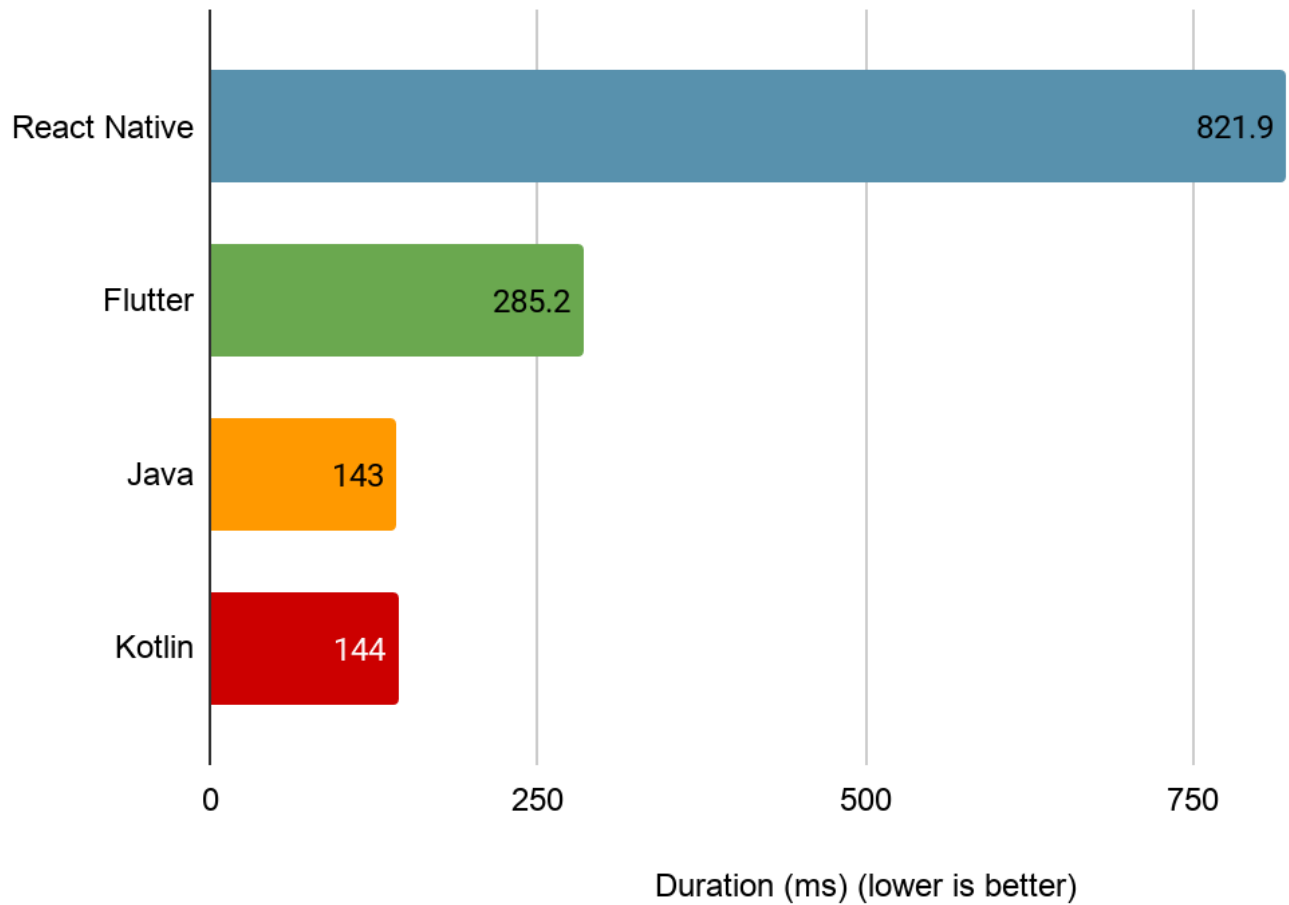


Figure 28: Borwein algorithm for Android

Besides performance, Flutter provides many customizable widgets that can be utilized for both Android and iOS. It also generally performs quicker than native libraries. With that being said, Flutter is the better option if the developer is already familiar with Dart. However, since Dart is uncharted waters for our group, choosing Flutter as our cross-platform framework would mean a steeper learning curve and more time spent learning.

3.2.6.5 *Arduino IoT Cloud Built-in Dashboard*

Arduino IoT Cloud’s dashboard is the center of attention in the cloud. It allows us to build our very own control panels to monitor and control our devices. We can build a unique dashboard for a unique thing, but we also have the possibility of monitoring several different things in the same dashboard. If we for example have five active devices, each linked to a thing, we can create one dashboard to rule them all.

But what is probably most valuable inside the dashboards are the different widgets that we can create. There are over 15 different widgets that we can use to represent the variables we create. We can choose from gauges, switches, RGB controls, maps, sliders and much more, to create powerful dashboards.

Table 4: Mobile App Platform Overview

Feature	Android SDK	iOS SDK	React Native	Flutter	Arduino IoT Cloud Dashboard
Language	Java or Kotlin	Swift or Objective-C	JavaScript	Dart	C++ (Arduino)
Nature of apps	Android only	iOS only	Cross-platform	Cross-platform	Cross-platform
Founded Year	September 2008	March 2008	March 2015	May 2017	Feb 2019
Developed By	Mostly Google	Apple	Facebook	Google	Arduino
Reusable Code	Allows reusable Java GUIs.	Swift allows reusable Views.	Allows code reuse but is restricted to few basic components.	Allows overwriting code. For recycling code purposes, Flutter is the best option.	Allows code reuse.
Third Party Libraries	Many open-source libraries are available but not maintained	Many useful libraries but not as abundant as other platforms.	Very popular framework so there are more third-party packages	Numerous third-party packages available.	Can import any libraries as needed.

	as well.		mainly due to utilizing Node.js.		
Popularity & Coverage	Very popular, with Java as the primary language.	Very popular, with Swift as the primary language.	65k+ Github stars makes it the most popular framework among developers.	30k+ Github means it is trending in the industry and it's growing.	Not very popular, but very easy to set up with an Arduino device.
Community	Strong	Strong	Strong	Weak	OK

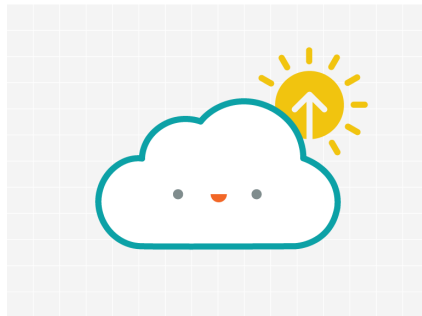


Figure 29: Arduino IoT Cloud Dashboard

Looking at the above table, any of the tools would satisfy our requirements for a mobile application. If we use the Android software development kit (SDK), we will have access to years of documentation of mobile development with Android. Also, the global market share of Android is much higher than iOS so our application will have a chance at exposure to more users. If we use the iOS SDK however, we will greatly benefit from a more modern language like Swift, compared to Android's Java. This would improve the performance and effectiveness of the application, while also still having a great amount of community support. While going natively with a separate Android application or Apple application can result in a greater user experience, the cross-platform frameworks like React Native and Flutter serve as arguably superior options when it comes to modern mobile applications. They provide exceptional performance while allowing the developer to only write code for one application that is compatible with both Android and iOS. They both are simple to use and have a great amount of documentation from their developers. However, although Flutter is proven to perform better than React Native in many cases, React Native seems to be the option that is more widely used and supported. There would be no issues while utilizing React Native due to their massive community support. In addition, since it is more popular because of its parent framework React, some members of our group have experience working with the framework which will make it

easier to ramp up and build an application with it. However, Arduino IoT Cloud dashboards, another platform that provides cross-platform eligibility, makes it very easy to set up a user interface since we will be using Arduino devices. As such, Arduino IoT Cloud is the best option for us as it is a reliable and modern cross-platform framework, and easy to use with good documentation.

3.2.7 Network topology

This section covers the network topology of our planned design and allows for a visual representation of how we plan on laying out the IoT system. This section will provide diagrams and illustrations on the network topology and the tentative layout of the project.

This section will also cover the differences between different network topology configurations and compares the benefits and downfalls to each approach. These pros and cons are outlined in a table later on in the section which covers these details in depth. The two main topologies which are discussed in this section include the mesh and star topologies which serve as the two competitors in terms of what the design will include. The illustrations below display the typical structure of these different topology configurations.

Each diagram associated with each topology configuration has a corresponding explanation and basic discussion regarding the implementation of the topology. The explanations also provide a basic overview on how these technologies could be incorporated into our final design. These network topologies will be later on built upon in order to design our final implementation.

3.2.7.1 *Star topology*

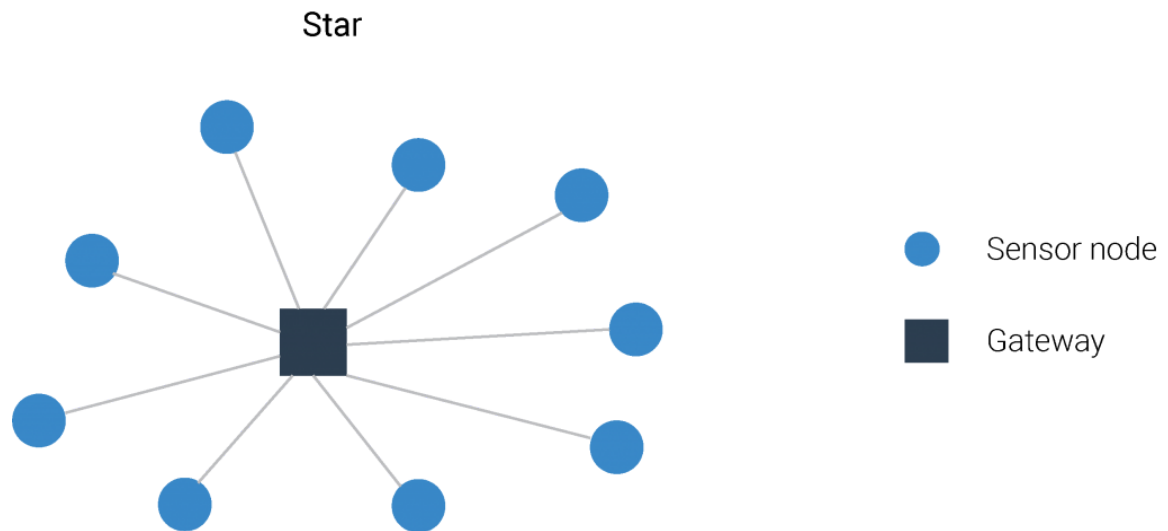


Figure 30: Star Topology

(Image source: <https://behrtech.com/blog/mesh-vs-star-topology/>)

Star topology is one of the methods to establish connections between devices within an IoT network. This wireless topology will connect every sensor node on an IoT network to a central hub, known as the gateway. The gateway can then connect the information to a cloud service, usually by wifi.

The central hub needs to handle all the data from the sensor nodes, as well as sending and requesting information from the IoT cloud. If the network has more advanced IoT devices that need to connect to another device, the connection will be relayed through the gateway. Therefore, the logic on the gateway can be very complex.

The connection between the gateway and sensor nodes are wireless point to point connection. This gives a security advantage since each connection between a sensor node and gateway is already established. Additionally, since each device has its own connection path to the gateway, troubleshooting the network is straightforward. If any device encountered a problem, from low battery, connection issue to being attacked by hackers, the gateway can detect and send reports to the user.

This, however, also makes the gateway the most vulnerable link of the IoT network. If errors occur on the gateway and crash it, the whole network stops. This means valuable information will not be able to be recorded, connection between end devices will be stopped completely. In more serious cases where IoT is responsible for security, such as protecting a house, the user's privacy will be at risk.

More downsides of star topology is that the size of the IoT network is limited to the range of the connection between the gateway and sensor, a hardware limitation. Being limited by hardware can create some big issues, since editing a hardware is not as easy as changing the software. Another disadvantage is that the connection between the node and the gateway can be interfered by environmental factors. This can include wall material blocking wireless signal, or rain affecting the signal attenuation.

Even though star topology is straightforward and simple to set up, having these limitations makes star topology heavily rely on the application and the environment that the IoT network operates in.

3.2.7.2 Mesh topology

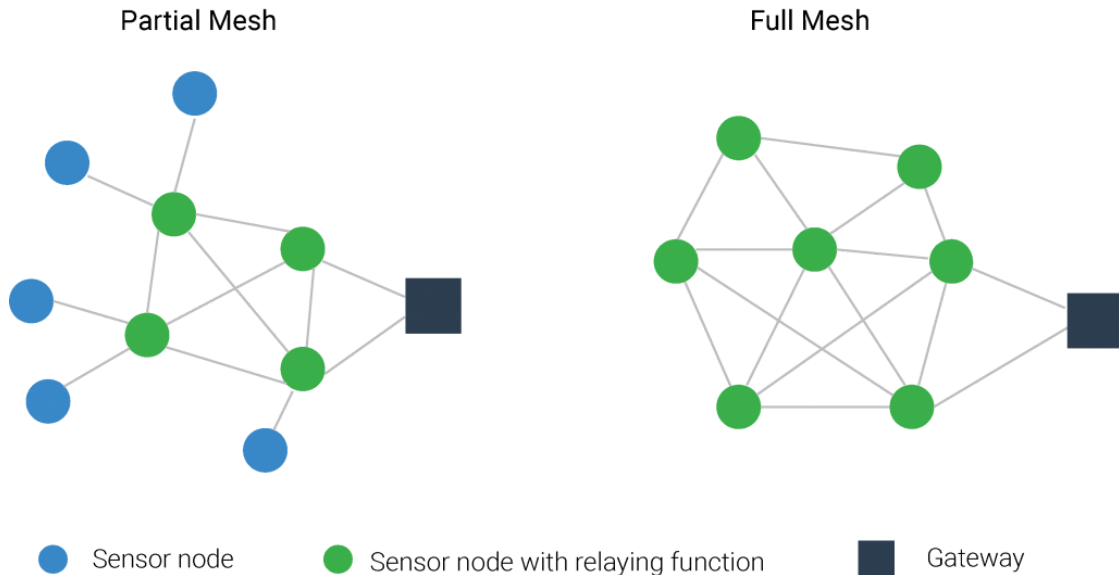


Figure 31: Partial VS Full Mesh Network

(image source: <https://behrtech.com/blog/mesh-vs-star-topology/>)

Mesh topology is an approach that overcomes the limitations of star topology. In mesh topology, each device can relay each other’s signal. A gateway device is in the mesh network to connect the entire IoT to the cloud service, so not all devices are required to perform this task. These characteristics allow a IoT mesh system to be extremely flexible, allowing any device to join the system. A part of the network can behave like star topology, creating a hybrid network.

Implementing a mesh topology can be very challenging, however. Firstly, since each device will make connections with more devices, there must be more security measures to make sure the network stays secured. Secondly, the network might require a lot of relay nodes to cover the entire application area, so implementation can be expensive. Finally, mesh topology requires each device to handle data autonomously, so the code can be very complex. Fortunately, there’s been a lot of protocols developed solely for IoT applications. By using an existing protocol designed for market application, developing a working prototype won’t be too much of a challenge.

3.2.7.3 Comparison

Table 5: Star vs Mesh Topology

	Star topology	Mesh topology

Pros	<ul style="list-style-type: none"> • Easy to create, only 2 possible roles (master and end device) • Can directly request information from an end device • More secured, since the route is already established. 	<ul style="list-style-type: none"> • Virtually indefinitely expandable network, dynamic network • Data will most likely reach the destination since there's another device that can relay the message • Can be set up in any environment
Cons	<ul style="list-style-type: none"> • Very centralized, not modular nor expandable. • Data processing by in the master device can be intensive • Due to environment factors (object obstruction, distance), a device can be isolated 	<ul style="list-style-type: none"> • Complex routing algorithm is required to create the communication system • Multiple device roles (hierarchy) is required • Security challenges, if 1 device got compromised, so does every data that goes through that device

Both topologies are suitable choices for our application. This is because an IoT network does not have a one size fit all solution. Therefore, more research should be done in order to determine which approach to take for the best solution. In the next section, we'll explore the possible mesh solutions.

3.2.8 Networking Standards

In these sections the topics which will be explored include networking standards such as Bluetooth Mesh, Zigbee, and Open Thread. These different technologies will be compared and depending on their receptive applicability to the design one will be chosen such that it meets our requirements and constraints.

Each technology is accompanied with a corresponding explanation of a description of how each technology works in addition to a little bit of history about each. Bluetooth for instance, is a very popular wireless standard and is used in many devices in order to connect technologies together over a wireless medium. In addition to this, each technology will have official specifications providing, reinforcing the information being presented throughout this section.

At the very end of the section after surveying these several technologies a table will be designed such that it compares the differences benefits and. This table will be integral in making the final decision on which networking standard will be used in the final implementation of the IoT design. Given how one of the design constraints is the wireless nature of the project it is imperative that the most appropriate option is chosen.

Some of the comparisons which will be found in the aforementioned table include device hierarchy, scalability, relaying data, package size, reliability, applicability. In addition to this, another table will be included which further compares Zigbee and Open Thread in terms of longevity, authentication, cloud integration, and the approach used when addressing.

3.2.8.1 Bluetooth Mesh

The Bluetooth Mesh is based on BLE (bluetooth low energy). The bluetooth mesh is a different protocol stack from the original BLE. The first mesh model was conceived in 2014 and adopted on July 13, 2017. This is the youngest technology model in IoT application. This can present some challenges since the technology might not fully mature; however, at the same time, this can be a big opportunity to get on board and involve with the model.

BLE mesh is a managed flood network

Message relay and managed flooding

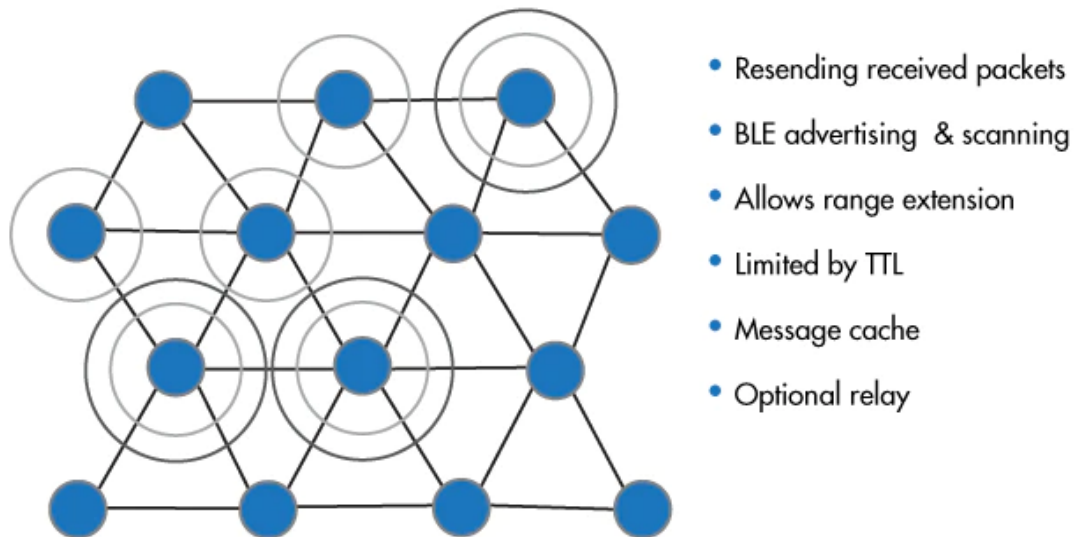


Figure 32: BLE Mesh network (source: <https://www.electronicdesign.com/technologies/iot/article/21807630/simplifying-ble-mesh-integration>)

The bluetooth mesh operates by what's known as a flood network principle. In some other IoT mesh networks like Zigbee, the devices first establish a routing path and then send data through established paths. Routing protocol can be complex, and has to be updated whenever a new device joined, or a device stopped working (very unpredictable).

Bluetooth mesh overcomes this complexity of routing algorithm by sending the data by flooding the message to all nearby nodes.

Flooding is when a device broadcasts the data to every nearby node, relaying the data to every single device. The idea is to have every device behave the same way, flooding and relaying the data if it's not for them, and hope that the designated device receives the data. The bluetooth mesh protocol stack is designed to ensure that the flood network is controlled and secured.

To protect against relay attacks, each message is encrypted and requires authentication. When the network is first set up, there's a process known as provisioning. The provisioner securely distributed network key and address space for each device. This will prevent other devices from eavesdropper. To prevent the message from echoing in the mesh indefinitely, each message has a TTL (time to live). After every relay, the TTL value will be decreased. Once the TTL expires, the message won't be relayed.

As the name suggests, BLE (bluetooth low energy) has a small packet size to minimize operation time. Bluetooth mesh stack expanded on BLE packet layout as follows:

	1		1	3	2	2	12 or 16	4 or 8
IVI	Network ID	CTL	TTL	Sequence Number	Source Address	Dest Address	Packet Payload	NWK MIC

Figure 33: Bluetooth LE package structure

The packet payload is from 12 to 16 bits. This is perfect for applications where small, simple data is required to be transmitted. However, this also means that for applications where large data is required to be sent, a lot of packages will be flooding the network whenever a device begins to transmit. This means it's very important to access the amount of data the network is required to be sent. If it's just a simple instruction to turn on/off a device, or some 8 bits value of a sensor reading, a bluetooth mesh network is perfect.

Another factor to consider is the size of the network. Because each message is being flooded, to reach from point to point on opposite sides of the network can require a lot of hops. The message needs to configure the TTL just enough so that it's long enough to reach the final destination, and short enough to not interfere with other devices in the network. As the network grows larger, chances that the message can't reach the destination also increases, hence the reliability of the network is reduced.

3.2.8.2 Zigbee

ZigBee is the most mature mesh protocol for IoT. The standard was standardized by Zigbee Alliance on June 13, 2005. Using mature technologies such as Zigbee provide a lot of benefits, since the protocol has been through the test of time. Countless experts have used and improved the Zigbee protocols. This also means that there's an abundance

of market ready products, for both consumers and developers, allowing integrating more devices into Zigbee network an ease. Zigbee are also very low-powered, each device can run off a button cell battery for years, making it perfect for IoT applications.

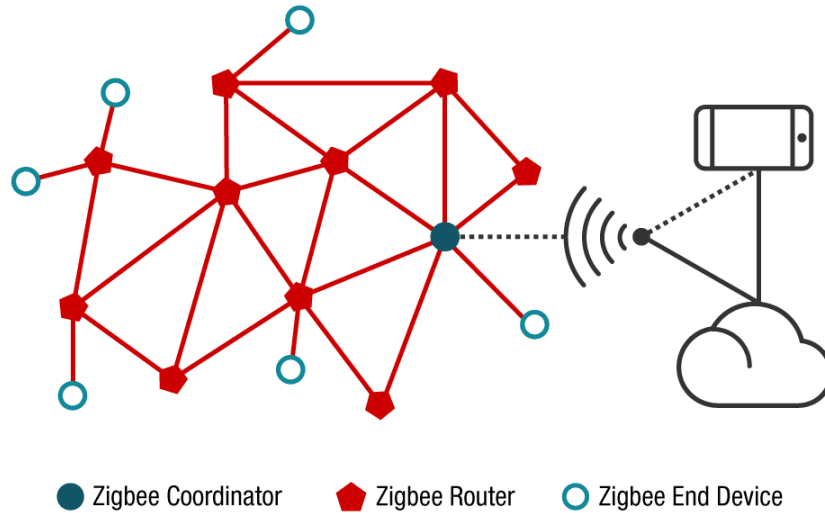


Figure 34: (source: <https://www.ti.com/wireless-connectivity/zigbee/overview.html>)

Unlike bluetooth mesh, Zigbee only performs flood operations when setting up the routes. This flood operation is initiated by a central device, known as the Zigbee coordinator. The Zigbee coordinator is responsible for creating the routing algorithm for each of the Zigbee routers inside the network. Once the routing tables are established, the Zigbee router will send information to its designated routes. The Zigbee coordinator usually acts as a gateway device as well, connecting the entire IoT network to cloud services. Zigbee end devices aren't required to relay messages, so they are usually simplified to reduce cost.

An analogy of a Zigbee network is, well, bees. The coordinators are like queen bees, in charge of the entire network; the routers are like worker bees, in charge of keeping the network connected; the end devices, sensor nodes, are like drones bees, in charge of interacting directly with the outside world. Zigbee devices work together in a hierarchy, just like a nest of bees. If a router fails, the network will self heal, redirecting the routes to make sure every device is connected.

Zigbee is also very flexible, it allows different kinds of connection between the devices within a network. This can range from star topology, to tree topology, to cluster tree topology, and finally mesh topology. This makes Zigbee suitable for most IoT applications, developers just need to configure the layout of the devices.

