Farming Assistant & Solar Tracker

F.A.S.T.

Group A

Spring 2020 – Fall 2020

Nicole Andrade	Computer Engineer
Christopher Badolato	Computer Engineer
Jan G. Iglesias Morales	Computer Engineer
Savannah Irvin	Electrical Engineer

Table of Contents

Table of Figures

Table of Tables

Table of Equations

1 Executive Summary

Modern citizens that use agriculture as their main source of income such as, large landowners, farmers, or cattle ranchers find it pertinent, or enjoy, gathering information about the weather. It is required to help predict the outcome of their financial future. In modern times people are busy and want this information from multiple sources as fast as possible with minimal hassle, whether that be on their phones, tablets, or computers. Contemporary farming and ranching require large yields for the ever-expanding population and to help their businesses grow for themselves and their employees. So, it is essential that people take advantage of whatever possible relevant technologies we can to advance the growth of plants, provide local area view of weather, or anything that can help reduce agricultural lost due to weather. With modern weather stations, it is possible to send real time weather data anywhere we want to in the blink of an eye.

One of the main disadvantages of modern weather stations is they only have one device gathering data from the environment. Meaning a large landowner would need to purchase multiple weather stations at a very high cost to place multiple weather stations throughout their properties. An important asset of the Farming Assistance & Solar Tracking (F.A.S.T.) device is the star networking capabilities that provide data from a large distance but is all viewable within one single android application. This data is provided to the application when connected to Bluetooth via the master device. Another disadvantage of modern weather station is they need prolonged external charging. The F.A.S.T. device is self-charging with a rotational solar panel. The main device has the ability to track where the most energy is being provided from, and in turn, rotate the solar panel to charge the battery as quickly and efficiently as possible.

The F.A.S.T. device is good alternative to other types of weather stations, especially if intended to be used across a large area of land. The device contains a plethora of sensors each providing a useful parameter for the user in their goal of maximizing their understand of their field as well as minimizing time spent on gathering data. The F.A.S.T. devices allow the user to have access to historical data as well a live data to make choices that are founded on previously recorded information about future crop yields. The F.A.S.T. device allows a farmer to easily monitor their fields at a moment’s notice with minimal cost.

2 Project Description

Making use of modern technology is important in all professions. The Farming Assistance & Solar Tracking (F.A.S.T.) device is used to provide energy efficiency and modern technology to all types of agriculturalists such as, farmers, large landowners, or ranch owners. The F.A.S.T. device system includes a star network implemented into the devices, so that each device can collect its own data across distances up to 1 mile. The data is then sent back to a main or Master device. Once received the data is be viewable by the user via an android application

2.1 Motivation

The total number of recorded climate disasters between 1980 and 1990 was 149. This number has more than doubled within the last decade with no signs of slowing down. As climate disasters become more frequent and seasonal climate becomes more intense crop loss due to weather is at an all-time high. This factor combined with exponential population growth is the perfect storm for not only food shortages but food deserts. The impact of crop loss is not just limited to food shortages but also effects individual quality of life and sets back sustainable development. From a purely economic standpoint the total loss associated with climate-related disasters has totaled in 14 billion from 1990 to 1980. This number has grown to 100 billion from 2004 to 2014. It is farmers who are absorbing the majority of this monetary loss at 25% of the cost being absorbed directly by those directly farming the crops. This reason this figure is so alarming is due to the lack of resilience of those it is affecting. [1]

Many parts of the world still rely on subsistence farming. Climate related crop losses for individuals who procure their own food supply can be economically, physically and even culturally devastating for them. Maize and wheat, stable crops for both developed and undeveloped nations, has significant crop loss in even 1-2 degrees Celsius increase in temperature. By 2055 smallholder maize farms in both Latin America and Africa will decrease by at least 10%. The methods for adaptation help soothe the symptoms but not cure the disease. One adaptation is the movement of crop location during the season of flood and drought but as the population in these areas increase there will be less available land to migrate to that is favorable in the current season. Another adaptation has been integrating more livestock in agricultural practices, then selling livestock to stay afloat during drought. However, this practice requires more labor and the markets in areas where subsistence or smallholder farmers take place are not always efficient enough to make a substantial difference. [2]

Climate changes' effect on total crop yield has also had an impact on the quality of life. Climate change and agriculture are very interconnected and experience multiple positive feedback loops. One loop has been a shift in worldwide diet trends shifting towards higher consumption in highly processed food as well as food sourced from animals. There has also been a decrease in the overall consumption of fruit and vegetables worldwide. 80% of land available for farming is used for livestock instead of fruits and vegetables. Methane

produced by livestock contributes to the greenhouse gas effect. Fruits and vegetables often require much more water than grain crops but less than livestock. If given the choice between vegetables or livestock most farmers get a more reliable and profitable yield from livestock. In many extremely arid and hot areas, crops that are able to grow are often high in carbs but lack protein and nutrients. [3]