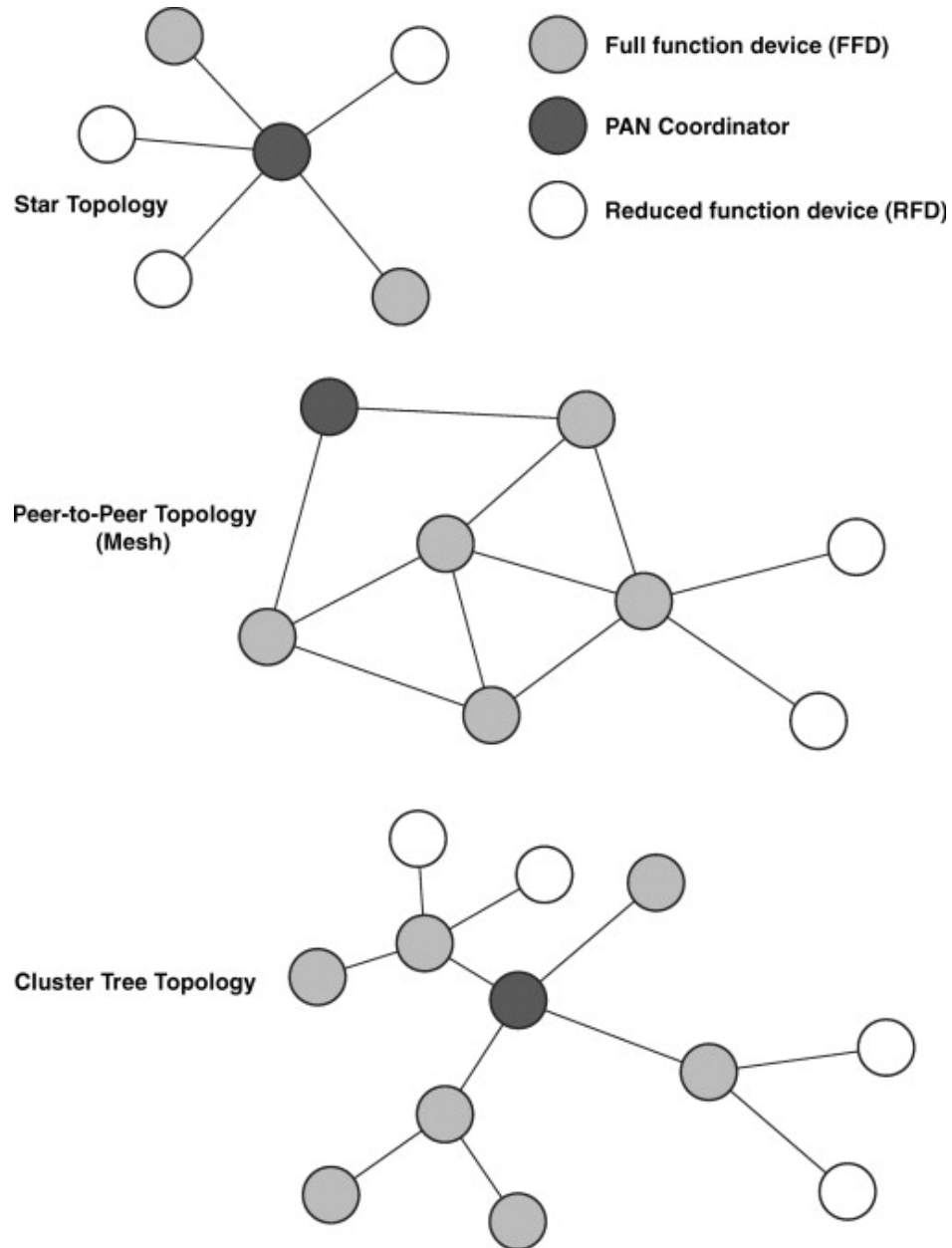
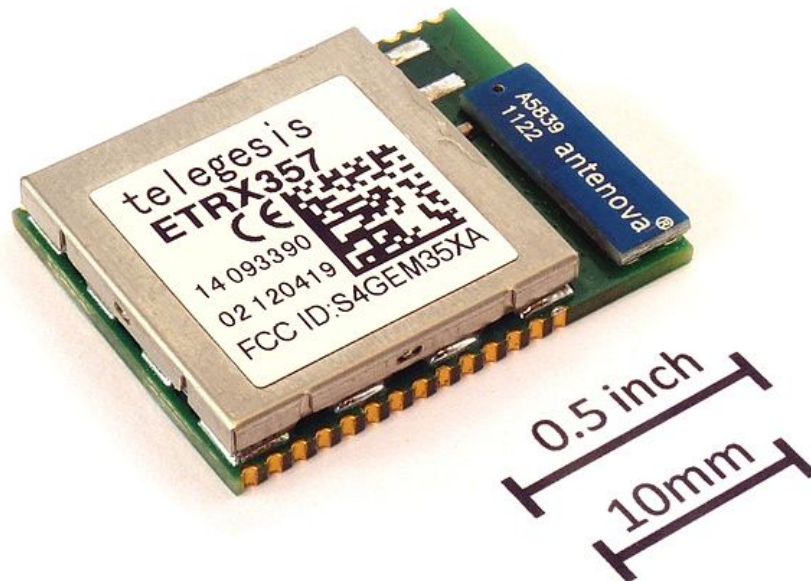


Figure 35: (source: <https://www.sciencedirect.com/topics/computer-science/reduced-function-device>)

A network of Zigbee devices can be considered as a personal area network (PAN). Every device is connected and can communicate with each other. Each device has a 16 bits address, an address that only Zigbee devices can recognize. This is due to when the protocol is created, the connection protocol is created from ground up. Therefore, even though it looks and operates like internet devices, it cannot connect with any other protocol.

This makes Zigbee gateway devices complicated, as there must be 2 separate devices. One device that runs and operates Zigbee protocol to connect with other Zigbee devices in the network, while another device connects to the internet. These 2 devices need to communicate with each other in order to make up a gateway device. One solution is to use Zigbee Network Processor (ZNP) interface, developed by Texas Instrument. This allows any microcontrollers to communicate with the Zigbee using UART.

Another approach to Zigbee gateway devices is to use a gateway device that's already on the market. As long as the gateway device (commonly known as Zigbee hub by consumers) allows connections with a Zigbee 3.0 device, we can create a device that can pair with a gateway device. This solution also opens up the possibility to easily integrate other consumer products like IoT sensors into our project. One example is AduroSmart ERIA, this hub allows every Zigbee device to pair up. More confirmation on whether this hub supports every Zigbee 3.0 is required, however. A custom Zigbee application might also be needed to utilize our products.



*Figure 36: A Zigbee Transceiver
(Creative Commons: AutolycusQ)*

https://commons.wikimedia.org/wiki/File:ETRX357_ZigBee_module_with_size_ref.JPG

Zigbee packages allow more data per package than Bluetooth mesh. This allows more data to be sent per package than Bluetooth mesh. Since each package will travel from 1 router to the next, there won't be as much interference on the network as flooding at every router. This makes Zigbee more suitable on applications that have lots of traffic and large data size.

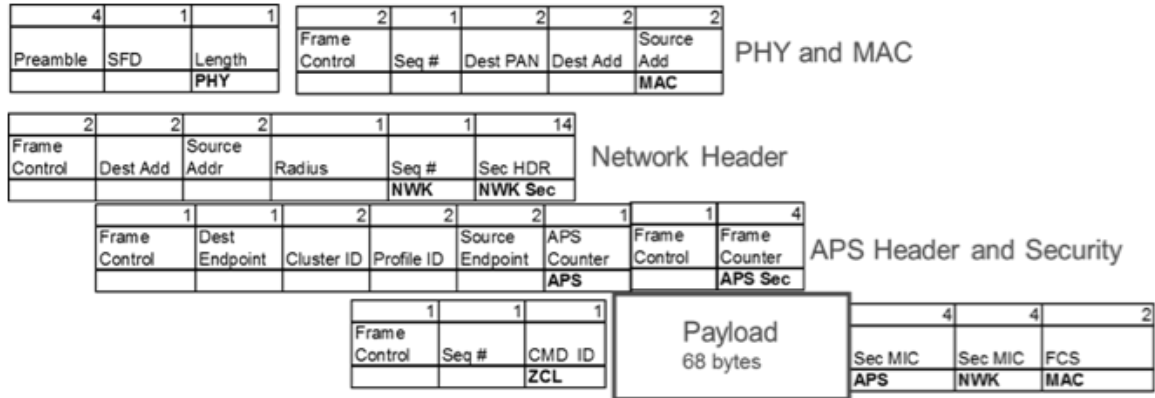


Figure 37: Zigbee Header and Package format

Zigbee has one big disadvantage, however. Despite allowing a mesh network, every data from any node must first pass through the Zigbee coordinator. That means if node A wants to send a message to node B, the data needs to go to the zigbee coordinator, only then, it arrives at B. This happens regardless of how close A and B are. This means that if the Zigbee coordinator fails, the entire network will stop working. The Zigbee network is still very centralized.

3.2.8.3 Open Thread

Open Thread is dubbed the successor of Zigbee. Although the protocol is quite young, it has developed a substantial market, having big Tech companies like Google creating Open Thread-based consumer products. The main difference is that while Zigbee is built from the ground up, having 16 bits for address, Open Thread devices use IPv6 addresses, allowing each device to interface directly with existing IPv6 based networks like WiFi.

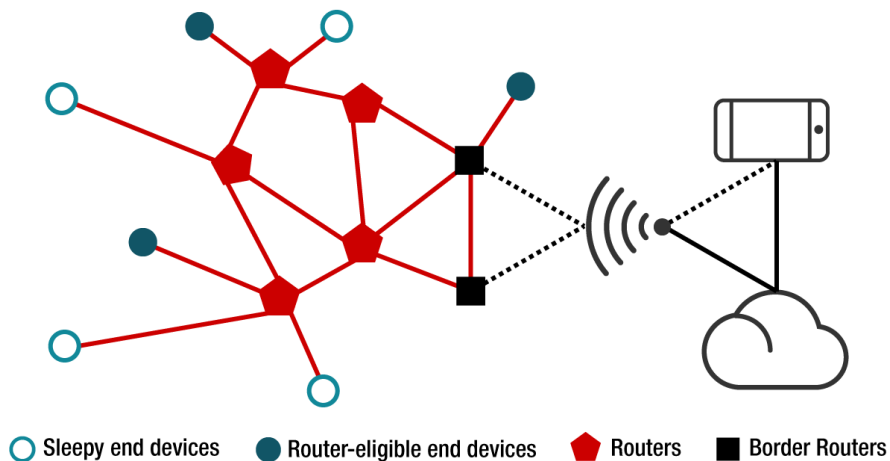


Figure 38: A Mesh network

Open Thread fixes Zigbee's biggest disadvantage by decentralizing the coordinator. It still has a hierarchy of devices; however, it has a self-elect function if one of the previous devices failed. In addition to routers, there are also router eligible devices that's ready to function as routers to keep every device connected. There can also be more than one gateway devices, allowing more access points throughout the network. This is made possible due to IPv6 addresses.

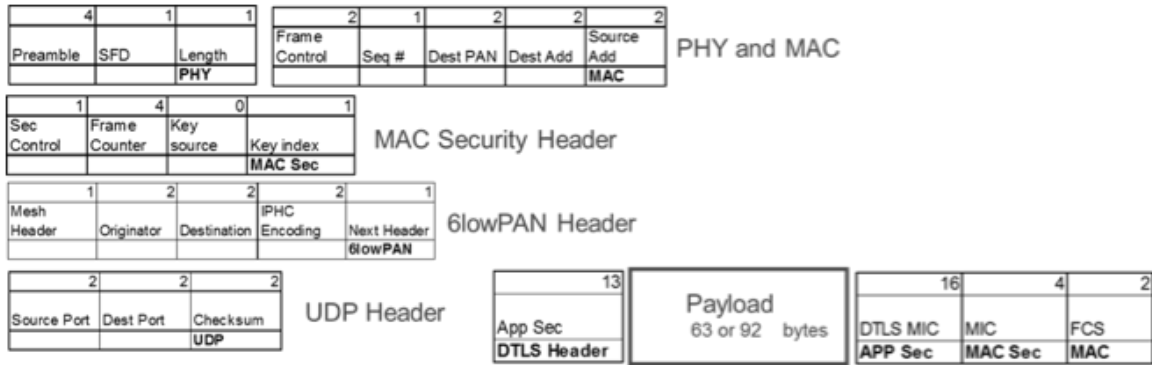


Figure 39: Packet Configuration of a Zigbee Package

Thread network allows more gateway devices, known as border routers, to be added to the network. This allows more access points to the internet in the IoT network. This feature with extra router eligible devices makes Thread protocol suitable for very large scale industrial applications.

This, however, doesn't mean that Thread is not a good application for household uses. In fact, Google's nest products were built on Thread. Having a large company such as Google shows promising signs for the Thread protocol.

3.2.8.4 Mesh protocol comparison

Since Zigbee and Open thread operate almost the same, we'll first compare the bluetooth mesh to the two, and then compare each of them against each other. First up, bluetooth mesh compared to Zigbee and Open Thread:

Table 6: BT Mesh vs Zigbee/Open Thread

	Bluetooth Mesh	Zigbee/Open Thread
Device hierarchy	Every device operates the same, repetitive, making it easy to manufacture.	Requires different component, each with its own role, can be very complex
Expanding adding new devices	A device need to enter advertise mode, sending signal that a new device has joined	Usually require an application (smartphone app) to set up routes for the data relay

	Bluetooth Mesh	Zigbee/Open Thread
Relaying data	Relay the data by flooding the network, letting every device around know the message, may interfere the traffic	Using established routes on the setup phrase, so only relays that required to the destination is used
Package size	Very small payload size, 12-16 bytes	Package size can varies, up to 60-90 bytes
Reliability	Reliable, but becomes less once the network gets large, as the message needs to hop over many devices.	Capable of mesh healing, able to redirect if a router fails, making sure that the data will always reach its destination.
Application	Suitable for small application with little traffic	Can handle large scale, industrial applications with a lot of traffic

Bluetooth mesh is more suitable with small, low traffic networks, while Zigbee and Open Thread can handle almost any application. The most advantages of bluetooth mesh is its repeatability, as its networks can be devices that were designed and programmed similarly. . Sophisticated applications can cause large latency and potential data loss. However, if the application is small and requires a small amount of data to be sent, bluetooth mesh is a variable option. Next, we'll look at the difference between Zigbee and Open Thread.

Table 7: Zigbee vs Open Thread

	Zigbee	Open Thread
Longevity	First revision in 2005, the most mature mesh network protocol, very reliable	First revision in 2015, however, Thread receives a lot of support from big company, such as Google and Samsung
Authentication	Proximity base commissioning, or push-button. Mobile apps were built to assist setting up the network	Fully based on smartphone to authenticate, which uses a QR code (each device need to be Thread-certified and get its own QR code)
Cloud integration	Zigbee gateway device, which compose of 2 separate device that translate internet and Zigbee protocol	Thread border router, can directly link the device to the internet thanks to the devices' address.

	Zigbee	Open Thread
Address	16-bit address, requires translation on the gateway device to connect with cloud.	IPv6 address, which can be directly addressed through border routers

Other than that, there's not much difference when applying Zigbee or Open Thread onto an application. It's heavy based on the preference of the developers. Zigbee is much more mature, so there's more documentation, software development kit (SDK), and off the shelf products that can be easily integrated into the IoT network. Meanwhile, Open Thread is younger, has a more intuitive communication method and a lot of potential development.

Security is an issue that all mesh protocols have to have to deal with, each with different approaches. Since cryptology is a very complex subject, we'll just have to leave to the experts to develop the protocols. The older the technology, the more trustworthy it is, as more tests and development were put into it. This makes Zigbee have an upper edge since it is the most mature. However, Bluetooth mesh stack was developed on existing Bluetooth (which was first introduced in 7 May 1989); Open Thread is built based on Zigbee. Therefore, it's safe to assume that all mesh protocols have reliable and trustworthy security.

3.2.8.5 *LoRa*

This section will primarily focus on the LoRA technology and its significance to our proposed design. In addition to this, the section will also cover the approach which will be used in implementing LoRA into the final IoT implementation. Also, the communication technique used by LoRA will be discussed along with the specifications of the communication protocol. In addition to this, this section will also cover why LoRA was chosen as opposed to other alternatives and the capabilities offered by the LoRA modules.

This section also plans to cover the methods in which are used in order to keep LoRA at a such a low power draw but retain its effective use. Considering the nature of the project in question as stated before regarding the constraints and restrictions the power consumption of the system as a whole must not exceed a predefined wattage. Exactly how LoRA manages to achieve such impressive performance with relatively low power consumption will be thoroughly reviewed and explained during this part of the report.

A topic of extreme importance is that of the chirp spread spectrum modulation technique used by the LoRA chips in allowing for long distance communication while conserving power. This chirp spread spectrum approach to signal processing will be integral to the long-term reliability of our implementation and will serve as the main form of communication between the boards in which the IoT system is based on.

In addition to the aforementioned material which will be covered in this section, this section will also provide images depicting the board layouts as well as figures of the chirp spread spectrum graph. These figures provide a better representation as to how these modules and communication techniques are defined as opposed to a simple description. The graph for instance shows that it is a derivative of a sign graph with an altered frequency with respect to time.

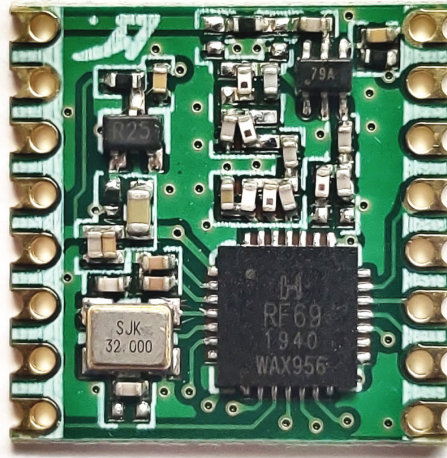


Figure 40: A LoRa module

Figure 9 showcases a typical LoRa module. LoRa which stands for long range is a low power wireless communication protocol developed by Semtech. LoRa allows for transmission of bits via radio frequencies. LoRa operates at sub-gigahertz frequencies of 433 MHz to 923 MHz depending on the location or continent it is being operated in. LoRa is able to achieve a data transfer rate up to 27 kbps. This however depends on the spreading factor of the signal.

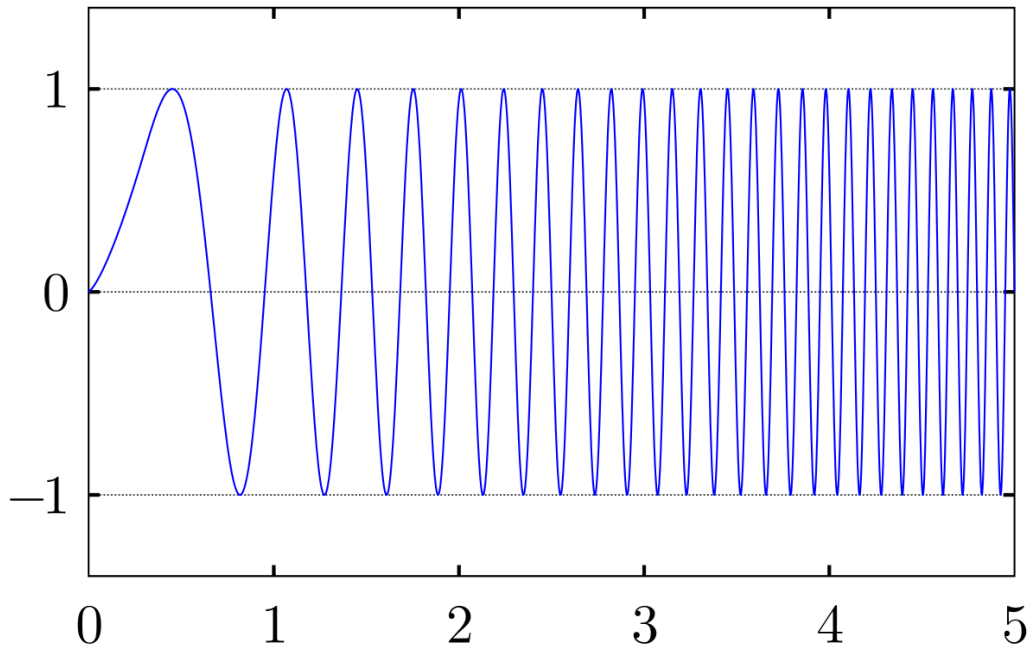
LoRa can transmit at distances up to 5 km to 15 km depending on the environment in which the device is transmitting. In more urban crowded areas the range of the LoRa module will decrease down to a maximum of 5 km. However, when in more open rural type areas the range of LoRa will increase to an absolute maximum of 15 km. In addition to this, due to the extremely low power consumption of these LoRa devices it is possible for battery operated devices to last for 10 years.

LoRa is used in many IoT applications and fits the role we are trying to fill in our system's design. LoRa will be integral to our configuration of this system in a home as we are designing a wireless solution in the effort to avoid routing cables through walls or other means. In addition to the benefit of LoRa's wireless nature, the fact that LoRa consumes a small amount of power makes it ideal for our application.

In order to achieve the low power consumption coupled with its long range transmission capability, LoRa uses chirp spread spectrum modulation. Chirp spread spectrum is another modulation scheme that spread a narrow band limited signal into a wide bandwidth. Spread spectrum approach has been used before and was first used in the

military. The concept was to create a communication system that can't be disrupted, also known as jamming the signal, by the enemy.

Previous approach to spread spectrum modulation has been frequency-hopping spread spectrum (FHSS) and direct-sequence spread spectrum (DSSS). These 2 modulation techniques require a pseudo randomly generated noise to modulate the original signal. Both the sender and receiver have to agree on how the noise is generated in order to decode the signal. In FHSS, the signal is spread by changing the frequency repeatedly in the pattern of the pseudo random noise; while in DSSS, the original signal is modulated with a pseudo random bit known as the spreading sequence.



*Figure 41: Chirp spread spectrum Graph
(Creative Commons: Georg-Johann)*

<https://commons.wikimedia.org/wiki/File:Linear-chirp.svg>

Unlike the previous spread spectrum modulation, LoRa chirp spread spectrum doesn't require a pseudo randomly generated noise, it is based solely on the linearity of its chirp pulse. A chirp pulse (shown in figure 41) is a sinusoidal wave that increases linearly in frequency over time, making the pulse very wide in frequency. LoRa will modulate the original signal with this chirp pulse, resulting signal power is spread over a wide spectrum.

Before the output signal is modulated, the chirp pulse is repeated indefinitely. LoRa will then modulate each segment of the pulses to represent the original data by shifting the cylinder. Each symbol of the data is represented in a certain pattern of frequency. These patterns are what carry the information.

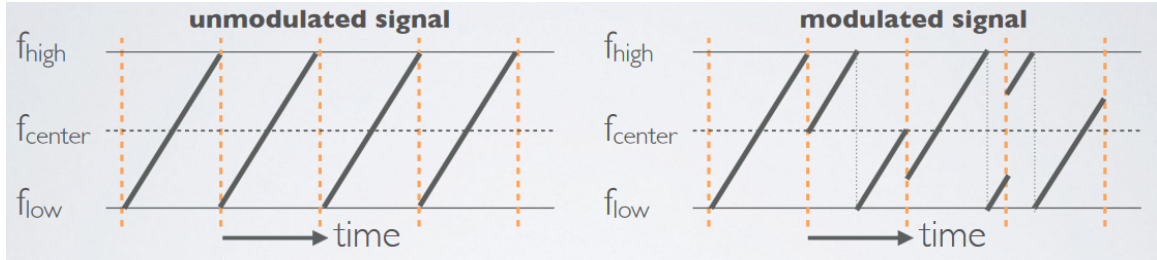


Figure 42: Output pattern of LoRa modulation

As seen in figure 42, there are 4 distinct patterns after the signal is modulated. The receiver just needs to catch how the pattern changes in order to decode a symbol of the signal. Since the pattern is quite distinct, the receiver doesn't require a highly accurate nor synchronized clock to capture the packet. The slope in which the frequency increases linearly can be adjusted with spread factor. The higher the spread factor, the longer the chirp pulse and linear slope, meaning slower the data transfer rate.

The linear patterns of each package are predictable, so it allows the receiver to identify and correct errors. This means LoRa can eliminate common errors caused by the Doppler effect, where a signal receives higher frequencies than it should, even at low frequencies. Additionally, chirp spectrum spreading has high noise immunity and multi-path fading, similar to other spread spectrum modulations.

The signal to noise ratio of LoRa also tends to be low due to the increase in chirp-rate of the signal. This in turn results in less noise as the data signal is multiplied by a corresponding chip sequence. This phenomenon is directly associated with the spreading factor which determines the amount of spreading code which is applied to the data signal.

These spreading factors have a significant effect on the bit rate and range of the LoRa communication and also result in differing times on air. That is, how long it takes for an 11-byte payload to be transmitted. Generally the lower the spreading factor, the higher the bit transmission rate and the range and time on air both decrease.

When the spreading factor is increased the range and time on air increase however the bit transmission rate decreases. The trade-off in this situation is that of speed/downtime vs range and this must be considered based on the application of LoRa.

3.2.9 Embedded development

As we're developing an IoT system on microcontrollers, it's important to access their documentation. Good documentation, as well as support from the developer community, are invaluable resources. We are looking for well documented, strong user communities to make sure that the development of our system goes smoothly. The quality of these factors will weigh in our final decision on which microcontroller to be used in the final product.

We will examine possible products as well as their communities, documentation, online guides, etc. This includes Arduino (used in the star topology approach), TI chips (used in the mesh approach, specifically Zigbee and Open Thread), and SiliconLab (used in the bluetooth mesh approach). We'll examine their products to rate its ease of use as well as its documentation. We'll end it with a comparison table of each development environment.

3.2.9.1 *Arduino*

Arduino is an open source development board powered by an ATmega microprocessor. The concept was first started in 2005 as an easy method to program ARM-based microprocessors. Over time, the project grows in popularity and becomes one of the go-to development boards. Huge makers communities were formed around Arduino, making this open source project more and more accessible. These boards can be made cheaply, making it an ideal option for developing our projects.

The goal of Arduino is simple, to simplify the process of programming a microprocessor as much as possible by hiding all of the complications involved with programming embedded systems. An Arduino board is a simple microprocessor with necessity components on board, such as an USB to serial converter, crystal clock, some LED for testing, etc. Thanks to the on board USB to serial converter, an Arduino can be programmed directly from a USB port with needed extra accessory. Meanwhile, the crystal clock allows the programs to operate at precise timing. All of these features available on board allow almost anyone, no matter the experience, to get started with programming microprocessors immediately out of the box.

The success of the Arduino open source project is mostly due to its IDE, Integrated Development Environment. As more and more people are involved in the project, the more sophisticated the IDE and its library becomes. The IDE allows anyone to program an Arduino without understanding how ATmega, or any of its variations that's used to power the board, function. The board and its IDE has automatically configured the settings, so users aren't required to read and understand its parameters, making programming an embedded system a child's play, literally.



*Figure 42: Arduino Development Board
(Creative Commons: SparkFun Electronics)*

https://commons.wikimedia.org/wiki/File:Arduino_Uno_-_R3.jpg

Figure 11 displays a typical arduino development board. These boards typically consist of an ATmega microcontroller in addition to many peripheral devices and modules on the printed circuit board in which the MCU is connected. In our implementation we hope to use an ATmega MCU in order to handle all of the logical operations associated with this IoT system.

This ATmega processor will then be connected to a custom PCB which will allow for it to be applicable to our implementation. However the programming side of this equation will be similar to that of other Arduino development boards and products as they are simply using the same MCU. This allows for the functionality of these Arduino boards to be implemented into more specialized designs which can't be fulfilled by these typical boards. It is in this way we will control all of the opening and closing mechanisms associated with each vent. These boards will be connected to the vents and also connected to the main MCU in order to provide a distributed design in which all of the boards are synchronized in their communication. This will also allow for temperature readings by the boards to be sent to the main hub.

Since the board is just a pure microprocessor, Arduino projects usually require extension boards to extend its functionality. This allows users to experiment with different components, to build different applications with just one board. On top of that, since there have been many members involved, it's very likely to find a well documented library for any components, with projects based on that. This makes it very easy to prototype and create embedded systems with the Arduino.

With these factors in mind, building a prototype for our application is straight forward. We can find previous Arduino-based IoT projects. We found that the most common

approach is star topology using the LoRa extension board. The board can be easily incorporated with an Arduino thanks to its library. Using this approach seems to be one of the least hassle, as everything is well documented by other makers and active support from community members is available.

There are things to consider, however. Even though Arduino may seem like the perfect choice for any application, it does come with trade back. Firstly, the board and IDE is pre-configured, making certain applications unsuitable. Take power consumption for example, the board is configured to run on 3.3V using an ineffective linear regulator; the crystal oscillator on board can also consume a lot of power. All of these factors make the board draws about 15mA without any component nor operation. Modifications, along with low power libraries, are required to make the board suitable for batteries-based application.

Secondly, Arduino is flexible to do all the tasks, but it's not optimized for 1 task. An analogy for this would be comparing Arduino with a Swiss army knife. It can perform any task, but it'll underperform when compared with tools that are built for a single purpose (screwdrivers, saw, knife, etc.). No one would like to use a bulky swiss army knife to take a screw off; they would rather use a cheap plastic screwdriver and finish the job quick and easy.

A good engineer practice is to use the correct tool for the job, that's why instead of using Arduino, we decided to look around for alternatives. We aimed to find an highly integrated MCU that's designed specifically for our application, an IoT network.

3.2.9.2 TI Launchpad & Simplelink platform

Texas Instruments has been around on the semiconductor market for a long time. With the rise of the Arduino, TI has adopted the concept of creating a user friendly development board. The first Launch pad was released in 2010, powered by the MSP430 family. Similarly, TI Launchpad is open-source, allowing anyone to access their software for free. The only difference is that TI Launchpad is aimed more at commercial use, so its UI and development platform isn't as friendly as Arduino.

TI development boards cover all of the hardware needed for the development of an embedded system; the most important being a flash emulator, allowing users to quickly program the board with a USB port. Additionally, The Launchpad also comes with extra MCU onboard to provide more advanced analysis, such as power tracking. This is why TI Launchpad is significantly more expensive than an Arduino board. While Arduino aimed to produce cheap, minimal boards that hobbyist can use in their project, TI aimed toward commercial use. Therefore, the Launchpad is to be used as an evaluation tool of the MCU, not to be used as part of a project.

Over time, TI introduces more evaluation boards, as well as more sophisticated libraries that makes their products more user friendly. The platform that we're interested in is the SimpleLink platform. SimpleLink platform created as a solution to wireless-connectivity

applications. The platform provides a myriad of products, each with built-in wireless capability. These wireless capabilities include Bluetooth, Zigbee, Thread, Wi-Fi, Sub-1Ghz and more. The platform offers software development kits (SDK) that can be imported to any MCU within the SimpleLink family.

Most importantly, TI provides resources and training for their platform, SimpleLink Academy. They offer free online courses for each of their SimpleLink products. Each course provides hands-on examples. Additionally, TI has online forums giving professionals answers from TI experts. These invaluable resources can help our group learn how to incorporate TI MCU into our prototype. However, since none of our group members has much experience with this platform, we must first overcome the learning curve to use these MCU.

3.2.9.3 SiliconLab Bluetooth mesh

SiliconLab is a rather new semiconductor manufacturer, when compared to TI, but they have played a big role in wireless MCU, most noticeably is the Wireless Gecko family. This family of MCU runs on ultra low power with built-in functionality for IoT systems. They provide multiple protocols, ranging from BLE, ZigBee, to Thread. Unfortunately, their evaluation kits are extremely expensive. Each kit contains more components than what we need, like an LCD screen. This makes each board more than what we could afford (at least \$100 per piece).

After some time browsing their products, we found that SiliconLab offers a more consumer friendly option, the Thunderboard Sense 2. This board has a lot of on-board sensors that allows quick testing of the features. However, Thunderboard MCU, which is EFRMG12, only supports BLE connection, so it really limits the application that we can use.

Interestingly, in SiliconLab's recent models, EFRMG12 can now perform bluetooth mesh protocol. Therefore, Thunderboard Sense can be a valid option for our application. However, upon looking for more documentation of their bluetooth mesh protocol, not much is written about the Thunderboard. All the document and recorded webinar is just about how to connect the board to your mobile app and to each other. There's not much information on how to develop our own applications. Therefore, in order to use this option, we need to do a significant amount of reverse engineering.

3.2.9.4 Comparison and conclusion

Each of the options provide different kinds of support. Arduino provides more community friendly support since there have been people simplifying the concepts, making it an easy option for anyone to start. Meanwhile, Texas Instrument provides a more paper-heavy, document-based support. To use TI MCU, we need to read a lot of their documents and follow their training. Lastly, SiliconLab provides example codes but

not many guides on how to use the product, so a lot of reverse engineering is needed. Because of the lack of documents, we've decided not to use SiliconLab's products.

Table 8: Arduino vs TI development Environment

	Arduino	Texas Instrument
Pros	<ul style="list-style-type: none"> ● Simplified programming interface, myriad of useful libraries. ● Massive community support 	<ul style="list-style-type: none"> ● Offer useful training to their products and question forums ● Uses low power technology ● Smaller PCB footprint thanks to built in wireless capability
Cons	<ul style="list-style-type: none"> ● The finished prototype is bulky, and costs a lot of PCB footprint due to external modules. ● Solutions is developed by homebrew makers, not experts 	<ul style="list-style-type: none"> ● Hard to digest documentations ● Limited libraries, only have libraries for basic functionality

At the point of this report, we have decided to do with the Arduino solution since it's the friendliest and cheapest method for students. We're interested in TI with their integrated ZigBee and Thread MCU as well, but without time to learn from their SimpleLink Academy, we aren't confident enough to consider this option. We have planned to test out TI SimpleLink platform once this semester is concluded.

3.3 Part Selection

The selection of parts is critical to the project as the parts we chose are what must meet the project specifications. Choosing parts is a balance of cost, specifications, and ease of integration. Choosing a part without proper research can cause issues further in development as they may end up causing the product unable to hit the targeted price point, fall behind on product capabilities, or become overly cumbersome to develop with.

We are attempting to undercut existing market competitors whilst providing equivalent or more features as such cost is an important factor in our product. Based on our requirements specifications our minimum technical specifications we do not need particularly high end components. This will allow us to achieve a low price point. However, there is a trade off with going with components that are too cheap even if they meet specifications as they may not be as simply integrated into the system we are building. Given this, ease of integration is also a high priority for us as we need to have a quick turnaround on developing our product.

3.3.1 MCU

The MCU is the most important part of the IoT devices. When choosing an MCU for IoT applications we want to choose the MCU Choosing the MCU has several implications

most importantly choosing an MCU determines the development environment and the available libraries. The primary considerations for an MCU in our system are its cost, power consumption, and peripheral compatibility. Our processing requirements are quite low as the complexity of the system is relatively low. The duty of most of the devices will be to interface with a sensor or motor and a radio communication module - with the exception of the central hub.

"In many SOC applications, the smallest possible microcontroller is all that's needed to coordinate the various tasks being performed on the chip,"

- Chris Rowen, Tensilica president and CEO

3.3.1.1 Microchip megaAVR Family - Arduino - ATmega4809

The ATmega chips have seen a large amount of popularity due to the rise in popularity of the Arduino development platform. Arduino is definitely the most popular development environment particularly in the hobbyist space. As a result there are many open source official and community libraries making interfacing with popular peripherals quick to implement. Additionally as a result of this popularity there is also plenty of documentation and sample projects. In terms of technical specifications of the chip the processor is approximately in the middle of the stack of the MCUs we are considering whilst being the cheapest. The primary concern with this chip is its simplicity and there is a possibility that it may not be able to fulfill our computing needs, given the relative simplicity of our system this will likely not be an issue.

3.3.1.2 Microchip SAMD Family - Arduino - SAMD21

The SAMD21 chips inherit most of the benefits of the aforementioned ATmega4809 while being faster and more power efficient. More importantly this MCU has many more configurations than really any of the other MCUs mentioned. In the specifications table below a mid range MCU is listed, but there are other configurations that offer more or less pins and different program memory capacities. This is useful as we can use the same development environment and choose the MCU that best fits our needs for each separate device and optimize cost.

3.3.1.3 TI SimpleLink Family - Launchpad - CC2652RB

Texas instruments is a large player in the microprocessor and embedded systems processor space. CC2652 chip is one of the most feature rich out of the proposed MCUs however this comes at a cost as it is the most expensive out of all of the options. The CC2652RB has built-in, multiprotocol support for ZigBee, Open Thread and BLE, as well as a brand new crystals-less BAW clock, allowing accurate timing while consuming less energy. This will simplify the PCB design as we will not need a separate peripheral for wireless connections nor an external clock. Texas Instruments offer a Launchpad, a development board, allowing users to test the chip functions.

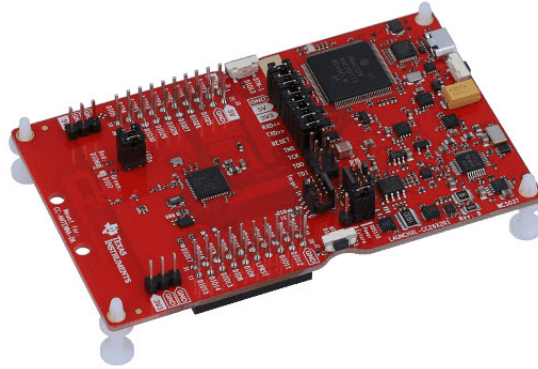


Figure 43: MSP430

(Reprinted with permission from Texas Instruments)

The Launchpad is good for testing out the features of the microprocessor, as it allows quick access to the CC2652RB, saving time and effort on making the PCB. The Launchpad does include a lot more IC than a typical application, making the footprint larger. These extra ICs allow users to evaluate power consumption, operation, etc. The finished product will only require CC2652RB and its passive element.

Texas Instruments provides guides and documentation on Zigbee and Open Thread SDK on their website, as well as useful libraries for the microcontroller. This option is suitable if we choose to go with the ZigBee or Open Thread solution. As CC2652RB supports multi protocols, it allows us to conveniently test and evaluate both ZigBee and Open Thread protocols.

3.3.1.4 SiliconLabs Gecko Family - ThunderBoard - EFR32BG22

EFR32BG22 is a low power microcontroller that solely integrates BLE. However, it's half the cost of CC2652RB, making it a viable option as it also doesn't require an external wireless peripheral. In their more recent models, EFR32BG22 is upgraded and now supports bluetooth mesh, making it a candidate for bluetooth mesh protocols.

Unfortunately, SiliconLabs does not provide a lot of consumer friendly evaluation boards, each costs more than \$100. The only board that we found that's affordable is the Thunderboard Sense 2, which includes extra sensors on board. This makes Thunderboard Sense 2 a suitable board for testing the bluetooth mesh approach. However, this approach will require a complete redesign of the project, as it doesn't require a gateway device. A possible solution is to connect an EFR32BG22 IC to the internet through WiFi, but this can create a lot of challenges due to the lack of documentation.

3.3.1.5 Espressif ESP Family - NodeMCU - ESP8266

The ESP family of SOCs have seen a growing interest in the same field as Arduino has, particularly due to the NodeMCU project. NodeMCU is an open source firmware project that allows for arduino-like programming but on an SOC that had wireless capabilities built in. However due to NodeMCU being relatively new in comparison than arduino this

means that not all drivers that are available for arduino have NodeMCU alternatives. Additionally despite being slightly more expensive than the ATmega4809 it is missing other significant features such as on board ROM and GPIO pins, many of the already few GPIO pins are reserved for wireless functions.

Table 9: MCU Specification Comparison

Feature	ATMega4809	ATSAMD21	CC2652R	EFR32BG22	ESP8266
Dev Envir	Arduino Atmel studio	Arduino Atmel studio	LaunchPad	Thunder Board	NodeMCU
Voltage	1.8V ~ 5.5V	1.6V ~ 3.6V	1.8V ~ 3.8V	1.7V ~ 3.8V	2.3V ~ 3.6V
Sleep Curr	8 μ A	2.70 μ A	1 μ A	1.4 μ A	20 μ A
Active Curr	11.4mA	6.32mA	3.4mA	8.2mA	15mA
Clock Speed	32 kHz - 20 MHz	32 kHz - 48 MHz	32 kHz - 48 MHz	32 kHz - 76.8 MHz	24 MHz - 52 MHz
RAM	6 KB	32 kB	80 KB	32 kb	50 kB
Prog Mem	48 kB	256 kB	352 kB	512 kB	External.
ADC Input	16	14	7 (Shared)	1	1
Digital I/O	18	26	31	17	15
GPIO Pins	41	38	31	18	16
CPU Arch	AVR	ARM	ARM	ARM	Xtensa
Instr Width	8-bit	32-bit	32-bit	32-bit	32-bit
Mfr	Microchip	Microchip	Texas Instruments	Silicon Labs	Espressif Systems
Size	6mm x 6mm	7mm x 7mm	7mm x 7mm	4mm x 4mm	5mm x 5mm
Chip Price	\$1.59	\$3.13	\$6.12	\$3.28	\$1.60



Figure 44: ATSAM21G18A-U Microprocessor

Looking at the above table any of the above MCUs would satisfy our requirements. The ESP8266 is an admirable choice but not in our system as it not only uses the most power out of all of the MCUs but it will likely not fulfil our GPIO needs.. The CC2652R is a very convincing option as it is fast, and more power efficient, and provides some form of internal radio controllers. However these benefits cannot be rationalized by the drawbacks in price and development hurdles.

The ATmega4809 is likely to be a good candidate as it is very inexpensive and will likely satisfy most of our needs. One of the concerns would be its Program memory as it can easily fill up due to any image data needed for a screen. A more concerning issue is that the ATmega4809 is a 5V logic level chip where all the others are 3.3V. Many peripherals are 3.3V and are not 5V compliant meaning we would need to complicate our board design with many logic level Converters.

Finally narrowing down to the EFR32BG22 and ATSAM21 the SAMD21 is likely to be a better option as it is more configurable so we could get significantly less expensive MCUs for the peripheral devices where a built in WiFi radio is not necessary nor the larger program memory. The larger issue with the EFR32BG22 is the amount of GPIO pins in the central hub we could possibly run out of Pins to assign to functions. As such the ATSAM21 series is the best option for us as it is more configurable, easy to program for, has many gpio pins, cheap, and has a low current draw.

3.3.2 Wi-Fi Transceiver

The purpose of IoT systems is tight integration with internet services. When it comes to wireless internet communication the only option is Wi-Fi. The primary issue with Wi-Fi capabilities is their power draw. This means that any device that runs off of a small battery should avoid using Wi-Fi as it is a power hungry method of transmitting data as it's focus is on data rate rather than power usage. Given this in order to satisfy requirement 1.5 only the central hub will have Wi-Fi capabilities and peripheral devices will communicate back to the central hub through more power efficient means.

These wifi cards typically are inserted into computing systems or distributed printed circuits boards. These are also accompanied by antennas in order to capture the signals from other wifi compatible devices. In our implementation we will require a wifi module

in order to connect the IoT system to the internet and allow for mobile phones to connect and therefore allow for remote access to the controls of the heating and cooling of the system.

3.3.2.1 ESP32-WROOM

This module is a Wi-Fi transceiver that is based on the ESP32 chip. The primary benefit of this module is that it provides very good power output for the price. Good power output and sensitivity is important because we need our IoT device to be able to communicate with the router of a home in a reasonably sized home. The primary attractive feature of this module is its price as it is very inexpensive.

3.3.2.2 NINA-W102

This module is also based on the popular ESP32 chip. The primary factor that brings this module into consideration about this module is that it is the same module that is integrated on the Arduino development board. As such using the same module in production would be beneficial as it will simplify the development process.

Table 10: Wi-Fi Module Specification Comparison

Feature	NINA-W102	ESP32-WROOM
Voltage	2.7V - 3.6V	2.7V - 3.6V
RX Current	120mA - 140mA	118mA - 118mA
TX Current	225mA - 320mA	165mA - 211
Power Output	16 dBm	20 dBm
Sensitivity	-96 dBm	-98 dBm
Max Data Rate	54Mbps	150Mbps
Frequency	2.5 GHz	2.5 GHz
Interface	SPI, UART	UART
Antenna	PCB	I-PEX connector
Manufacturer	U-Blox	Espressif Systems
Size	10mm x 14mm	18mm x 19.2mm
Cost	\$7.28	\$4.20



Figure 45: ESP32-WROOM-32U WiFi Transceiver

Looking at the above specifications the choice is not as clear as it may seem. Though the ESP32-WROOM is cheaper it would complicate the development process as our development platform would be significantly different over the prototype. Additionally the NINA-W102 has an integrated antenna over requiring an external antenna which is a hidden cost of implementing ESP32-WROOM over the NINA-W102. From our testing so far the transmission range of the NINA-W102 is sufficient and modifying the firmware of the ESP32-WROOM is an endeavor that may prove to be fruitless.

3.3.3 ISM Band Transceiver

The connectedness and ease of communication between devices and the internet is the primary pillar of IoT. As such communication is one of the most important features of an IoT system. General ISM is the license free radio band that is reserved for industrial, scientific and medical purposes (868 MHz in Europe and 915 MHz in North America and Australia). As a result this means there is no cost to use this radio band. As a result of being a lower frequency than Wi-Fi it is able to have a much longer range even though its transmission power is much lower than Wi-Fi. The primary requirements of concern are requirements 1.5 and 1.6. To satisfy these requirements we need a wireless protocol that uses a small amount of power and has enough range to transmit a reliable signal from anywhere in a regular home.

3.3.3.1 HC-12

The HC-12 module is a wireless transceiver module that communicates over the 433.4 MHz section of the ISM band. This module is the cheapest out of the options and is simple to interface and operate. This module communicates over UART which means that integrating it will not consume many of the digital communication pins we have. The communication model of this transceiver acts as an open data channel and does not natively support any sort of built in encryption model.

3.3.3.2 LoRa - RFM95W

LoRa is a relatively new protocol that utilizes the same ISM sub 1GHz license free radio bands. What is unique about RFM95W and other LoRa transceivers is that it allows for low bitrate data to be transmitted for very long distances despite the low transmission power. LoRa achieves this through a patented modulation technique that allows for its long transmission range. As a result these modules are more expensive as they have the cost of the patented LoRa technology built into it. Refer to the specific section concerning LoRa for more details on the modulation technique and specifications (Section: 4.2.6.5).

3.3.3.3 RFM69HCW

The RFM69HCW communication module is also able to enjoy a relatively long range at such a low transmission power because its data rate is quite low. Given the application this is not an issue at all. What is beneficial about the RFM69HCW is that it supports encryption and assigning individual nodes and network IDs. These radio modules are optimal for wireless communication of simple data between embedded hardware.

Table 11: Wireless Communication Specification Comparison

Feature	HC-12	RFM95W	RFM69HCW
Voltage	3.2V - 5.5V	3.3V	1.8-3.3V
Sleep Current	80 μ A	0.2 μ A	0.1 μ A
Idle Current	3.6 mA	1.5 μ A	1.2 μ A
RX Current	16 mA	10mA - 12mA	16 mA
TX Current	100mA	20mA - 120mA	16mA - 130mA
Max Data Rate	236kbps	300kbps	300kbps
Line of Sight Range	1 km	1 km - 10 km	500 m - 1.45 km
Frequency	433.4MHz	915MHz	915MHz
Interface	UART	SPI	SPI
Max Nodes	Open Channel	255	255
Manufacturer	Seeed Tech	RF Solutions	SparkFun Elect
Size	27mm x 13mm	16mm x 16mm	16mm x 16mm
Cost	\$3.40	\$13.57	\$5.95

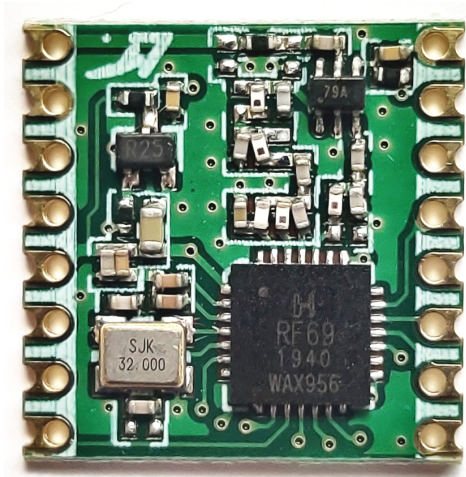


Figure 46: RFM69HCW Radio Transceiver

Out of the above options the RFM95W is the highest end as it features the patented LoRa modulation technique. However, this comes with a significant cost that cannot be rationalized particularly because it would be overkill for the application of sending signals across a home. The HC-12 seems like a decent choice as well but it's communication protocol does not natively implement network and node IDs or encryption. RFM69HCW is the best option as it retains the more advanced features and network protocol but does not utilize the actual LoRa modulation technique in order to save costs.

3.3.4 IoT Cloud Services

In order to send and receive data between the mobile application and the central thermostat hub, we require an IoT cloud service to act as a connector to handle our requests. There are many notable IoT platforms we can use that will fit our needs. We have narrowed our choices down to three platforms: Amazon Web Services (AWS) IoT, Microsoft Azure IoT, and Google Cloud IoT. All three options have a core solution with basic functionality and a set of additional modules you can add when necessary. In addition to IoT functionality, these platforms will provide us with the database that the user's data will be stored in. The platforms will effectively be the host that our mobile application will send requests to for user data. This section will highlight the key features of each platform in more detail.

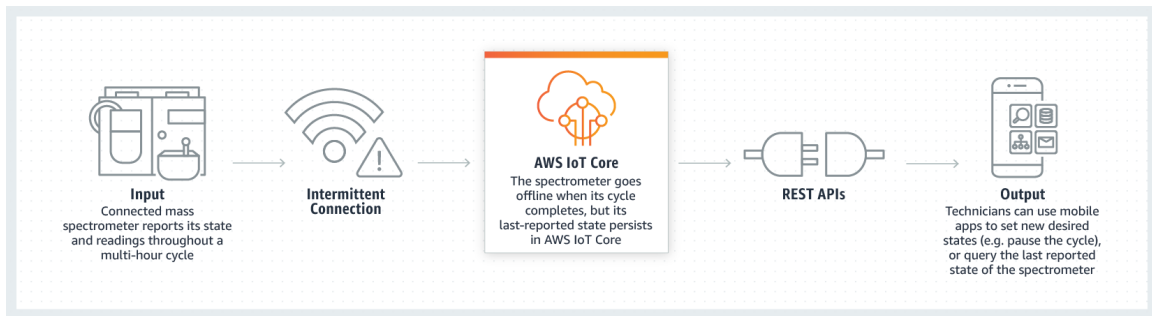
3.3.4.1 Amazon Web Services

AWS provides many IoT options. The most notable option, and the platform that AWS puts out the most for, is AWS IoT Core. AWS IoT Core provides and manages device authentication, connections, and communication between the different devices and different services. The entry point of the platform is called Device Gateway, which supports many communication protocols including MQTT, HTTP, and WebSocket. AWS states that the platform is capable of handling over a billion devices and assigning each

one a unique identity. Each point of connection requires authentication and contains encryption of data so the devices connected over IoT core never exchange unverified data.