A constant stream of weather data on cultivable land can be analyzed so that farmers can more efficiently plant and harvest crops. Climate change is only likely to get more severe especially with so many positive feedback loops to the agricultural industries so to protect worldwide food shortage or loss of crop biodiversity it is important to take measures to anticipate the weather. In a perfect world the increase in data related to weather near crop sites could influence farmers to plant more water intensive but less reliable crops in certain areas that get more intense rainfall or drought. It could also allow farmers to plan better times to plant crops to maximize yield.

One particular country that is a perfect example of how population growth and climate change have negatively impacted people's quality of life and sustainable agricultural practices is Chad. The climate of Chad is already very harsh and the high caloric crops that can thrive in this area are not the most nutritious. Crops like vegetables and fruits require too much water to be productively grown in this area. The crops that do grow in Chad include millet, cassava, and sorghum. These are all high carbohydrates and low in protein and minerals. These crops are also grown via slash and burn practices. Slash and burn occur when crop land is cleared by slashing and burning down a field of natural vegetation. After the area is cleared the ash from the burned indigenous vegetation acts as a fertilizer for the new crop. However, as the population increases and the need for lumber, firewood and other secondary resources from trees this size of the forest in Chad are dwindling. These farmers are leaving less time between the grow back stages of their cropland. This means less native vegetation is being burned and fertilizing the new crops decreases yield per area. There is also less slash and burn cultivable land available. Another effect of the climate in combination with population growth is the inability to sufficiently nurture livestock, due to the lack of water and area. This means most for the free-range livestock to not produce enough meaning another protein source is taken away from the people.

Another country whose agricultural practices changed with climate change is Afghanistan. While climate change was not the only influence of the poppies becoming a major crop grown in Afghanistan it has played a factor. Poppy flowers are the origin of opium and where about 90% of the world's heroin supply originates from the poppies grown in Afghanistan. Efforts from the United States and the United Kingdom have taken place to limit the growth of poppy in Afghanistan but to no avail. The country has experienced consistent drought which has made poppy one of few financially productive crops to grow in this extremely arid region. While this situation has many factors at play anticipating the rainfall and drought trends could give opportunity to return to wheat and corn which were once staple crops before inconsistent conditions made it unfavorable. [4]

Many nonprofit organizations try to teach sustainable farming practice based on a countries available resource and climate. This “teach a man to fish” mentality may be the key to adapting agriculture to climate change. However, these organizations cannot accurately serve the need of a countries they are trying to teach without proper data. This is where the F.A.S.T. come into play. If a robust, low maintenance, easy use interface weather station could record data in the locations that are trying to tailor their crops to a specific set of needs there is great potential to reduce crop loss. With more accurate climate data nuance in weather pattern can be much more easily detected and available. It will also allow more options to implement crops that can meet the nutritional needs of the people harvesting. [5]

The main disasters in Chad responsible for crop loss are drought and flood. Currently a special rainwater collect technique called Zaï is the main way farmers are combating the flooding and drought. This method however is very labor intensive. If it was possible to be able to anticipate the timeline of flood and drought more efficient farming could take place. For example, in times of drought high calorie low nutrients could be grown as livestock feed and in times where Zaï and flood they could grow more water intensive nutritive crops as well as nurture their livestock and then slaughter in times of drought. [6]

Overall, it is clear between climate change and population growth food shortage have the potential to become a more relevant issue. The F.A.S.T. aims to collect data for cultivable areas such that new rainfall, drought, temperature, humidity, and light intensity trends can be anticipated to lead to more efficient and reliable farming.

2.2 Function

Originally the overall function of the device is to provide climate data at different farming locations however the use case has expanded to anyone trying to maintain large areas of land. The use of the data can be varied but ultimately the F.A.S.T weather station should be able to provide data in a low maintenance and easy user interface fashion.

2.3 Objectives and Goals

The main objective in designing this project was to create a portable solar tracking device that will implement measuring wind speed, temperature and humidity, soil temperature and humidity, and ultraviolet light to assist farmers. We decided to name this device F.A.S.T. which stands for Farming Assistance and Solar Tracking.

Since the objective is to make a device that tracks the sun, one of our goals was to be able to build an apparatus that will hold the solar panel. This apparatus was mostly 3D printed. The goal was to make this device sturdy enough to be able to hold solar panels but still maintain portability.