The Rules Engine, which is the platform's function processing engine, is where vetted messages are sent to and encrypted. It then routes them either to a device or cloud AWS service. Some of the few cloud services they have are AWS Lambda (a serverless computing platform), Amazon Kinesis (a solution for processing big data in real time), and Amazon S3 (a storage service).

AWS's Device Shadow, another useful feature of IoT Core, stores the state of every device that it is either currently in or is desired to be in. So if the device connected through IoT Core is offline or busy, cloud applications are still able to manage the device and send commands to it. As soon as the device is back online, it synchronizes its final state with updates.



*Figure 47: AWS IoT Core Overview
(Reprinted With Permission From Amazon Web Services)*

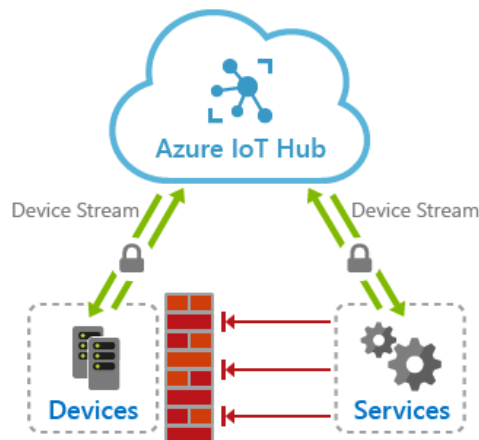
AWS IoT Core is a very popular choice among developers, for a reason. It offers SDKs for Android and iOS applications as well as for Embedded C, C++, JavaScript, and Python. AWS makes the development process quicker by providing a large variety of templates along with IoT Things Graph, a visual drag-and-drop tool that simplifies the workflow process across IoT devices.

When it comes to sending and receiving data in an IoT system, it is beneficial to have a Real-Time application that can get data quickly. AWS claims to provide microsecond latency and high throughput to support millions of requests per second. Their AWS database services for real-time applications are Amazon ElastiCache and Amazon DynamoDB. ElastiCache allows developers to seamlessly deploy, run, and scale popular open source compatible in-memory data stores. This is used for things like caching, real-time analytics, pub/sub apps, etc. DynamoDB on the other hand is a key-value database that Amazon claims to provide single-digit millisecond performance at any scale. This is used more suitable for personalized mobile applications, which is the option we would most likely utilize if AWS is chosen.

3.3.4.2 Microsoft Azure

Microsoft's Azure IoT Hub is the heart of Azure's IoT suite, being their primary PaaS (platform-as-a-service) product. IoT Hub handles connectivity, management and communication between all devices. The platform provides two tiers, basic and standard, each having different features that are supported. The basic tier gives services like messaging from device-to-cloud, authentication between devices, and support for communication protocols like HTTP, MQTT, and AMQP.

Another notable Microsoft Azure IoT service is IoT Central, which is their scalable SaaS (software-as-a-service) IoT platform. It offers IoT software that is rapidly designed with security features that are ready outside of the box. IoT central comes with integrated device monitoring and management functions to connect, reconfigure, and update devices. The module includes numerous application templates for different industries to accelerate development speed. If combined with Azure IoT Hub, it enables building more complex apps, capable of supporting millions of devices.

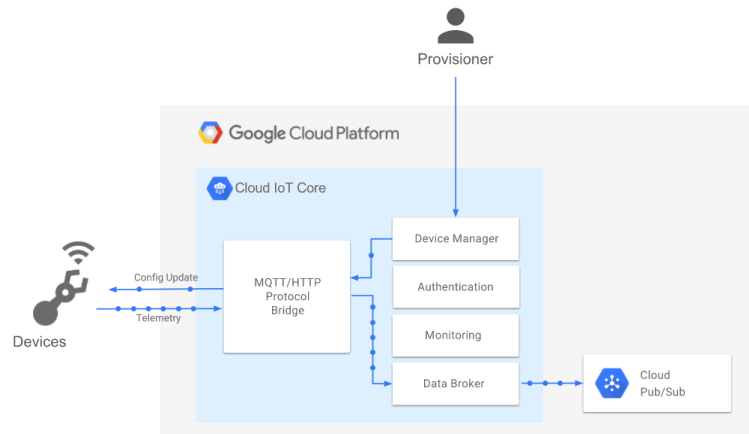


*Figure 48: Azure IoT Hub Device Streams
(Reprinted with permission from Azure Cloud)*

As for the storage of user and device metadata, Azure also has a variety of architectures we can use. This variety includes Azure Blobs, Azure Files, Azure Queues, Azure Tables, and Azure Disks. The one that fits our specific needs to store user data or any device metadata would be Azure Tables which allows you to store structured NoSQL data in the cloud, providing a key/attribute store with a schemaless design. It is used when you want to store flexible datasets like user data for web applications, address books, device information, or other types of metadata your service requires. While the other architectures are notable, like Azure Blobs which allows unstructured data to be stored and accessed at a massive scale in block blobs, they do not fit our needs as they are used for files and media random access. If Microsoft Azure were to be used, Azure Tables would be the storage architecture of choice.

3.3.4.3 Google Cloud

The heart of Google's IoT suite is its top-tier product Cloud IoT Core, which they claim is powerful enough to manage data from millions of devices. Google IoT Core provides two architectures: Device Manager and Protocol Bridge.



*Figure 49: Google Cloud IoT Core
(Reprinted with Permission from Google Cloud)*

Device Manager provides you with a service to monitor device health and activity, update device configurations, and manage credentials and authentication. Protocol Bridge, using MQTT and HTTP formats, is responsible for connectivity. MQTT is an industry-standard IoT protocol that stands for Message Queue Telemetry Transport. It publishes data streams to a Cloud Publish/Subscribe service, which makes merging messages from different sources into a single system possible. From the Cloud Pub/Sub, data is then forwarded to other Google cloud services.

Google is also very well known when it comes to their storage architectures as well, most notably Google's Firebase. They provide a real-time database that is cloud-hosted. Data is stored as JSON and synchronized in real-time to every connected client. Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Google claims to provide collaborative and immersive experiences without thinking about networking code.

3.3.4.4 *Arduino IoT Cloud*

Arduino IoT Cloud is a powerful service, allowing anyone to create IoT applications with just a few simple steps. With a combination of smart technology, user-friendly interfaces and powerful features, Arduino IoT Cloud makes it really easy to manage your IoT device, especially if it is an Arduino device.

Table 12: IoT Cloud Services Comparison

Platform	Communication protocols	Key offering and its main functions	Pricing (Basic Tiers)
Amazon Web Services	HTTP MQTT WebSockets	AWS IoT Core: <ul style="list-style-type: none"> • Connectivity • Authentication • Rules engine • Development environment 	2,250,000 minutes of connection free, \$0.08 (per million minutes of connection) after that
Microsoft Azure	HTTP MQTT AMQP over WebSockets	Azure IoT Hub: <ul style="list-style-type: none"> • Connectivity • Authentication • Device monitoring • Device management • IoT Edge 	8,000 messages/day free, \$10/month for 400,000 messages/day
Google Cloud	HTTP MQTT	Google Cloud IoT Core: <ul style="list-style-type: none"> • Connectivity • Device management 	Up to 250 MB data volume/month free, \$0.0045/MB after that
Arduino	HTTP WebSockets	Arduino IoT Cloud: <ul style="list-style-type: none"> • Connectivity • Authentication • Device monitoring • Device management 	Up to 1 thing with unlimited messages/day for free.



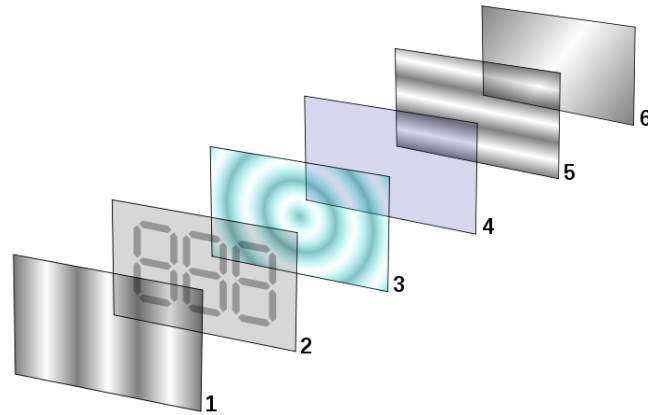
*Figure 50: Arduino IoT Cloud
(Reprinted with Permission from Arduino)*

All four IoT Cloud platforms from above would enable us to have a functional IoT system that allows communication between devices and a mobile phone, at least over HTTP or MQTT. However, it would also be a great addition if the WebSockets communication protocol was available for us to use, which Google Cloud does not provide. WebSockets allows each device/user to both send and receive messages in real-time. This would come in handy for the user to perform speedy requests to the central hub. That leaves Amazon Web Services (AWS), Microsoft Azure, and Arduino IoT Cloud. Although Azure does have a way to use WebSockets, it requires another layer which is to use AMQP over WebSockets. While the AMQP does add message orientation, reliability, and security, it will serve as an unnecessary layer for our purposes.

The only data that we need to send between devices are temperature details and temperature change requests. This does not require more advanced routing and security, especially if using AMQP will make the communication more complicated if we would like to use WebSockets. That leaves AWS and Arduino IoT Cloud. Since we are using Arduino devices for our system, Arduino IoT Cloud would provide a very easy to use platform that has all functionality we would need already built-in. In addition, they provide out of the box front end customizations that we can then export into a mobile application. Therefore, Arduino IoT Cloud proves to be the most solid IoT platform for our needs and is the winning candidate for us.

3.3.5 Display

The display is important to convey information to the user. Although we will have a mobile app that will be able to fully control the system it is still useful to be able to convey system information to the user without the use of the app. In our system there are three separate hardware components - the central hub, distributed thermostats, and the distributed vent dampers. Each of these components have separate needs and will be compared separately.



*Figure 51: LCD Display Construction, 1. Polarizing Filter 2. Glass Substrate 3. Twisted nematic liquid crystal 4. Glass Substrate 5. Polarizing Filter 6. Backlight
(Creative Commons: Ed g2s)*

These LCD displays could be very useful in debugging and correcting errors with temperature sensors as well as other components in the system. They also provide an easy way to measure the real time temperatures associated with each room in the model. These displays can be connected directly to the distributed boards which are spread throughout the rooms.

These LCDs can be relatively easily programmed to display the relative via connecting them to the temperature sensing modules of the boards. This would also allow for verification via comparing the results collected in the web application to the results of the LCD display. The figure above showcases a typical LCD display and it is of the seven segment variety.



*Figure 52: Casio Watch LCD Display
(Creative Commons: Ricce)*

https://commons.wikimedia.org/wiki/File:Casio_W-59_digital_watch.jpg

Figure 51.1 showcases an LCD display used in a Casio digital wristwatch. This display found in this Casio watch is similar to what we plan on incorporating into our IoT

system. This display utilizes a seven segment system in order to represent the digits 0 through 9. This is ideal for displaying the current time as all that is required is simply digits. We encounter a similar situation when it comes to temperature sensing modules we plan to use in the smart air conditioning IoT system. They will allow for an easily programmable display which will display the temperature at each node in the system.

Once again, these types of displays are a good fit for this application as they. These displays are ideal for IoT systems as they consume relatively little power and also have seven segment displays. These displays are very widely used and as Electrical/Computer engineers who have taken embedded systems. This class is required for ECE students and therefore the members of the group will be able to easily transition into programming these displays to work such that they display the temperatures from the sensors on the distributed pcbs.

Given our group's experience regarding embedded systems we should be able to program the displays relatively easily. However, the experience we have programming seven segment displays are with the Texas Instruments MSP430 board in which the interface may be different than the planned Atmega chip we will use throughout the system.

3.3.5.1 Character LCD - ACM1602K

Character LCDs are a type of LCD that are able to show a predefined set of characters to the display. The character LCD is a reasonable type of display when simple text data is needed to be displayed. The primary issue with this type of display is that its flexibility to display information is quite limited and simultaneously draws a lot of power. It's primary benefit is that it is relatively cheap.

3.3.5.2 Color LCD - Waveshare 2.4" LCD

LCD displays are able to be controlled at a pixel by pixel level rather than indexing rows and columns like a character LCD. These types of displays are able to show great detail in many colors. Due to the popularity of LCDs in modern technology small screens such as this one are able to be high quality and relatively cheap. This particular display module has a built in controller so it is able to be controlled directly through SPI so it is easy to control.

3.3.5.3 E-Paper Display - E2154FS091

E-Paper displays are a relatively new type of display that electrically controls pigments in a substrate and does not emit any light unlike an LCD or an OLED. Another special characteristic of E-Paper displays is that they retain the image on the screen even without power. Given this these types of displays excel in areas where low power usage is required and only mostly static images need to be shown.

3.3.5.4 No Display

Finally a display option is to not have a display at all. Displays complicate the design and not every application requires one. Not using a display allows for a simpler MCU to be

used as well as reduce power usage. Many IoT devices do not utilize displays because many times it is not necessary and would complicate and interfere with the design of small low power devices IoT calls for.

This would improve the cost of the system and lead to a final IoT system with a lower power draw. However, the consequences to making this decision is it may make testing and other features less convenient. Also, given the nature of our design and the fact that many IoT systems don't use displays it may be beneficial to go with this approach.

Table 13: Display Specification Comparison

Feature	ACM1602K	Waveshare 2.4" LCD	E2154FS091	No Display
Voltage	4.75V - 5.25V	2.5 V - 3.3 V	2.3 V - 3.6 V	N/A
Current Draw	160 mA	160 mA	5 μ A - 8mA	N/A
Interface	Custom	SPI	SPI	N/A
Feature	ACM1602K	Waveshare 2.4" LCD	E2154FS091	No Display
Resolution	16 col x 2 row	240 x 340	200 x 200	N/A
Colors	2	65K	3	N/A
Manufacturer	AZ Displays	Waveshare	Pervasive Disp	N/A
Size	55.73mm x 10.98 mm	2.4" (Diagonal)	1.54" (Diagonal)	N/A
Cost	\$4.76	\$9.99	\$6.80	\$0



Figure 53: Waveshare 2.4" LCD

(Reprinted with Permission from Waveshare)

The central hub will replace the traditional thermostat in the HVAC system and will be where any controls outside of the app will be performed. The display we need does not need to be low power as the central hub will not run off of battery power. Additionally it may need to display a large amount of information. Given this, the 2.4" LCD is the best solution as it can display the most information and is relatively easy to interface with.

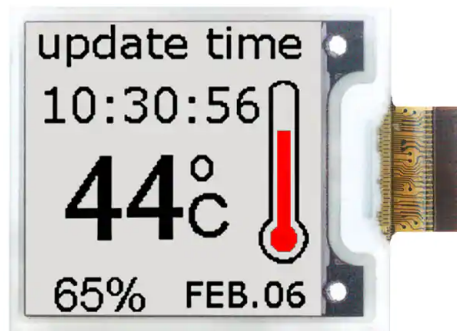


Figure 54: E2154FS091 E-Paper Display

(Reprinted with Permission from Waveshare)

When it comes to the distributed devices Neither types of LCDs are viable as they constantly consume a lot of power. This leaves the two low power options of the E2154FS091 and no display. Particularly for the vent damper it is entirely unnecessary for it to have any sort of display. However an E-Paper display would be quite beneficial to show the current temperature reading. However due to E-Paper displays being a relatively new technology these displays are relatively expensive - particularly considering their small size.

3.3.6 Temperature Sensor

For our system we need to be able to measure the ambient temperature of the rooms in the home so that it may be reported back to the central hub. Our use case does not require the ability to measure extreme temperatures. Humans are able to detect small changes in

room temperature to comply with requirement 3.2 the temperature sensor must be accurate to within 1°C so that when a desired temperature is set the system will be able to accurately hit the target temperature.

3.3.6.1 Thermocouple

Thermocouples are excellent because they are very accurate and self powered. One of the issues is that a thermocouple requires an amplifier which will increase complexity and cost - on top of being a more expensive type of sensor. More importantly thermocouples do not typically come in a package that is designed to be mounted to a PCB - be it through hole or SMD. Given all of these drawbacks a thermocouple cannot be our temperature sensing method.

3.3.6.2 Low Voltage Analog Temperature Sensor - TMP235A4DCKR

This sensor is a cheap analog sensor that has a very low power usage. It is also SMD which means that it is able to be integrated on an final development board without occupying a significant area on the final PCB. Using an analog sensor means that we will utilize one of the few analog input pins however this is likely not going to be an issue as we do not have many analog peripheral devices. Using an analog sensor means that we have a sensor that is less expensive and keeps cost low as there is not an I2C or a OneWire bus that we need to communicate over.

3.3.6.3 Low Voltage Digital Temperature Sensor - MAX31820

The digital sensor is a decent balance between cost and complexity as it is still relatively inexpensive and provides a greater temperature accuracy over the analog device. This specific temperature sensor communicates over a one wire bus. This is beneficial as it minimizes the use of GPIO pins as it only requires a single wire.

Table 14: Temperature Sensor Specification Comparison

Feature	Thermocouple	TMP235A4DCKR	MAX31820
Type	Analog	Analog	Digital
Voltage	N/A	2.3 V - 5.5 V	3.0 V - 3.7 V
Current Draw	Self-powered	9 μ A	0.75 μ A - 1 mA
Temp Range	-270 °C to 1372 °C	-40 °C to 125 °C	-55 °C to 125 °C
Accuracy	$\pm 0.1^\circ\text{C}$	$\pm 1^\circ\text{C}$	$\pm 0.5^\circ\text{C}$
Manufacturer	SparkFun Electronics	Texas Instruments	Maxim Integrated

Size	2mm x 2mm x 3mm	2.15mm x 1.4mm x 1.1mm	4.95mm x 3.94mm x 2.92mm
Cost	\$13.95	\$0.41	\$1.95



*Figure 55: TMP235A4DCKR Analog Temperature Sensor
(Reprinted with Permission from Texas Instruments)*

Given the above specifications TMP235A4DCKR looks like the best option for our usage. Our primary selection factors are price and power usage. Given that there isn't a clear reason as to why to spend 4x the price on a more advanced digital temperature sensor when an analog sensor would do the same functionality for cheaper. Additionally the TMP235A4DCKR is tolerant to many voltages and uses a similar amount of current to the MAX31820 when it is idle. As such the TMP235A4DCKR is the best option for our system.

3.3.7 Motor

For our system we need some way to physically manipulate the vent dampers. In our prototype system we do not need high torque motors to move the valves to open or close the vents. As such this means that the primary deciding factor will be cost and power usage.

3.3.7.1 Stepper Motor - 28BYJ-48

Stepper motors are a type of motor that are controlled by sending step and directional signals to the motor rather than applying a straight DC to two terminals. This means that this type of motor requires a motor controller. One of the benefits of a stepper motor is its torque and holding torque are typically quite high for their size.

3.3.7.2 Linear Actuator

Linear actuators are a type of motor that provide a linear driving motion rather than a rotational one. Linear actuators are typically easy to drive but due to their greater mechanical complexity they tend to be more expensive. In this case it is prohibitively expensive due to being much smaller than a typical linear actuator.

3.3.7.3 Servo Motor - SG90

Servo motors are a type of motor that have an integrated motor controller and can be easily interfaced with. Servo motors come in all sizes but particularly the SG90 is especially small as a result it's torque is not very high.

Table 15: Motor Specification Comparison

Feature	28BYJ-48	Linear Actuator	SG90
Voltage	3V - 5.5V	6 V	4.8 V - 7.2 V
Idle Current	5 mA	7.2 mA	2.5 mA
Running Current	200 mA	300 mA	220 ± 50 mA
Stall Current	700 mA	460 mA	650 ± 80 mA
Manufacturer	Adafruit Industries	Actuonix	Adafruit Industries
Size	35x22x19 mm	52x18x 14.9 mm	23x11x29 mm
Price	\$4.95	\$70	\$5.95

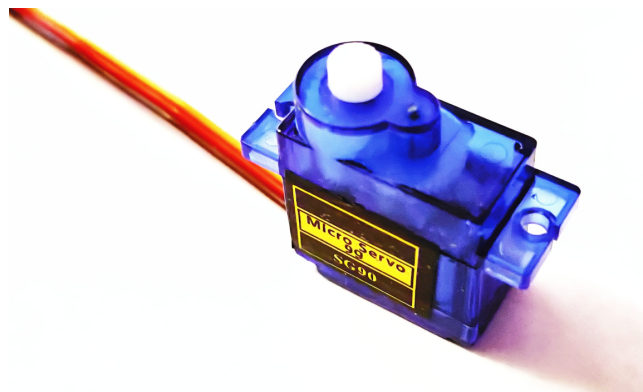


Figure 56: SG90 Micro Servo

For our use case the linear actuator is clearly not the option we are looking for as it is very expensive. Between the 28BYJ-48 and the SG90 the SG90 is likely the better option as it is not only smaller but will be less complex to implement as it does not require a motor controller. This lesser complexity means that the servo motor implementation could end up being cheaper to implement because it does not need an external motor controller. Additionally the servo motor is simply easier to integrate to the system.

3.3.8 Power

With every electronics project there is the requirement of some form of power delivery. Most IoT devices run off of some form of battery and particularly due to the very low

power consumption of IoT devices this means that batteries can last for months or even the entire lifespan of the device. In our system charging/replacing of batteries is the primary maintenance action. Requirement 1.5 specifies that the system only need maintenance every month which means that we should have a sufficiently large battery but also without going over budget. Because our devices are not required to be very small or portable the primary constraint on part selection is price.

3.3.8.1 *Wired Power*

Wired power is a good option considering that the distributed devices should remain stationary. Wired power is optimal because the system requires virtually no maintenance in the form of recharging or replacing batteries. However, the issue that arises is if the user needs to instal the device somewhere far away from the nearest outlet, such as a ceiling vent, wired power becomes impractical.

3.3.8.2 *Alkaline battery*

Disposable batteries are very convenient as they provide great power density at a very low cost. Many low powered devices are able to be powered off of a disposable alkaline battery for many months, meaning that it easily satisfies requirement 1.5. One of the primary issues with disposable batteries is the loss in customer satisfaction. Often products that require disposable batteries are perceived as cheaper and lower quality. And unlike rechargeable batteries are completely unable to be run off of wired power.

3.3.8.3 *Li-ion Battery*

Li-ion Batteries are one of the most common batteries used in small electronics as they are relatively cheap, provide good power density, and come in a variety of shapes and capacities. One of the downsides of having a rechargeable battery in general is the need to have the power regulation circuit also be able to charge the battery which will increase the cost slightly. However, the primary advantage the rechargeable battery has over either of the previous options is the ability to run off of wall power or off of battery. This allows for the flexibility of mounting peripherals without regard to outlet positions but also allows for a lower maintenance install.

Table 16: Power Source Specification Comparison

Feature	LR6C (2C AA)	Li-ion	Wired
Voltage	3 v	2.8v - 4.2v (Nom 3.7 v)	5 v
Max Discharge	1 A	400mA	2 A
Capacity	2700 mAh	400-1200 mAh	N/A
Manufacturer	Fujitsu	SparkFun	FONKEN

		Electronics	
Size	14.5mm x 50.5mm	26.5mm x 36.9mm x 5.0mm	42.9 mm x 32.0mm x 22.1mm
Price	\$2.70	\$4.95 - \$9.95	\$3.99



Figure 57: 400 mAh Li-ion battery
(Reprinted with Permission from DigiKey)

The power source for the central hub will be the power supplied by the HVAC wiring as it will replace the standard thermostat. Using the power supplied by the wall does not require an external USB power source and will reduce cost significantly meaning that also it. For the distributed thermometer a Li-ion battery is likely best as it allows for the flexibility of not being near an outlet but is capable of running off of a USB power supply. As for the distributed peripherals given the above specifications we believe the extra cost is worth the increase in customer satisfaction.

3.3.9 DC Power Regulators

The power regulator is a very important part of the product as it must regulate the voltage for every component in the product. The Voltage regulators ultimately tie together the entire product. As such its requirements are based upon the requirements of all of the devices it is powering. In our system we have 3 different types of devices and each of them have different electrical needs based on the components it will be powering. In order for reliability we want to budget for extra power than what we actually need in order to increase reliability. Additionally a when it comes to switching power supplies Higher switching frequency means t

3.3.9.1 12v to 3.3v Buck Regulator

The central hub will be powered off of the 12v that the AC system provides. As such it must be able to convert that voltage down to 3.3v. This device will be powering the MCU, Display, WiFi, and Radios. This means that this device needs to have a relatively high output current. Because this device is not powered off of a battery efficiency is less of a priority.

Requirements:

- Input Voltage: 10-14 V
- Output Voltage: 3.3 V
- Output Current: 500 mA
 - MCU: ~ 10 mA
 - WiFi: ~ 150 mA
 - Radio: ~ 150 mA
 - Display: ~ 150 mA
 - 10% Headroom: ~50 mA

3.3.9.1.1 TPS62160DGKR

This buck switching regulator is a decent option because it is relatively inexpensive and is able to provide more than the current needed at the voltage required. However, because this is not a fully integrated module it will require external inductors and other components in order to function.

3.3.9.1.2 TPS82150SILR

This regulator is very much like the TPS62160DGKR but is a fully featured module with an integrated inductor. As a result it's footprint is larger but also means that it overall reduces the part count which means it makes integration and assembly easier.

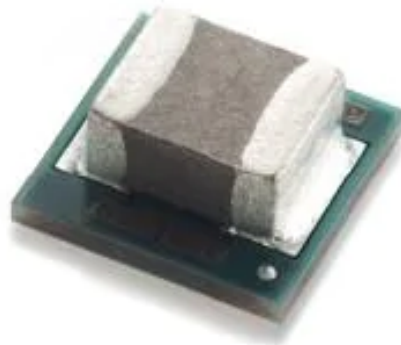
3.3.9.1.3 TLV1117-33CDCYR

This power regulator is an enticing option particularly due to its price. Low dropout regulators are very cheap but also because all of the dropped voltage is converted into heat this means that it is very inefficient. LDOs are most useful when the input voltage is relatively close to the needed output voltage - which is not the case in this application.

Table 17: 12v to 3.3v Buck Regulator Options

Feature	TPS62160DGKR	TPS82150SILR	TLV1117-33CDCYR
Type	Buck Switching	Buck Switching	LDO
Switching Freq	2.25MHz	2.0MHz	N/A
Input Voltage	3v - 17v	3v - 17v	5v - 15v

Efficiency ($I_o = 500$ mA)	~85%	~87%	~30%
Quiescent Current	17 μ A	20 μ A	10 mA
Max Out Current	1000 mA	1000 mA	800 mA
Manufacturer	Texas Instruments	Texas Instruments	Texas Instruments
Size	2.1mm x 2.1mm	2.9mm x 3.1mm	6.7mm x 7.3mm
Price	\$1.59	\$3.18	\$0.62



*Figure 58: TPS82150SILR Power Regulation Module
(Reprinted with Permission from Texas Instruments)*

The TLV1117-33CDCYR is the cheapest out of all of the options and thus looks like the best because it is the lowest cost. However, TLV1117-33CDCYR will likely not work due to thermal issues. Because LDOs dissipate a lot of heat this voltage regulator will not work in our system without any sort of heatsink. The TPS62160DGKR then seems like the best option as it has about the same specifications as the TPS82150SILR. As a result of it not having an integrated inductor means that its cost is not as low as it seems. For smaller runs of the product the additional parts needed may actually be higher than the TPS82150SILR. In a large product manufacturing runs TPS62160DGKR would be a better alternative.

3.3.9.2 LiPo/5v to 3.3v Regulator

This regulator is needed in both of the battery operated devices in order to provide 3.3v to the MCU and Radio. This regulator will be mostly powered off of the LiPo battery but will also need to be able to regulate the 5v that a USB charger will supply as such it must be able to both step up and step down the voltage to an adequate level. The current load is much less on this device as the MCU and radio are very low power devices. As such it

needs to maintain high power efficiency at high and very low current draw in the 2.8-4.2 v range of the battery.

Requirements:

- Efficiency: 80% +
- Input Voltage: 2.8-5 v
- Output Voltage: 3.3 v
- Output Current: 200 mA
 - MCU: ~ 10 mA
 - Radio: ~ 150 mA
 - 10% Headroom: ~20 mA

3.3.9.2.1 *TPS63031DSKR*

This Power regulator is a Buck and boost regulator which means that it is able to regulate the input voltage to the required voltage of the MCU even if its above or below the input voltage. Switching power supplies are a bit more expensive as a result of being a more complex component but as a result they are very efficient. Which is paramount for our battery powered devices.

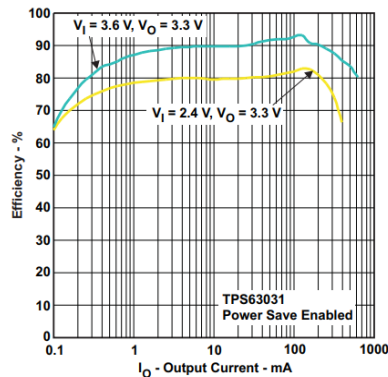


Figure 59: *TPS63031DSKR* Efficiency Curve
(Reprinted with Permission from Texas Instruments)

3.3.9.2.2 *TPS63001DRCR*

This power regulator is very similar to the previous regulator but is able to deliver a higher current. Additionally it runs at a lower switching frequency which means that we would need to use a larger inductor which would increase costs. Otherwise this option is

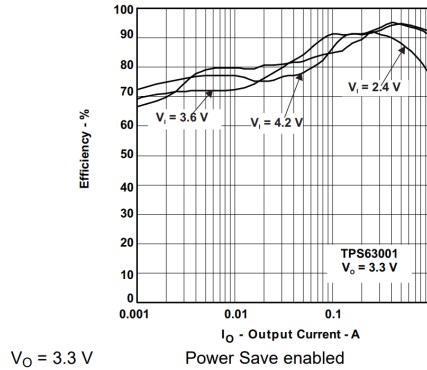


Figure 60: TPS63001DRCR Efficiency Curve
(Reprinted with Permission from Texas Instruments)

3.3.9.2.3 LTC3553

This power regulator is a particularly interesting solution as it also integrates an LDO and a battery charger. Using this power regulator would allow us to reduce our part count which is a positive aspect. The primary downside is that this part does not have much headroom in terms of max current delivery and does not integrate a boost converter which means that at lower voltages of the battery we may not be running at the optimal voltage. LTC3553 from Linear Technology is a one chip solution that is a battery charger IC with a low dropout voltage regulator (LDO) and a buck converter. LDO voltage regulator is a linear circuit, so the efficiency can't be as high as a switching voltage regulator. Therefore, we're only interested in the buck converter.

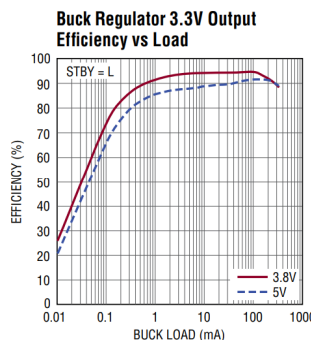


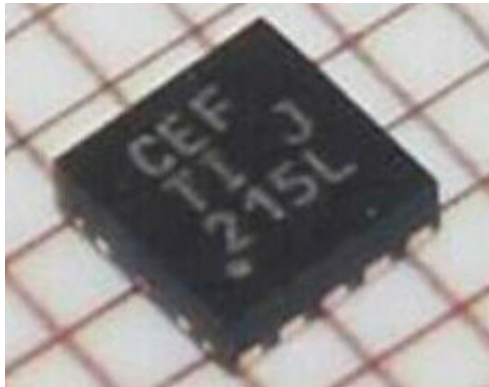
Figure 61: LTC3553 Efficiency Curve
(Reprinted with Permission from Analog Devices)

3.3.9.2.4 TLV75533PDBVR

One of the primary benefits of utilizing an LDO is that they are very cheap and simple to implement. However as a result of this simplicity this means that they are quite inefficient. Additionally they are not able to boost the voltage of the input which will be an issue because the voltage of the battery falls below 3.3v before completely discharging.

Table 18: LiPo/5v to 3.3v Regulator Specification Comparison

Feature	LTC3553	TLV75533PD BVR	TPS63031DS KR	TPS63001DR CR
Type	Buck Switching + LDO	LDO	Buck + Boost Switching	Buck + Boost Switching
Switching Freq	1.125MHz	N/A	2.4MHz	1.25MHz ~ 1.5MHz
Input Voltage Range	2.7V - 5.5V	1.45V - 5.5V	1.8V - 5.5V	1.8V - 5.5V
Efficiency (Io = 200 mA)	~90%	~60%	~80-90%	~85-90%
Quiescent Current	12 μ A	31 μ A	50 μ A	50 μ A
Max Boost Current (Vin=2.8, Vo=3.3v)	N/A	N/A	500 mA	800 mA
Feature	LTC3553	TLV75533PD BVR	TPS63031DS KR	TPS63001DR CR
Max Buck Current (Vin=4.2, Vo=3.3v)	200 mA	250 mA	800 mA	1200 mA
Manufacturer	Analog Devices Inc.	Texas Instruments	Texas Instruments	Texas Instruments
Size	3.0mm x 3.0mm	3.0mm x 3.05mm	2.6mm x 2.6mm	3.1mm x 3.1mm
Price	\$5.61	\$0.39	\$1.87	\$2.21



*Figure 62: TPS63031DSKR Power Regulator
(Reprinted with Permission from Texas Instruments)*

Overall we believe that the TPS63031DSKR is the best power recollection IC for this application. The TLV75533PDBVR is an LDO and does not deliver the efficiency we need although its price is very enticing. The LTC3553 is a nice regulator as it also integrates a battery charger and an LDO but does not integrate a boost converter and is relatively borderline on its max current delivery. Finally TPS63001DRCR is a bit overpowered and thus needlessly increases cost when it is otherwise mostly equivalent to TPS63031DSKR.

3.3.9.3 LiPo to 5v Boost Regulator

In the Vent Damper device there is a motor that needs to be powered off of at least 4.8 V. This is outside of the voltage range that the battery can provide as such we need a boost regulator that can power the relatively high current motor. Because the motor is not constantly used, what is more important over running efficiency is that the regulator is able to be disabled by an MCU to save on power.

Requirements:

- Disable Pin
- Input Voltage: 2.8-4.2 V
- Output Voltage: 5 V
- Output Current: 700 mA (1.25 A switching)
 - Running: ~200 mA
 - Stall: ~650 mA
 - 10% Headroom: ~50 mA

3.3.9.3.1 TPS81256SIPR

This specific boost regulator looks particularly enticing because of the integrated inductor. This is beneficial because it reduces the part count and thus likely the overall cost. However this is at the cost of being less versatile. A boost converter that does not integrate the inductor can typically be configured to best work for the application needed.

As a result this means that at the lower side of input voltages the battery provides this boost converter may not be able to deliver the peak current draws that a motor may require.

3.3.9.3.2 *TPS61256YFFR*

This is a boost converter that requires an external inductor as a result it's output can be tailored more to the requirements. However, as a result this means that it needs a few external components that will complicate the design. This is the case for most switching regulators.

3.3.9.3.3 *LMR62014XMF*

This option is particularly enticing as it is the cheapest out of all of these boost regulators. However, this is because it does not have the built in diodes required for its operation. As a result this means that the ultimate cost of implementation is not less. Additionally at the lower voltages the battery provides this boost converter may not be able to deliver the peak current draws that a motor may require.

Table 19: LiPo to 5v Boost Regulator Specification Comparison

Feature	LMR62014XMF	TPS81256SIPR	TPS61256YFFR
Switching Freq	1.6MHz	3.5MHz	3.5MHz
Input Voltage Range	2.7 v - 14 v	2.3 v - 5.5 v	2.5 v -
Efficiency (Vin=2.8, Io=200mA)	~75%	~88%	~88%
Feature	LMR62014XMF	TPS81256SIPR	TPS61256YFFR
Quiescent Current	1 μ A	30 μ A	25 μ A
Max Switch Current	2 A	4.6 A	2.4 A
Max Output Current (Vin=2.8, Vo=5v)	300 mA	500 mA	900mA
Startup Time	N/A	400 μ s	600 μ s
Manufacturer	Texas Instruments	Texas Instruments	Texas Instruments
Size	1.75mm x 3mm	2.9mm x 2.5mm	1.2mm x 1.3mm

Price	\$0.83	\$2.52	\$1.39
-------	--------	--------	--------



*Figure 63: TPS61256YFFR Boost regulator
(Reprinted with Permission from Texas Instruments)*

Out of these options the only real solution is the TPS61256YFFR because it best fits our electrical requirements with extra headroom. The primary downside is that there is the requirement for a few external components. However this is a necessary drawback as any of the other switching regulators that fit our requirements will have the same issues. The primary concern will be the difficulty of manufacturing as this is a small BGA package that may be difficult to solder.

3.3.10 Battery Charger

The battery charger is important because we must be able to regulate a USB charger to a single cell LiPo battery. It is most important that this charger be reliable in order to ensure safe charging of the device. Particularly in the potential use case is being constantly left in a charger in order to be operated with a wired power supply.

3.3.10.1 MCP73831

The MCP73831 is a simple charging IC that is very inexpensive. This is its primary benefit as it is very simple and cheap to integrate. However this comes with a few downsides as a result of this simplicity it is less efficient and less powerful. More importantly it does not integrate any sort of overcharge protection and will simply cut off charging after 4 hours regardless of the state.

3.3.10.2 BQ24230RGTR

This charging IC is a bit more featured than the MCP73831 and is targeted specifically towards USB charging devices. It features more control and feedback pins, overvoltage protection, and charge termination. Overall this IC provides greater battery charging protections.

3.3.10.3 BQ24195LRGER

This charging IC is also similar to the above one but has even more features. It is able to be controlled by an MCU over I²C and includes even more charging features. It is also able to accept a larger range of input voltages and charge the battery at a higher current rate. However, these advantages come at a slightly higher cost.

Like the below one but with control pins

3.3.10.4 LTC3553

Overall LTC3553 is a good battery charger, It accepts the proper voltage ranges allows for a reasonable charging current. However, the primary benefit of the LTC3553 is that it also integrates a switching voltage regulator into the chip.

Table 20: LiPo to 5v Boost Regulator Specification Comparison

Feature	MCP73831	BQ24230RGTR	BQ24195LRGER	LTC3553
Control	1 CE Pin	1 CE Pin	I ² C	1 CE Pin
Quiescent Current	2 μ A	6.5 μ A	5 μ A	12 μ A
Max Input V	6V	6.4V	17V	5.5V
Chg Current	500mA	500mA	2.5A	420mA
Manufacturer	Microchip Technology	Texas Instruments	Texas Instruments	Analog Devices Inc.
Size	3.1mmx3.2mm	3.1mmx3.1mm	4.1mmx4.1mm	3.0mmx3.0mm
Price	\$0.56	\$2.07	\$2.80	\$5.61

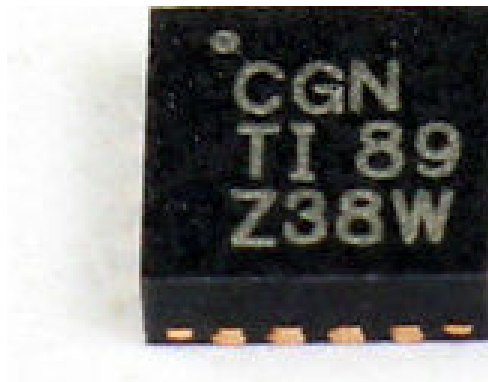


Figure 64: BQ24230RGTR Battery Charging IC
(Reprinted with Permission from Texas Instruments)

Overall the BQ24230RGTR is likely the best IC for our application. The BQ24195LRGER is likely over-specified for our application as we do not need all of the features of this IC and its communication will take up IO pins that are best reserved elsewhere. For the LTC3553 its additional integrated hardware is not as enticing as we determined that we will use a different ICs for charging due to our requirements and its higher cost. Finally the MCP73831 is good but does not allow for the level of control and charging safety that we need in our device

3.3.11 Transistors

There are times in which we need to control a larger current with an MCU. Transistors allow us to control larger currents from the MCU without the risk of damaging the MCU. One of the primary use cases would be controlling a signaling relay as the coil current is often too high for the MCU to directly supply. Overall, The use cases of the transistor vary based on the situation but the primary requirement is cost, size, and current capability.

3.3.11.1 2N4401

The 2N4401 is a larger sized transistor because it is a through hole component. This is a primary benefit because it will be easier when it comes time to finally manufacture our prototypes. However, This can also be a drawback as larger components may complicate routing and board design.

3.3.11.2 ZXTP5240F

There are times when a PNP transistor is needed over an NPN primarily when we need a transistor that is normally open. Its max current flow is 2A so it is sufficient for controlling a high power device.

3.3.11.3 NSS40201LT1G

This transistor is an NPN transistor which means that it is normally open which means that it will normally allow for a device to be turned on. Again like the previous IC this transistor is rated for a high current throughput which means that it can control a higher current device.

Table 21: LiPo to 5v Boost Regulator Specification Comparison

Feature	2N4401	ZXTP5240F	NSS40201LT1G
Max Ic	600 mA	2A	2A
Type	NPN	PNP	NPN
Max Power	625mW	730mW	460mW
Manufacturer	NTE Electronics	Diodes	ON Semiconductor

		Incorporated	
Size	5.2mm x 4.2mm	2.4mm x 2.9mm	3.04mm x 1.04mm
Price	\$3.00	\$0.37	\$0.39

Overall these components are best because they are significantly cheaper. Additionally these parts are able to handle a larger current and power throughput so that they can be used to control high current devices. As well as being in a package that will be relatively easy to solder.

4. Design

This section will cover the overall planned design associated with our IoT implementation and will also review some common circuit and electronic designs which may be relevant to the system. Some of the specific sections in this chapter include relevant electronic circuits accompanied by schematics and diagrams, the prototyping system, testing of various components which will be incorporated into the system, breadboard implementation and testing, PCB schematics and eagle trace diagrams, and the process in which 3D printing will be used towards the design.

This section is the most important section associated with the report as it outlines some of the most crucial steps in the design process and showcases the bulk of the testing and prototyping which has been accomplished thus far. This section will cover the result of all of the research and work which lead up to these prototypes and experiments with said technologies.

This section also showcases the product of the Electrical and Computer Engineering education we acquired throughout our time at the University of Central Florida. It shows that we as upcoming engineers have the ability to apply the knowledge and concepts gained from the engineering curriculum we've gone through over the last few years spent in university.

4.1 Relevant Circuits

This section will outline some of the relevant circuits associated with the electric design of our IoT implementation. These circuits will be integral to accomplishing the design of the smart airflow IoT implementation. As Electrical and Computer engineers, we now have the ability to accurately assess these circuits and properly implement them into our design depending on the specifications associated with the circuits.

In addition to this, the section will show exactly how these circuits will be incorporated into the final system and how they will be used to contribute to the final working implementation. This also means that this section will be technically focused with electrical components, diagrams/ schematics and many specifications benign heavily discussed. Also, more fundamental electrical engineering concepts will be covered which are relevant to the nature in which these circuits operate as well as why they are useful to not just our implementation but in the general case.

4.1.1 Voltage Ladder

This section the voltage ladder circuit and how it will contribute to the system and furthermore be implemented into the final design. In addition to simply explaining the circuit, its basic functionality, and how it will be incorporated into the design, we plan to also provide some diagrams and schematics which further represent the circuit in its entirety. These diagrams and schematics will provide better detail on the specific

components associated with the circuit such as individual resistors, voltage sources, capacitors, and such.

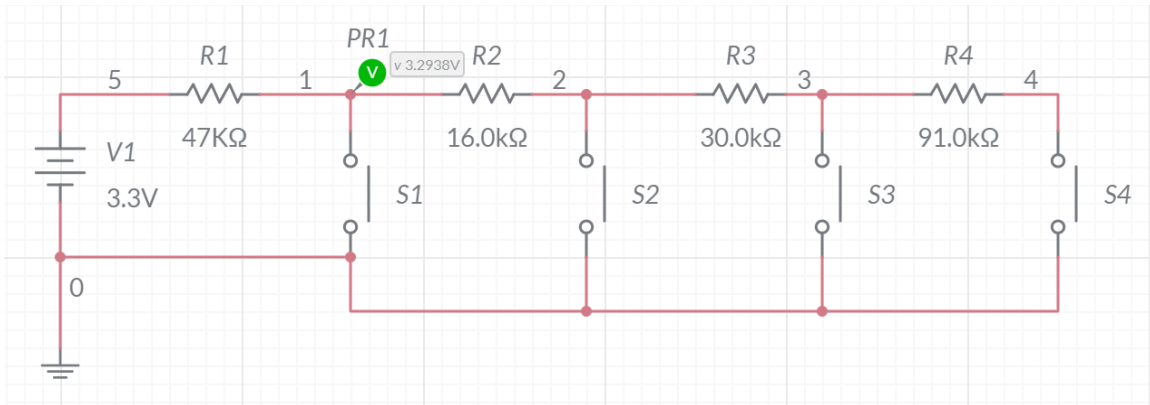


Figure 65: Voltage Ladder

Figure X showcases a Voltage Ladder circuit. Voltage ladders are simple electronic circuits in which a number of resistors are connected in series with a voltage source powering the entire network of resistors. This type of circuit is primarily used for allowing easy access to references of voltages. Depending on where the nodes are connected to the auxiliary component a different voltage will supply the component. This concept is due to the voltage drops which occur with each subsequent resistor which is placed in series in the circuit.

Due to the elementary electrical engineering concept known as Ohm's law these voltages can be relatively easily calculated using the $V = IR$ equation. As most Electrical/Computer engineers know this is quite possibly the most important equation in the entirety of electrical physics as it is the most commonly used equation when solving for currents and voltages in an electronic circuit.

Continuing on with the explanation of the voltage ladder, the current going through each resistor is the same considering the nature of them being in series. Since the resistors are in series and given the nature of the circuit each resistor's voltage can be calculated by multiplying the current going through all the resistors by the respective resistance of each resistor.

4.2 Prototyping

Throughout the development process it is important to prototype and mock up early proof of concepts to ensure our ideas were even feasible. It is also a relatively quick way to get insight on the potential final configuration of the product. This is an incredibly important step in the design process and cannot be overlooked. Not only does prototyping provide a basis to build upon throughout the future development of the design but it also allows for the relatively easy testing and discovery of unintended bugs or design flaws.

This section will provide images and explanations on the state of the prototypes which have been developed thus far. This includes but is not limited to the model HVAC systems, breadboard testing prototypes, and hardware/module testing. These tests and models will allow for a seamless transition into actually developing and improving the system we hope to have finished by the end of the term.

The main emphasis of this section is to showcase the product of the research and development which has occurred over the duration of the semester and provide a working model in which to base the final IoT implementation off of later down the line.

4.2.1 Mock HVAC System

From the beginning of the project we knew that we would not be able to fully implement this system in a full scale home. Primarily due to the inability to demo its full capabilities in person. As such we are designing our product to work around a mock HVAC system that will replace the need for a full sized home.

One of the primary concerns of this part of the project was if we could find an efficient and effective method to regulate the temperature of a small enclosed space. Our solution was to use a peltier unit as a heat pump. One side of the peltier unit has a dense copper heatsink and a fan that will cycle the air inside of the enclosed space which emulates the evaporator inside of a full sized AC system. On the other side a larger heatsink and fan which emulates the condenser of a typical full sized AC system.



Figure 66: Early Prototype of the Mock AC System

Above shows the initial prototype of the cooling system. At the top is the external heat sink that acts like the condenser in a full scale system. At the right is the impeller that circulates the air from “inside” and through the heatsink mounted to the peltier that acts as the evaporator.

Because a peltier acts as an electrical heat pump we can utilize it as both a heater and a cooler. In order to test the effectiveness of this system we ran the system to find the steady state temperature when both in heating and in cooling. The below figures show our temperature readings after a few minutes of running the system.

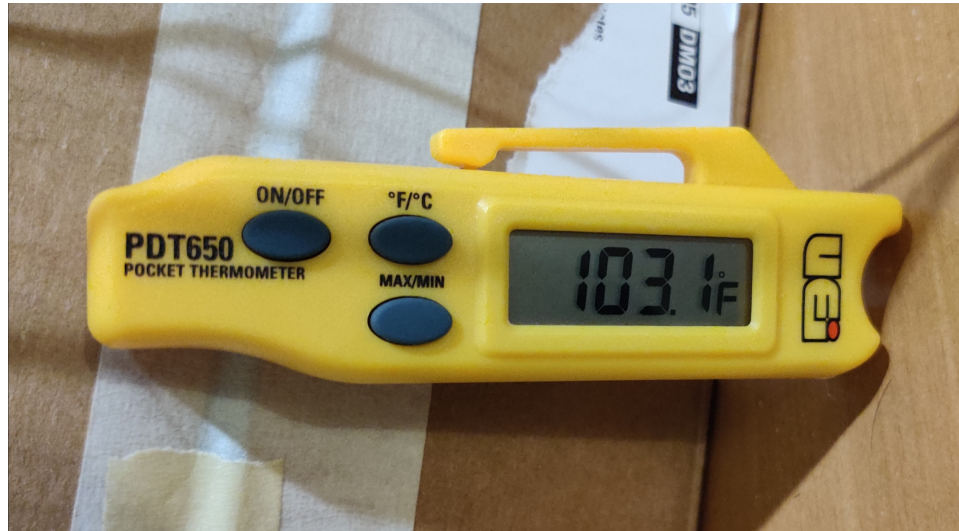


Figure 67: Internal Temperature In Heating Mode

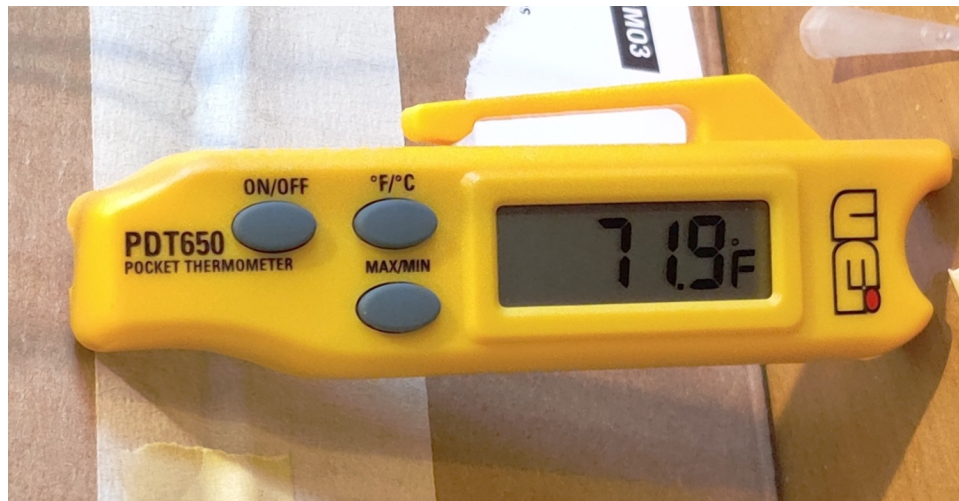


Figure 68: Internal Temperature In Cooling Mode

From our tests we saw that the system is a reasonable testing environment for both prototyping and demonstration purposes. One area to improve in is the cooling performance which we can increase by either getting a more effective external heat sink adding another peltier unit or a combination of both.

4.2.2 Embedded Hardware Testing

There are several software hurdles that need to be overcome with our system. Primarily the messaging system of our product. The requirements of our product led us to use a design where each of the battery powered devices utilize a low power long range radio transmitter and to communicate with a central hub that will then connect via WiFi to our IoT services.

4.2.2.1 Radio Communication

Our decision on which protocol to use was not made without prior testing. We set up a simple messaging protocol between two independent development boards in order to test the feasibility, range, and ease of integration before committing to utilizing the board. Additionally testing these boards revealed certain limitations that informed other hardware choices. Primarily because these boards are not 5V tolerant in order to reduce the part count it was necessary to switch to a 3.3V logic MCU such as the SAMD21 that is utilized on the Arduino Nano 33.

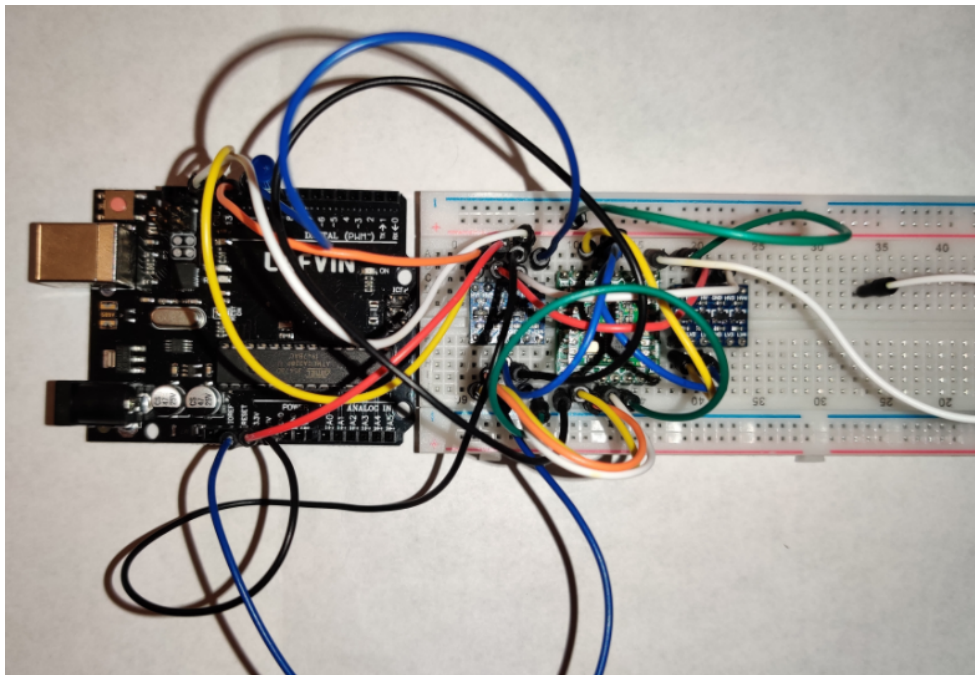
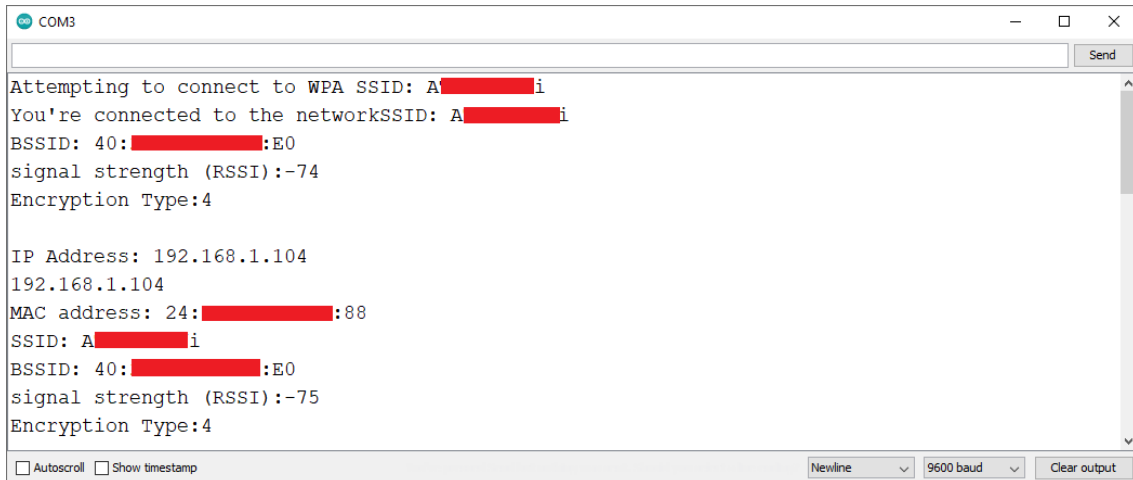


Figure 69: Breadboard Implementation of Radio Communication

The figure above shows our breadboard implementation of our long range low power communication testing. From our tests we found the hardware to be feasible as it was able to easily transmit across a large home even with a suboptimal antenna arrangement. With a more optimal antenna in the final prototype we are confident that it will be able to easily transmit across any sized home.

4.2.2.2 WiFi Communication

One of the other primary components we needed to test was the WiFi communication. In order to confirm the functionality of our system we set up the WiFi communication module and tested the capabilities.



```
COM3
Attempting to connect to WPA SSID: A[REDACTED]i
You're connected to the networkSSID: A[REDACTED]i
BSSID: 40:[REDACTED]:E0
signal strength (RSSI):-74
Encryption Type:4

IP Address: 192.168.1.104
192.168.1.104
MAC address: 24:[REDACTED]:88
SSID: A[REDACTED]i
BSSID: 40:[REDACTED]:E0
signal strength (RSSI):-75
Encryption Type:4

 Autoscroll  Show timestamp
Newline 9600 baud Clear output
```

Figure 70: Serial output of connecting to a password secured network

One of the primary tests we performed was the ability to connect to a WPA secured wireless network. This allowed us to test the functionality of the hardware and how to interface with the arduino drivers. Ultimately we determined that the capabilities of our wireless communication system was sufficient.

4.3 3D printing

In our product 3D printing is a valuable technology because it allows us to rapidly prototype the physical components of our product. Additionally due to the extremely low initial cost it is the most reasonable form of manufacturing for our final products as it is not part of the project's goals to move forward into large scale manufacturing.

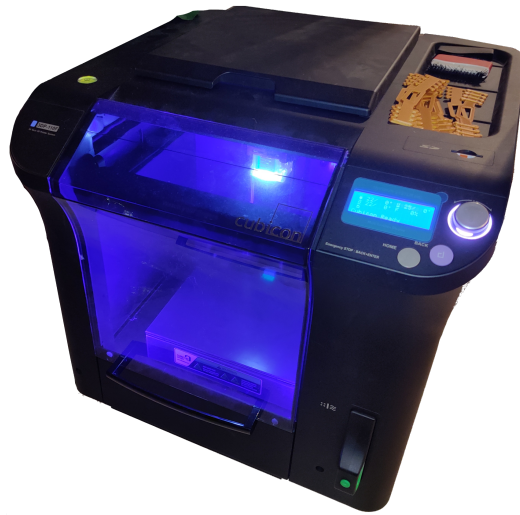


Figure 71: The 3D Printer Used For Prototyping

4.3.1 Valves

The vent valves are an important part of the system as they are the physical mechanism that interfaces with the motor and the vents inside of the system. Due to the flexibility of 3D printing a moving part is able to be printed in place with minimal assembly. As such with careful design and appropriate tolerances we are able to print a sufficiently air tight ball valve.



Figure 72: Iterative Design of Vent Ball Valves

4.4.1 Central Hub

The central hub has the most peripheral devices connected to it and as such is the most complex device in the system. We ultimately utilized nearly every single pin on the MCU meaning we chose an MCU that was not overly complex for the job nor too rudimentary for our needs.

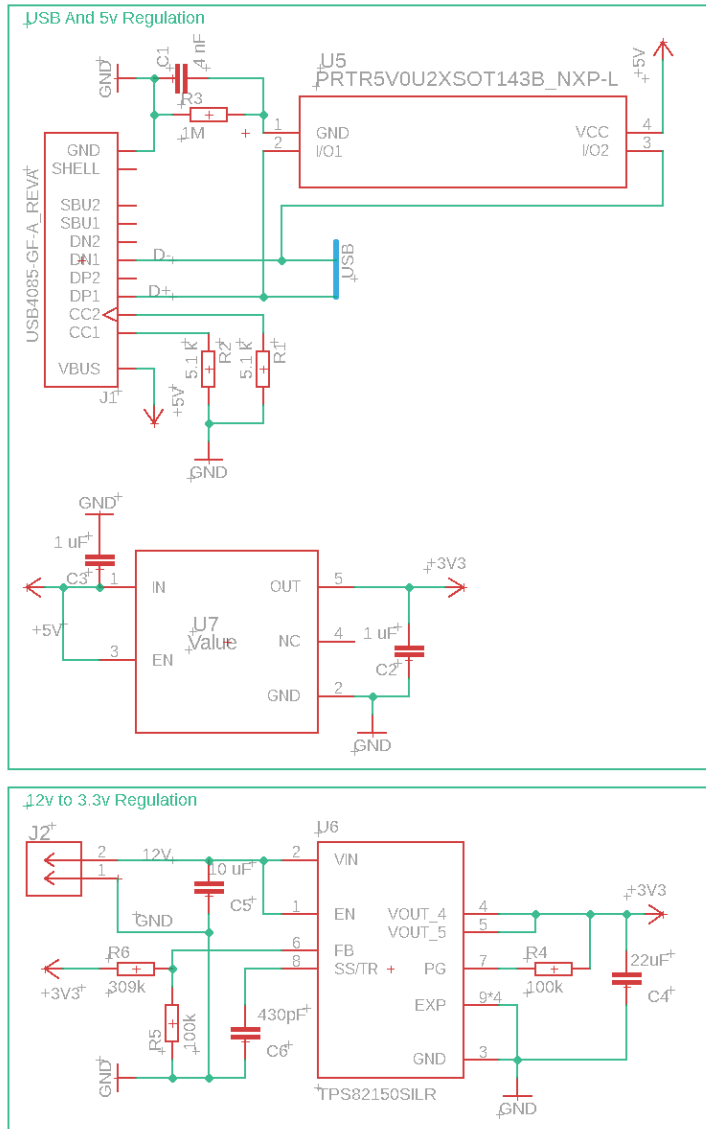


Figure 74: USB and Power Regulation

Above shows the USB and power regulation parts of the board. This device will be primarily powered off of 12v from wire terminals that the AC system provides. Though during operation the USB is not needed for any sort of use but it is still necessary for firmware programming and development purposes. As such we included a cheap LDO that will be able to power the device off of USB power when necessary.

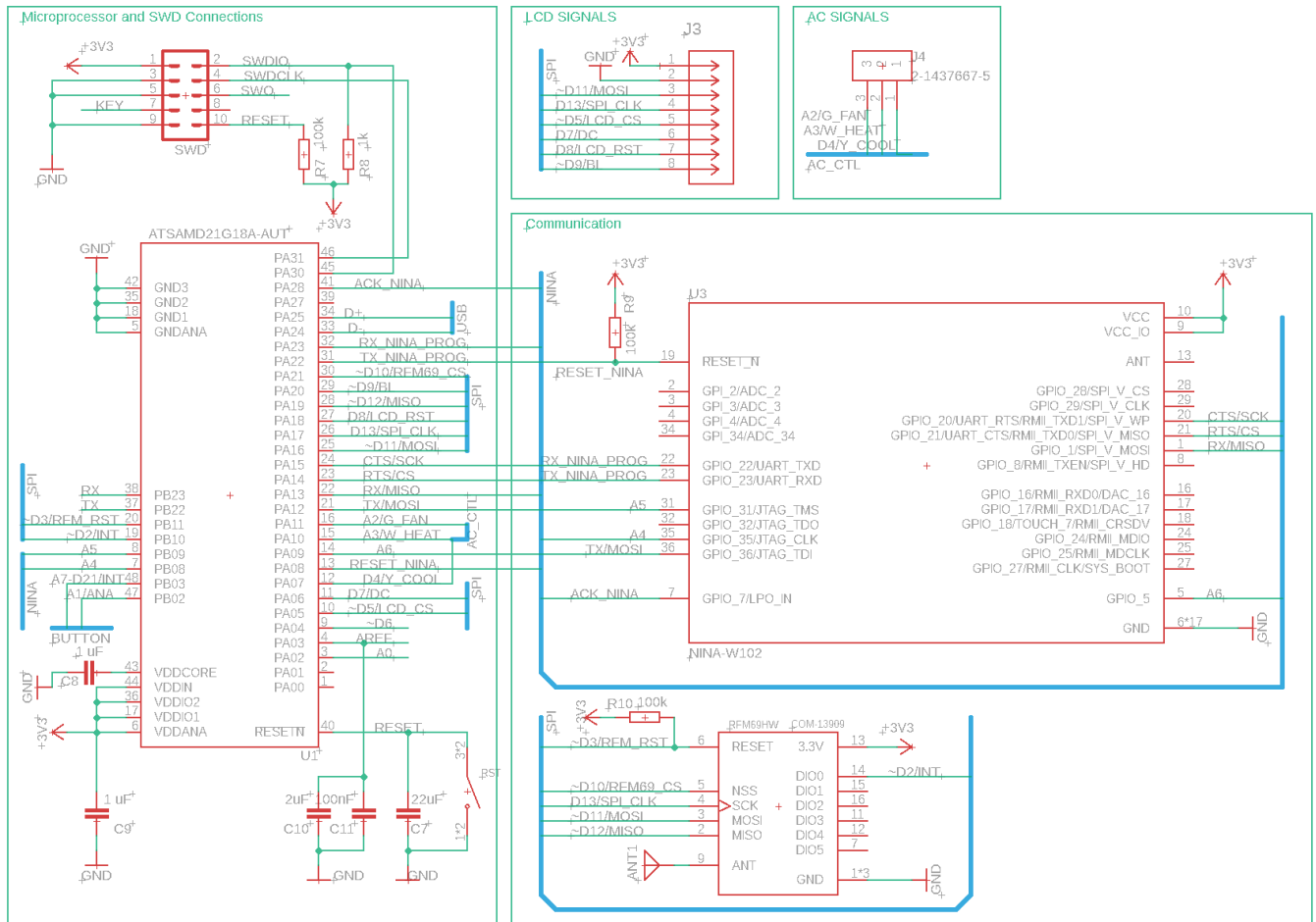


Figure 75: MCU, Communication, LCD, and AC Signals

The above figure shows the majority of the components for the central hub. The primary Connections are the low power long range radio communication, WiFi communication, LCD signals, and Signals for the AC system. The simplest of which is the AC signals which are signals that the MCU will send to the AC system in order to send fan, cooling, or heating commands. Just like in a full scale system it is up to the AC system to interpret these signals. Next is the low power radio communication and LCD display.

To save on needed pins they are able to be put on the same SPI bus so long as each device has its own CS line so that the MCU can choose which device is active. Finally there is the WiFi module that takes up the rest of the pins on the device. These were mapped following the reference schematic of the device. This device has its own firmware that can be updated in addition to its own SPI bus.

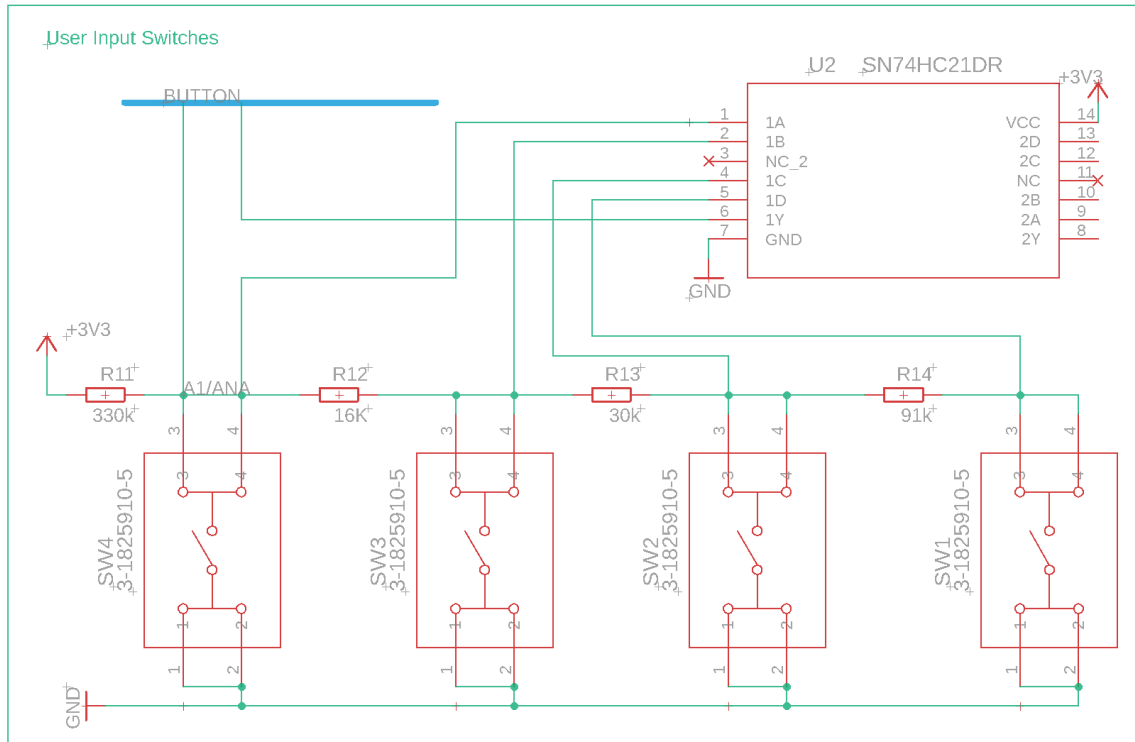


Figure 76: User Pushbuttons

The pushbuttons are an important part of the device’s operation because it allows for the user to interface with the system in a tactile and cost effective way. Utilizing a touch screen would have been costly not just in the BOM cost but also in system complexity. Using buttons requires little much less pins over a touch screen interface as well as is much less complex to interface with meaning the device will be much easier to interface with.

The primary design issue was to be able to read 4 different button presses with only a few pins. As discussed in a previous section using a voltage ladder means that we can read the voltage and determine which buttons are being pressed. An alternative would have been to use a multiplexer which can also serve this purpose. This is better over using a multiplexer because not only is it a more complex implementation but multiplexing is most efficient when it comes to many buttons not only a few. In fact this solution only requires one pin to read the button state.

One issue with the voltage ladder is if you choose the same resistor for each section of the voltage ladder the voltage reading follows a logarithmic curve rather than a linear one. Meaning that the analog margin of error in the last few buttons are disproportionately harder to read over the first few. To solve this you can calculate the optimal resistor arrangement in order to gain this linear relationship.

The other issue with this design is that it is impossible to connect an interrupt pin to the device without breaking down the functionality of the voltage ladder. Even though there

are two pins on the switch it is still a single pole single throw switch. A solution would have been to find a dual pole single throw switch. However, it appears that this type of switch is not readily available in a tactile momentary switch form. This means we needed to add in some external logic via a hardware AND gate. This is much better over the alternative of constantly polling the analog value as this means the cpu will not be overly burdened with the simple task of detecting user input and can focus on the primary task of LCD output and wireless communication. Using this AND gate allows us to quickly and efficiently respond to user input without ruining the functionality of the voltage ladder.

4.4.2 IoT Sensor

This section will provide sample schematics generated in EAGLE which represent the intended design of the IoT sensor which will be used throughout the system. These schematics provide an inside look to how these technologies work and show the individual components and modules which make up the relatively complex hardware. In addition to simply showcasing these schematics, this section will also accompany them with explanations of exactly what these devices are intended to be used for and how they will be incorporated into the final design of the IoT small scale model of a smart air conditioning system.

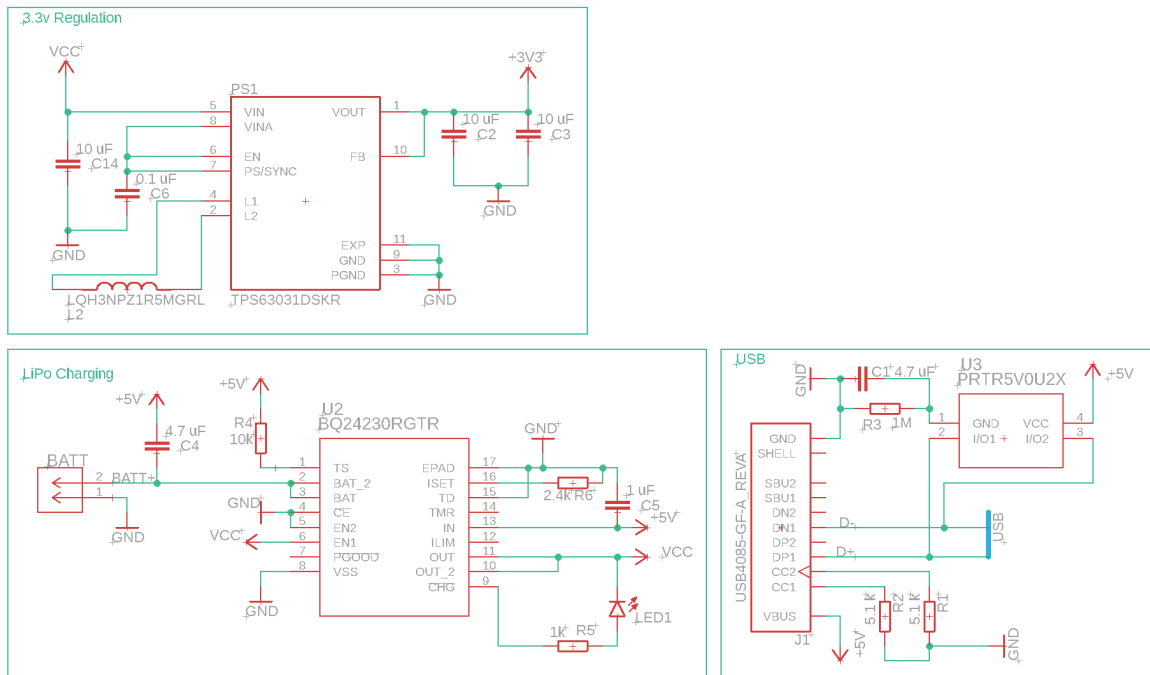


Figure 77: IoT Sensor Power Regulation

In the IoT sensor the power needs of the device are significantly different than that of the central hub. This is because it runs off of lipo power or 5v USB power. This means that it needs to include a more efficient voltage regulator vs the cheap LDO for 5v on the central hub. Additionally this means that there needs to be a lipo charging circuit to handle the battery management power supply and charging. As a result the wire terminals of the

central hub are forgone for a JST battery connector and a high efficiency boost-buck switching regulator in order to regulate and deliver the power to the device. This solution is obviously slightly more expensive but as discussed in the part selection and research it is needed in order to meet the operational requirements of our device.

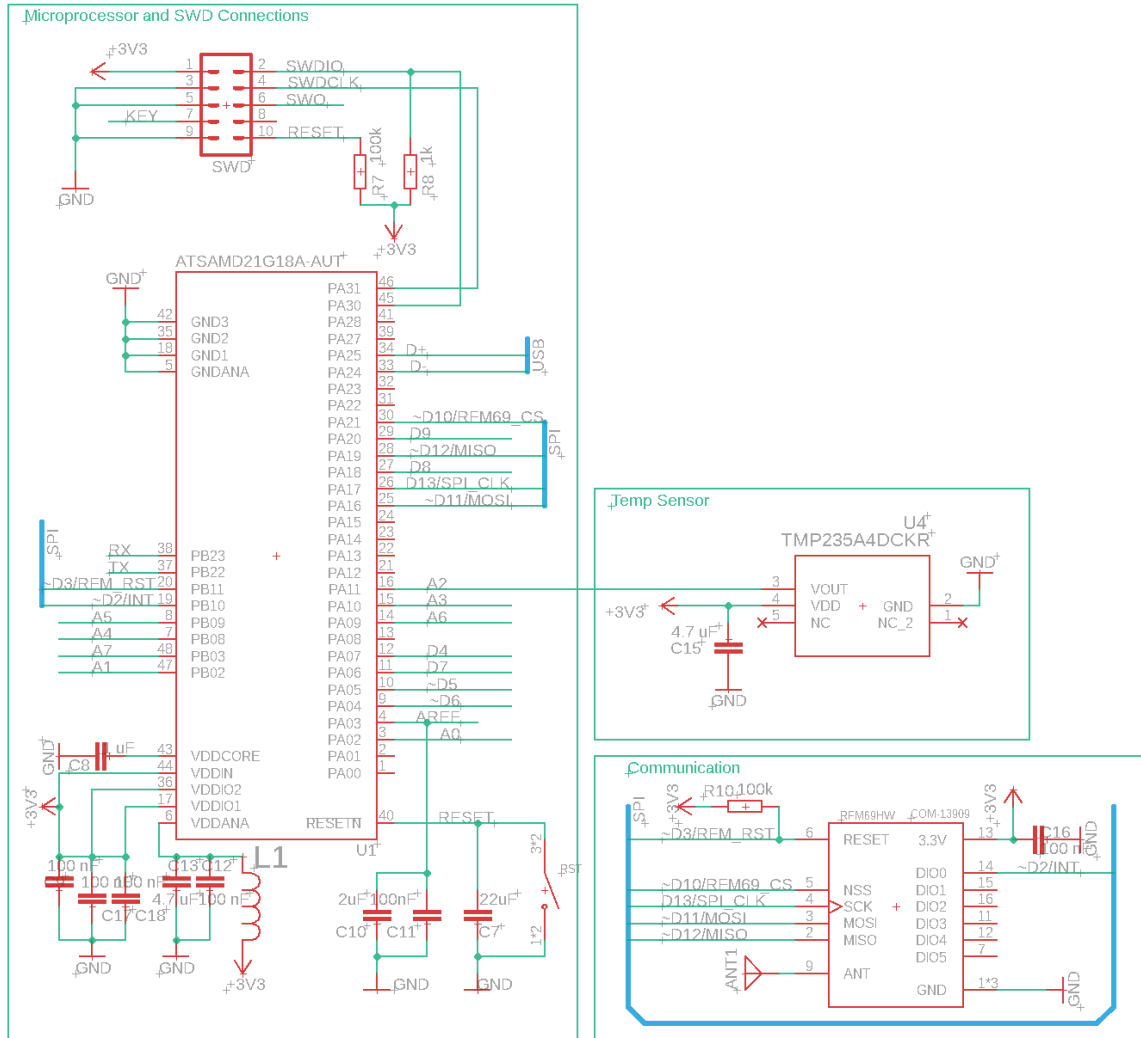


Figure 78: MCU, Communication, and Temperature sensor

The main portion of the schematic compared to the central hub is significantly more simple. This is because it does not need to interface with the internet, have a display, or communicate with an external AC system. As a result the system is much less complex. What it does keep is the lower power radio communication in order to communicate with the central hub so that it may report its temperature readings. This leads into the final difference as it includes a simple analog temperature sensor which it will periodically take measurements with in order to then send over the updated temperature to the central hub utilizing the radio.

4.4.3 IoT Vent Damper

This section, similar to the previous one provides schematics and explanations regarding the IoT vent damper which will be used in our final IoT implementation. This allows for an in-depth look as to how these components are organized in their respective devices. Similarly this section provides almost the same use for IoT vent dampers as the previous section serves to IoT sensors. These schematics were generated in EAGLE and represent the design of the electrical design of these technologies.

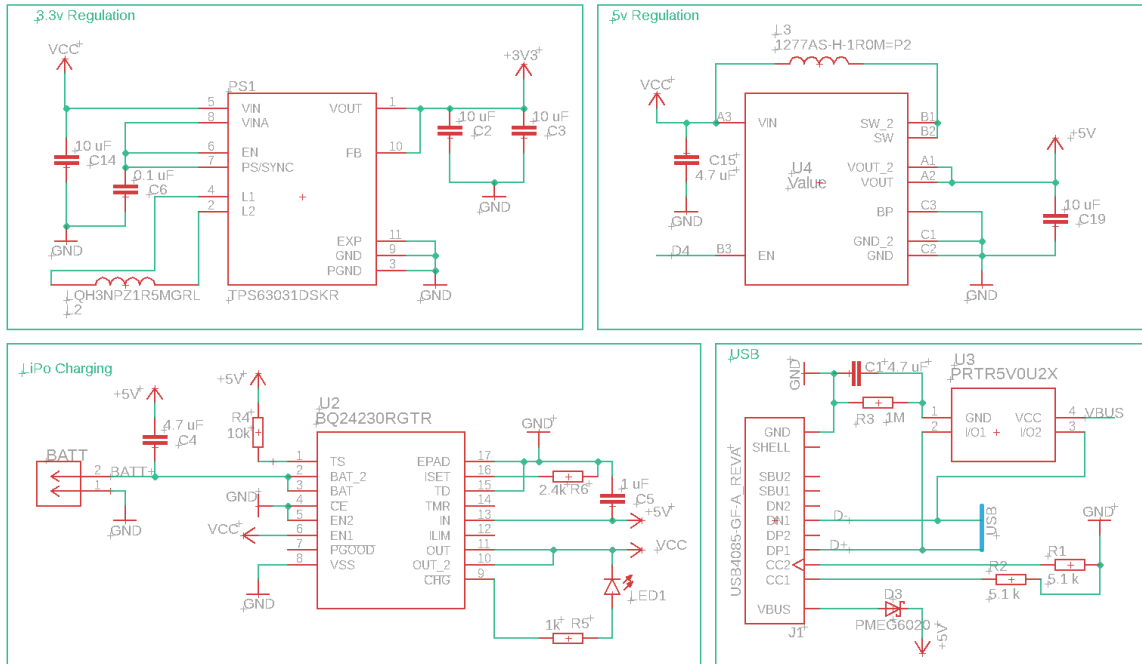


Figure 79: IoT Vent Damper USB, 3.3v Regulation, 5v Regulation, Lipo Charging

Though very similar to the IoT Sensor due to the power requirement of the servo motor an additional IC is required. From the specifications of the motor it requires a minimum of 4.8v which means that we need a separate IC that can regulate the 3.7 nominal voltage of the battery to 5v for the servo. This does not pose much of a design issue other than needing to protect the usb port from backpowering. It is dangerous for a usb device to supply bus power when it should not. Therefore we must insert a diode to protect the device.

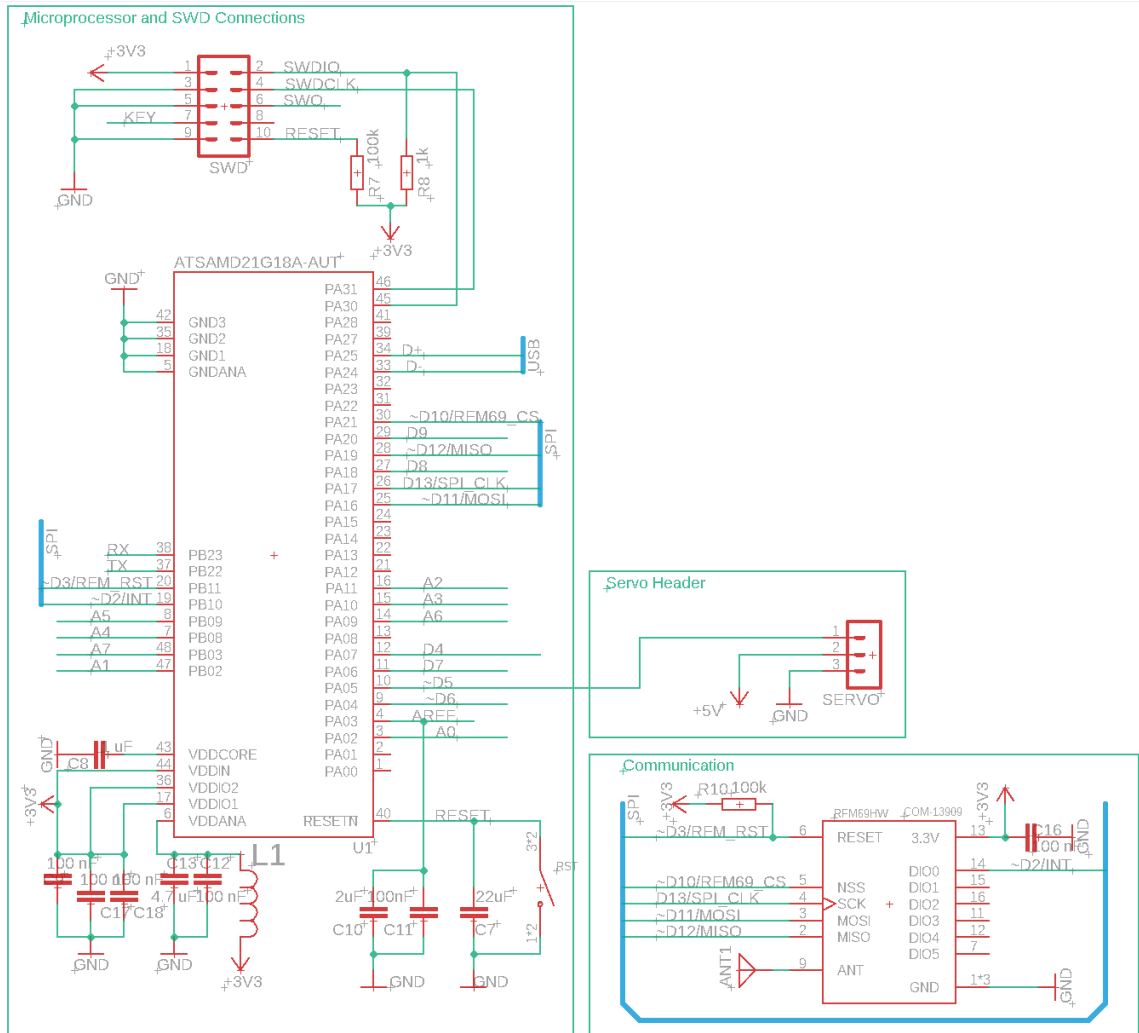


Figure 80: IoT Vent Damper MCU, Servo Headers, and Communication

The rest of the schematic is very similar to the IoT sensor in that it forgoes the more complex IO that is not needed such as the display and wifi. However instead of interfacing with a temperature sensor it interfaces with a servo motor with one of the pwm pins. One of the important portions is that the MCU is connected to the 5V regulation so that it can shut it off in order to preserve power.

4.5 PCB

The design of the PCB is also critical to our product. A poorly designed PCB can take a good schematic and ruin it by making it hard to manufacture, decrease signal integrity, reduce power supply efficiency (by load line voltage drop), or make the product larger than intended. In terms of manufacturing the PCB there are many services that will cheaply produce the boards and even manufacture most if not all of the board.

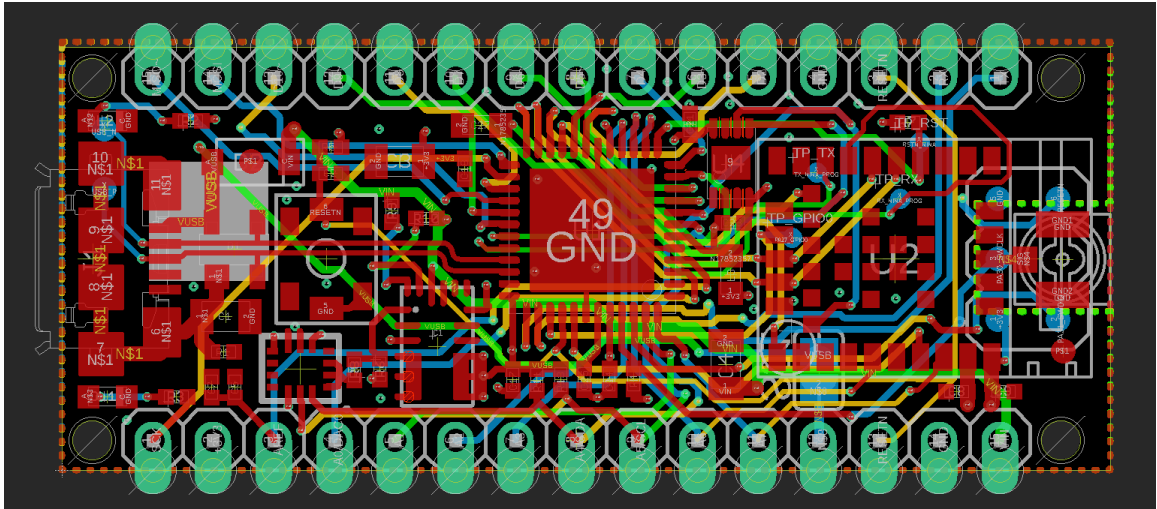


Figure 81: Nano 33 IoT Reference PCB (Creative Commons: Arduino)

4.5.1 Central Hub

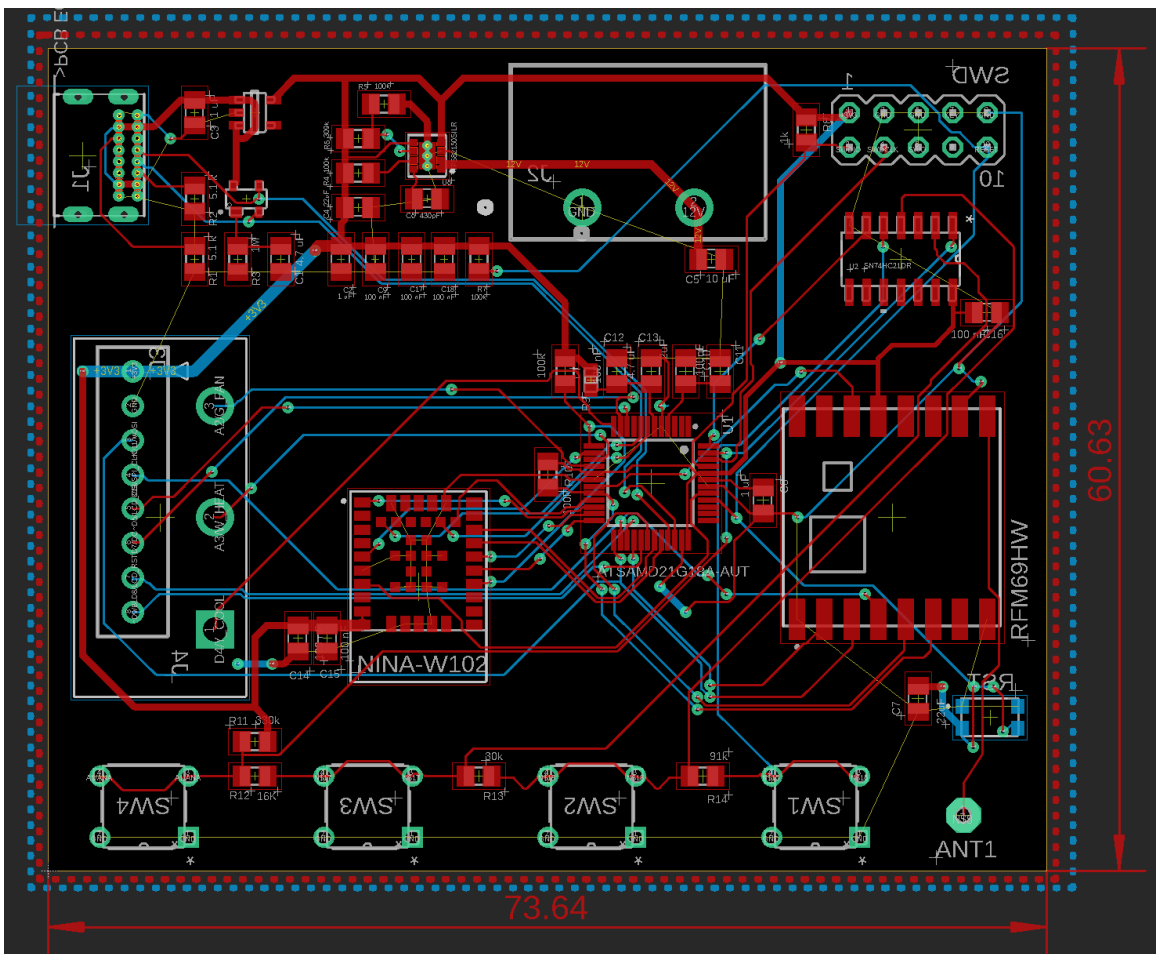


Figure 82: Central Hub PCB

4.5.2 IoT Sensor

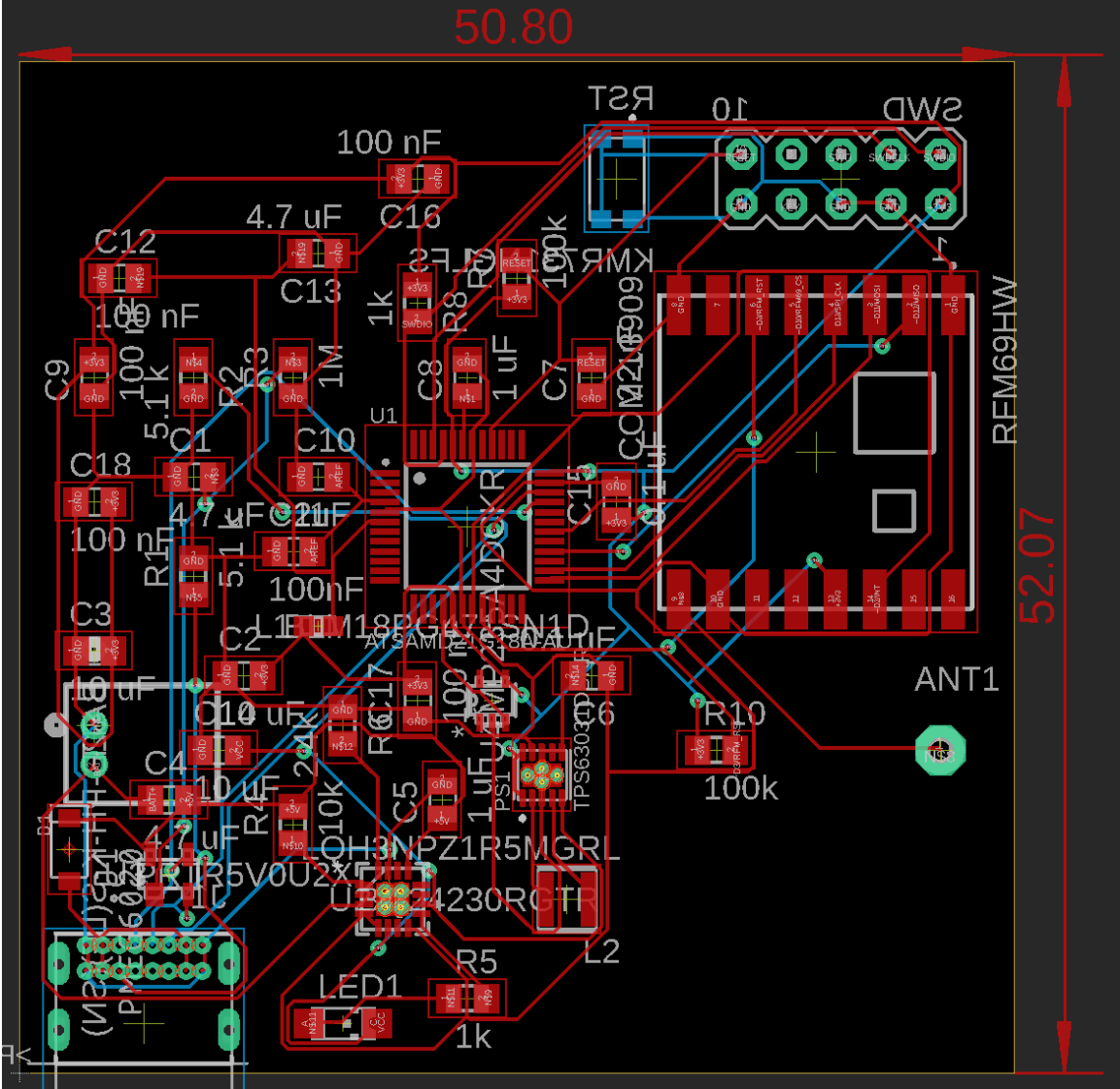


Figure 83: IoT sensor PCB

4.5.3 IoT Vent Damper

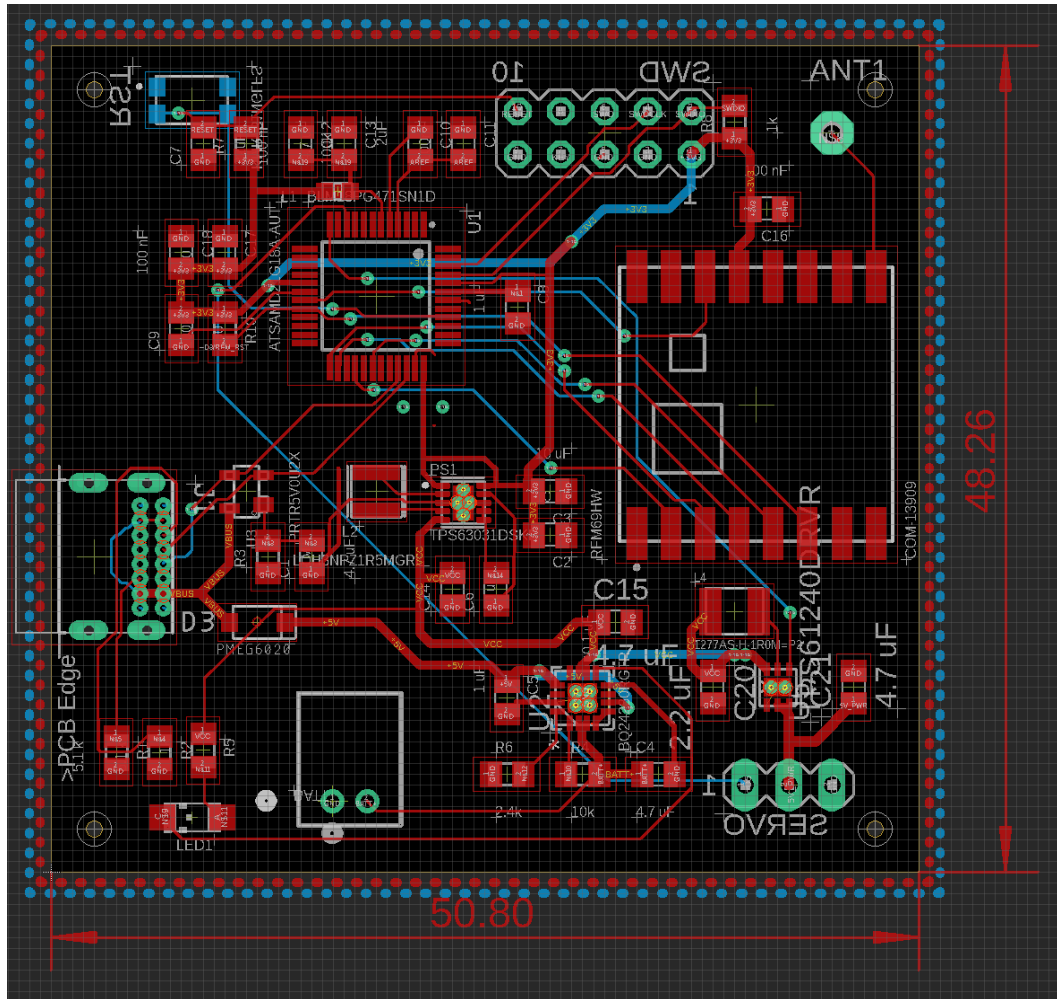


Figure 84: IoT Vent Damper PCB

4.6 Application Design

While our mobile application will provide only basic functionalities to the user for them to connect to and manage devices, it is still important to design an application that has both a nice user interface (UI) and a positive user experience (UX). Our mobile application will serve four main purposes: Login/Register, device connectivity, temperature control and view. We have created a mobile application design mockup using Figma. This section will highlight the main functionalities of the mobile application, and it's corresponding mockup from Figma.

First, the mobile app will prompt the user to register for an account if they do not have one already. If they do, they can log in. Once they register an account with an email and a password, they can log into the application.

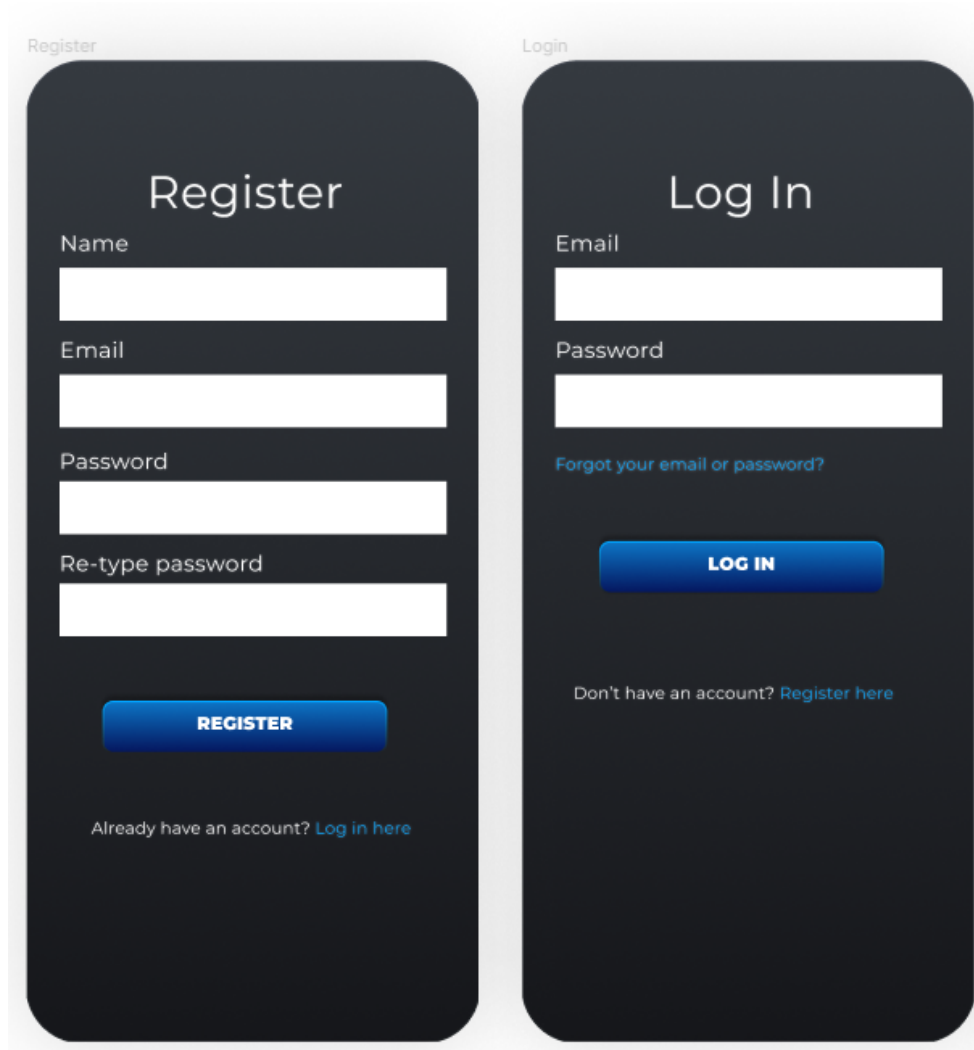


Figure 85: Register and Login Screens

For new users, the application will then prompt the user to search for new central hub devices on their WiFi network. The user will then select an available hub. This will register the selected hub under the user's account. This will be the hub that controls the vents and sensors throughout the user's home.

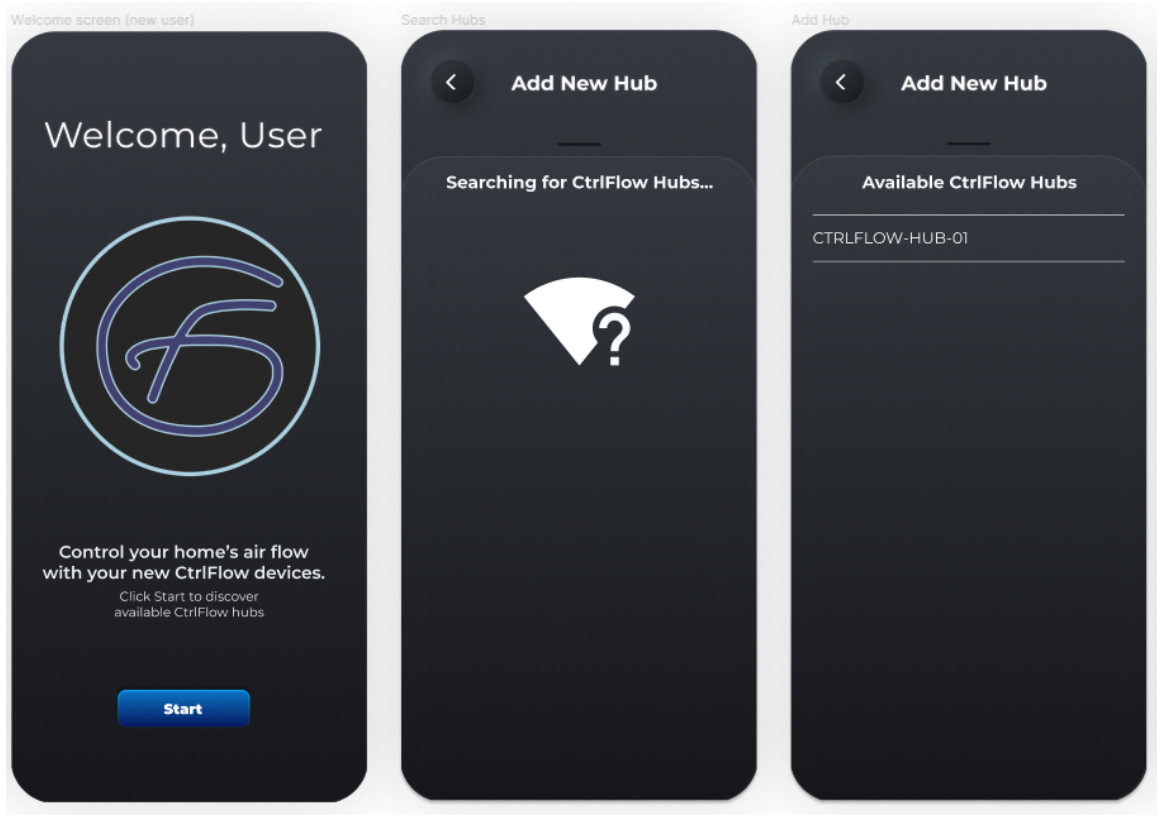


Figure 86: Welcome and Add New Hub Screens

After a hub is selected, the user will then be able to add new vent/sensor pairs and assign them to rooms. These vent/sensor pairs each act as an individual device because in order for a vent to be able to know to allow air flow or not, there needs to be a sensor paired with it in the same room to give it temperature details. As for the sensor, there is no point for a sensor to be assigned to a room without there being a corresponding vent in the same room. Once a pair is selected and assigned to a room, it is then registered under the central hub that was added in the previous step above. The central hub can then communicate with the sensor which communicates with its corresponding vent.



Figure 87: Add New Device and Assign Room Screens

Once there are vents/sensors assigned to rooms and registered with the central hub, the user can then view and update the temperatures in each room that the vents/sensors are assigned to. Aside from changing the temperature, the user also has three modes to set for the room: Auto, Cool, and Heat. Another option the user has when they click on a specific room is to turn on or off the system completely for that room. When the user is on the Home screen, they can view all of the rooms, their current temperatures, and what they are set to. They can also rename rooms, delete rooms, and add new devices to assign to other rooms.



Figure 88: Home Screen and Room Details

5. Budget and Financing

The items included in our budget below include price estimates for parts in the final prototype as well as any development kits. Our project will be self-funded.

Table 21: Developmental Costs

Sub System	Item	Quantity	Price Per Item	Price Estimate
Central Hub	WiFi Module	1	\$8	\$8
	LCD Display	1	\$9	\$2
	Relays	3	\$5	\$15
IoT Sensor	Temperature Sensor	2	\$0.5	\$1
IoT Vents	Servo	2	\$5	\$10
All	Custom PCBs + SMD Components	5	\$20	\$100
	MCU	5	\$3	\$15
	Radio Module	5	\$5	\$25
Heating & Cooling System	Peltier Module	1	\$10	\$10
	Heatsink	1	\$15	\$15
	Fan	1	\$5	\$60
Misc	Arduino Development Kits	3	\$20	\$60
	Cloud Hosting Service	1	\$10	\$10
			Total	~ \$430

6. Project Milestones

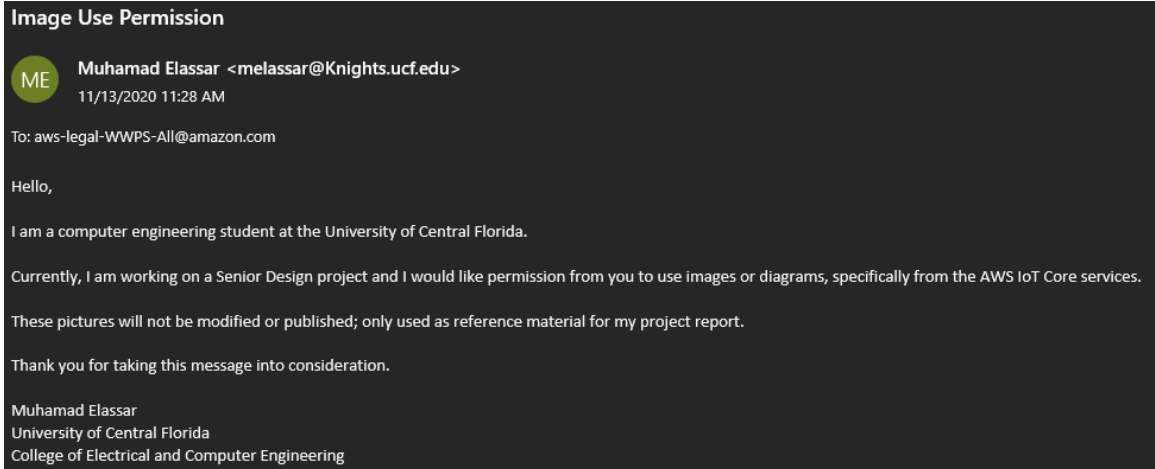
The milestones included below are the tasks that need to be completed and their associated date ranges to complete them. It also includes who will be responsible for each task.

Table 22: Project Milestones

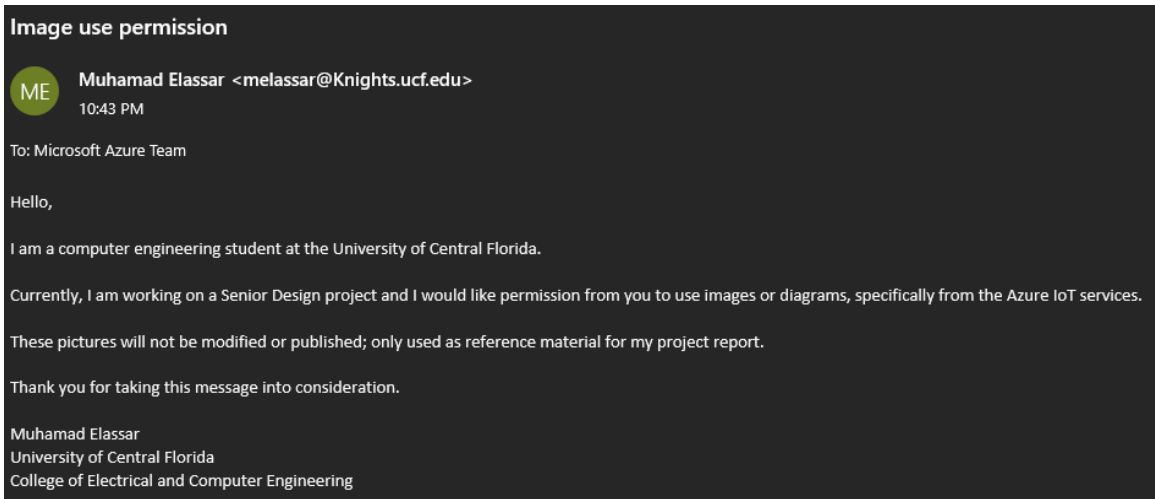
Number	Task	Start	End	Responsible
1	Ideas	08/24/2020	08/28/2020	All
2	Project Selection	08/31/2020	09/11/2020	All
	Project Report			
3	Divide & Conquer	09/07/2020	10/02/2020	All
4	Table of Contents	09/21/2020	10/23/2020	All
5	First Draft	10/05/2020	11/13/2020	All
6	Final Document	08/24/2020	12/08/2020	All
	Research & Design			
7	Power management	09/21/2020	10/02/2020	Joseph + Vinh
8	Hardware design	09/21/2020	10/16/2020	Joseph + Vinh
9	Connection protocol	09/28/2020	10/16/2020	Joseph
10	Mobile app backend	10/12/2020	11/06/2020	Yoseph+Muhamad
11	PCB schematic	10/12/2020	10/16/2020	Joseph
12	Main hub PCB layout	10/19/2020	10/23/2020	Vinh
13	End devices PCB layout	10/26/2020	11/06/2020	Vinh
14	Mobile app frontend	11/09/2020	11/27/2020	Muhamad+Yoseph
16	Order and Test parts	10/19/2020	12/08/2020	Joseph

Appendix A: Copyright Permission

Permission to use images from Amazon Web Services




Permission to use images from Microsoft Azure



Permission to use images from Google Cloud

Image use permission

 **Muhamad Elassar** <melassar@Knights.ucf.edu>
3:47 PM

To: lers-help@google.com

Hello,

I am a computer engineering student at the University of Central Florida.

Currently, I am working on a Senior Design project and I would like permission from you to use images or diagrams, specifically from the Google Cloud IoT services.

These pictures will not be modified or published; only used as reference material for my project report.

Thank you for taking this message into consideration.

Muhamad Elassar
University of Central Florida
College of Electrical and Computer Engineering

Permission to use images from DigiKey



Yousef Mansy <yousefmansy1@gmail.com>
to sales ▾

Hello,

I am a computer engineering student at the University of Central Florida.

Currently, I am working on a Senior Design project and I would like permission from you to use images or diagrams, specifically DigiKey product images.

These pictures will not be modified or published; only used as reference material for my project report.

Thank you for taking this message into consideration.

Joseph Mansy
University of Central Florida
College of Electrical and Computer Engineering

 Reply

 Forward

Permission to use images from Waveshare



Yousef Mansy <yousefmansy1@gmail.com>
to service ▾

Hello,

I am a computer engineering student at the University of Central Florida.
Currently, I am working on a Senior Design project and I would like permission from you to use images or diagrams, specifically LCD display product images.
These pictures will not be modified or published; only used as reference material for my project report.
Thank you for taking this message into consideration.

Joseph Mansy
University of Central Florida
College of Electrical and Computer Engineering

Permission to use images from Texas Instruments



Yousef Mansy <yousefmansy1@gmail.com>
to efishpaw ▾

Hello,

I am a computer engineering student at the University of Central Florida.
Currently, I am working on a Senior Design project and I would like permission from you to use images or diagrams, specifically switching power regulators and other ICs product images.
These pictures will not be modified; only used as reference material for my project report.
Thank you for taking this message into consideration.

Joseph Mansy
University of Central Florida
College of Electrical and Computer Engineering

Permission to use images from Analog Devices



Yousef Mansy <yousefmansy1@gmail.com>
to CorpComm ▾

Hello,

I am a computer engineering student at the University of Central Florida.
Currently, I am working on a Senior Design project and I would like permission from you to use images or diagrams, specifically product images and datasheet graphs.
These pictures will not be modified; only used as reference material for my project report.
Thank you for taking this message into consideration.

Joseph Mansy
University of Central Florida
College of Electrical and Computer Engineering

Permission to use images from Flair



Burt Reich (Flair)

Nov 12, 2020, 9:23 PST

Hello Yoseph!

I spoke with our CEO who granted you permission to use the images of Flair products in Amazon.com's product descriptions.

Good luck with your Senior Design Project!

Best Regards,

Burt

Flair Customer Support



Yoseph Hassan

Nov 10, 2020, 19:58 PST

Hello,

My name is Yoseph Hassan. I am computer engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the following images from your listing on Amazon.com. I have attached the pictures to this email. These pictures will not be modified or published; only used as reference material for my project report. Thank you for taking this message into consideration.

Yoseph Hassan
University of Central
College of Electrical and Computer Engineering

Permission to use images from Keen

Hello,

My name is Yoseph Hassan. I am computer engineering student at the University of Central Florida. Currently, I am working on a Senior Design project and I would like permission from you to use the following images from your listing on Amazon.com. I have attached the pictures to this email. These pictures will not be modified or published; only used as reference material for my project report. Thank you for taking this message into consideration.



Yoseph Hassan
University of Central
College of Electrical and Computer Engineering

Appendix B: References

Similar Projects

“Smart Vents.” Flair. Retrieved October 13, 2020, from <https://flair.co/pages/central-systems-smart-vents>

“How it works.” Keen Home. Retrieved October 13, 2020, from <https://keenhome.io/pages/how-it-works>

Internet of Things

“Internet of Things (IOT).” SAS: Analytics Software & Solutions. Retrieved December 2, 2020, from https://www.sas.com/en_us/insights/big-data/internet-of-things.html

“What is AWS IoT? - AWS IoT Core.” Amazon Web Services, Inc. Retrieved December 2, 2020, from <https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>

“Introduction to Azure IoT Hub.” (2019, August 8). Microsoft Docs. Retrieved December 2, 2020, from <https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub>

“What is Azure IoT Central.” (2020, November 23). Microsoft Docs. Retrieved December 2, 2020, from <https://docs.microsoft.com/en-us/azure/iot-central/core/overview-iot-central>

“Cloud IoT Core overview | Cloud IoT Core Documentation .” Google Cloud. Retrieved December 2, 2020, from <https://cloud.google.com/iot/docs/concepts/overview>

Mobile Application Development

“Mobile Operating System Market Share Worldwide.” StatCounter Global Stats. Retrieved November 21, 2020, from <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201910-202010-bar>

K, A. (2017, September 26). “Is Swift the Future of Server-side Development” Solutions Review. Retrieved November 21, 2020, from <https://solutionsreview.com/application-development/is-swift-the-future-of-server-side-development/>

“Flutter vs Native vs React-Native: Examining performance.” (2020, July 28). Medium. Retrieved November 22, 2020, from <https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980>

“AWS Databases for Real-Time Applications - Amazon Web Services.” Amazon Web Services, Inc. Retrieved December 3, 2020, from <https://aws.amazon.com/products/databases/real-time-apps-elasticache-for-redis/>

“Introduction to Azure Storage - Cloud storage on Azure.” (2020, April 8). Microsoft Docs. Retrieved December 3, 2020, from <https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction>

“Firebase Realtime Database.” Firebase. Retrieved December 3, 2020, from <https://firebase.google.com/docs/database>

Appendix C: Datasheets

Microcontroller Units

“ATmega4808/4809 Data Sheet.” Microchip. Retrieved November 21, 2020, from <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega4808-09-DataSheet-DS40002173B.pdf>

“SMART ARM-Based Microcontroller.” Atmel. Retrieved November 21, 2020, from https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Atmel-42181-SAM-D21_Datasheet.pdf

“CC2652R SimpleLink™ Multiprotocol 2.4 GHz Wireless MCU.” Texas Instruments. Retrieved November 21, 2020, from <https://www.ti.com/lit/ds/symlink/cc2652r.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607224338035>

“EFR32BG22 Wireless Gecko SoC Family Data Sheet.” Silicon Labs. Retrieved November 21, 2020, from <https://www.silabs.com/documents/public/data-sheets/efr32bg22-datasheet.pdf>

“ESP8266EX Datasheet.” Espressif. Retrieved November 21, 2020, from https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

Wi-Fi Transceivers

“NINA-W10 series.” U-Blox. Retrieved November 25, 2020, from https://www.u-blox.com/sites/default/files/NINA-W10_DataSheet_%28UBX-17065507%29.pdf

“ESP32WROOM32D & ESP32WROOM32U Datasheet.” Espressif. Retrieved November 21, 2020, from https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf

ISM Band Transceivers

“HC-12 Wireless RF UART Communication Module v2.4 User Manual”. Seeed Technology Co., Ltd. Retrieved November 21, 2020, from http://statics3.seeedstudio.com/assets/file/bazaar/product/HC-12_english_datasheets.pdf

“RFM95/96/97/98(W) - Low Power Long Range Transceiver Module.” HopeRF Electronic. Retrieved November 21, 2020, from https://www.rfsolutions.co.uk/downloads/1463993415RFM95_96_97_98W.pdf

“RFM69HCW ISM TRANSCEIVER MODULE v1.1.” HopeRF Electronic. Retrieved November 21, 2020, from <https://cdn.sparkfun.com/datasheets/Wireless/General/RFM69HCW-V1.1.pdf>

Display Units

“ACM1602k(3.8v) Series Character Module Ver2.0.” AZ Displays. Retrieved November 22, 2020, from <https://www.azdisplays.com/PDF/acm1602k.pdf>

“2.4inch LCD Module - Waveshare Wiki.”. Waveshare. Retrieved November 21, 2020, from https://www.waveshare.com/wiki/2.4inch_LCD_Module

“1.54inch e-Paper Module - Waveshare Wiki.” Waveshare. Retrieved November 21, 2020, from https://www.waveshare.com/wiki/1.54inch_e-Paper_Module

Temperature Sensors

“Thermocouple Type-K - Glass Braid Insulated (Bare Wire).” (2015, February 18). SparkFun Electronics. Retrieved November 21, 2020, from <https://www.sparkfun.com/products/251#features-tab>

“TMP23x Low-Power, High-Accuracy Analog Output Temperature Sensors.” Texas Instruments. Retrieved November 21, 2020, from <https://www.ti.com/lit/ds/symlink/tmp235.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607195744230>

“MAX31820 - Wire Ambient Temperature Sensor.” Maxim Integrated. Retrieved November 21, 2020, from <https://datasheets.maximintegrated.com/en/ds/MAX31820.pdf>

Motors

“Small Reduction Stepper Motor - 5VDC 32.” Adafruit Industries LLC. Retrieved November 21, 2020, from https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/858_Web.pdf

“Miniature Linear Motion Series - L12.” Actuonix. Retrieved November 21, 2020, from <https://s3.amazonaws.com/actuonix/Actuonix+L12+Datasheet.pdf>

“SG90 Micro Servo.” Adafruit Industries LLC. Retrieved November 21, 2020, from <https://datasheetpdf.com/pdf-file/791970/TowerPro/SG90/1>

Power

“AA size/ LR6C Universal Power.” FDK America, Inc. Retrieved November 21, 2020, from <https://media.digikey.com/pdf/Data%20Sheets/FDK/LR6C.pdf>

“Polymer Li-ion Rechargeable Battery.” DATAPOWER. Retrieved November 21, 2020, from <https://cdn.sparkfun.com/datasheets/Prototyping/spe-00-502535-400mah-en-1.0ver.pdf>

“FONKEN Power Supply 5V 1A Universal Portable Adapter USB Plug.” FONKEN. http://www.fonken.net/fonken.net/product/index.php?lang=en&metid=56&pseudo_jump=1&pcok=pc

DC Power Regulators

“TPS6216x 3-V to 17-V, 1-A Step-Down Converters with DCS-Control™.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tps62160.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320377783>

“TPS82150 17-V Input 1-A Step-Down Converter MicroSiP™ Module with Integrated Inductor.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tps82150.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320469026>

“TLV1117 Adjustable and Fixed Low-Dropout Voltage Regulator.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tlv1117.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320516116>

“LTC3553 Micropower USB Power Manager With Li-Ion Charger, LDO and Buck Regulator.” Linear Technology. Retrieved November 24, 2020, from <https://www.analog.com/media/en/technical-documentation/data-sheets/3553fc.pdf>

“TLV755P 500-mA, Low IQ, Small Size, Low Dropout Regulator.” Texas Instruments. Retrieved November 24, 2020, from <https://www.ti.com/lit/ds/symlink/tlv755p.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320665019>

“TPS6303x High Efficiency Single Inductor Buck-Boost Converter With 1-A Switches.” Texas Instruments. Retrieved November 24, 2020, from <https://www.ti.com/lit/ds/symlink/tps63030.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607278113893>

“TPS6300x High-Efficient Single Inductor Buck-Boost Converter With 1.8-A Switches.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tps63001.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320768635>

“LMR62014 SIMPLE SWITCHER® 20Vout, 1.4A Step-Up Voltage Regulator in SOT-23.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/lmr62014.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607293959524>

“TPS81256 3-W, High Efficiency Step-Up Converter In MicroSiP™ Packaging.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tps81256.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607298877106>

“TPS6125x 3.5-MHz High Efficiency Step-Up Converter In Chip Scale Packaging.” Texas Instruments. Retrieved November 25, 2020, from <https://www.ti.com/lit/ds/symlink/tps61256.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607320906491>

Battery Chargers

“MCP73831/2 Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers.” Microchip. Retrieved November 23, 2020, from https://cdn.sparkfun.com/assets/learn_tutorials/6/9/5/MCP738312.pdf

“bq2423x USB-Friendly Lithium-Ion Battery Charger And Power-Path Management IC.” Texas Instruments. Retrieved November 28, 2020, from <https://www.ti.com/lit/ds/symlink/bq24230.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607321109527>

“bq24195 I²C Controlled 2.5-A /4.5-A Single Cell USB/Adapter Charger with 5.1 V at 1 A /5.1 V at 2.1 A Synchronous Boost Operation.” Texas Instruments. Retrieved November 28, 2020, from <https://www.ti.com/lit/ds/symlink/bq24195l.pdf?HQS=TI-null-null-digikeymode-df-pf-null-ww&ts=1607321161030>

“LTC3553 Micropower USB Power Manager With Li-Ion Charger, LDO and Buck Regulator.” Linear Technology. Retrieved November 28, 2020, from <https://www.analog.com/media/en/technical-documentation/data-sheets/3553fc.pdf>

Transistors

“2N4400 & 2N4401 Silicon NPN Transistor General Purpose TO92 Type Package.” NTE Electronics, Inc. Retrieved November 25, 2020, from https://www.nteinc.com/specs/original/2N4400_01.pdf

“ZXTP5240F 40V PNP LOW SATURATION TRANSISTOR IN SOT23.” Diodes Incorporated. Retrieved December 7, 2020, from <https://www.diodes.com/assets/Datasheets/ZXTP5240F.pdf>

“NSS40201LT1G, NSV40201LT1G Low VCE(sat) Transistor, NPN, 40 V, 2.0 A.” ON Semiconductor. Retrieved November 29, 2020, from <https://www.onsemi.com/pub/Collateral/NSS40201L-D.PDF>