Since our device is a solar tracker, one of the decisions we had to make was the way our tracker should be able to rotate. There are two ways solar trackers rotate, it is either in a single-axis rotation or a dual-axis rotation. Single axis trackers rotate on a single point. The advantage of having a single-axis tracker is that it is lower in cost and is more reliable than dual-axis. However, the disadvantage is that it has lower energy output during sunny

conditions. The advantage for having a dual axis is that is more flexible. Two other advantages are that it has more energy output during sunny days and will be more accurate. The disadvantage is that it is less reliable. Since one of our goals was to achieve a more accurate tracker as well as maximum energy output, we decided on dual axis-rotation. Because our device will be designed to be a dual-axis tracker, this means that it will rotate on both X and Y axes. This made our device move in a more circular path and was less retrained. To implement this, we attached multiple servos to our device. We picked servos strong enough to hold the weight of our device.

One of our goals was to attach multiple sensors to this device that will enhance its effectiveness and overall functionality. One of the sensors our device has is a sensor that measures the temperature and humidity. Farmers need to be able to know the weather outside and how humid it is going to be. Our goal was to be able to make this simple weather station easily accessible since a lot of farmers activities consist of being outdoors. Knowing the weather beforehand helps better prepare the farmers.

Another sensor attached is a sensor that measures soil temperature and humidity. This will assist farmer in knowing the conditions of their soil. Plant growth is affected by soil temperature and different crops require different soil temperature and humidity to thrive. A crop can experience stress when temperature is high and soil moisture is low. Knowing the condition of their soil will help make the farmers' job easier.

Our goal was to make our device as user friendly as possible for the farmers to use. Originally we planned on including an LCD display on the device that will output certain data coming from all the different sensors in a clean and easy to understand format. Another our device was made more user friendly was the ability to access different data and send different commands to the device through an application. This software application can be accessed through a computer and on a smart phone. The application displays information regarding the different data all our sensors are getting such as the day's temperature, humidity and more. By making data available through smart phones, farmers will not have to walk all the way to where the device is located. This will make it more convenient for farmers. This is especially helpful when weather conditions are not great.

We originally wanted the F.A.S.T to be a sustainable device. Since the device will be designed to be used outdoors, one of our goals was to ensure that every component that must not be exposed to certain outdoor elements be protected. It is universally known that electronics do not mix well with water. These components include the printed circuit board, the wires, and some of the sensors we will need for the device. The team can achieve this by enclosing these components in a protective weatherproof case. The team must also ensure that the stand to which these components are placed be strong enough to avoid major falls. If the stand falls, we also risk damaging our component by them hitting the ground. The goal is to ensure that our device is robustly designed. We were not able to implement this waterproof design.

Having a safe to use device was also very important. Our goal was to be able to have our device safe for our users. To achieve this goal, we must ensure that everything is properly placed. This includes, but is not limited to, ensuring all connections are properly grounded, there are no loose wires, all components are properly placed, and components that are not

weatherproof are enclosed. Another thing we must ensure as a team is that there are no sharp edges on our device that could potentially cut the user. Overall the device was safe.

Another very important aspect to consider is time. Since the team is only given two semesters to complete this design project, we must put a lot of effort into making sure that everything will be done in time. Our goal is to be able to get this project designed, implemented, and tested before the due date in December 2020. We must make a schedule that includes the tasks and the date to which these certain tasks need to be done. It is important to make sure that certain components are ordered in time and to always keep in mind where components come from. Printed circuit boards that are made oversees will be cheaper, but this also means that shipping time will be longer when compared to ones that are made locally.

This project is not sponsored, therefore the budget for our project must come directly from the members' pockets. Our goal was to be able to stay within our budget of five hundred dollars. Since we are working under a budget, we were not always be able to choose the latest and greatest when it comes to deciding which components to buy. The team ensured to spend time doing research and comparing different components. The team considered cost and quality.

In order to have a successful project, great teamwork is crucial. As a team we kept everyone accountable and constantly meet to discuss everyone's progress. We kept each other motivated. With that said, another one of our goals was to stand by the values which all four of us agreed on during our Senior Design Bootcamp. These include, Respect, Communication, Empathy, Teamwork, and Do What You Say You Would Do. The team conducted weekly meetings and consistent communication. Because of the recent outbreak of the Corona Virus and its impact, the campus will be closed for rest of the Spring semester. Social distancing is also important and has been recommended. Because of this unexpected obstacle, the team had to conduct meetings online. Our goal was to ensure that this obstacle will not slow down our project or get behind on any of the due dates. As a team we ensured to consistently communicate despite the circumstances.

2.4 Requirements Specification

To create a weather station that could provide data that has the capability to change agricultural practices rigid specifications was required. Mechanisms of the F.A.S.T including user interface, the microcontroller used, sensors, and the power system all needed to meet specific specification to deliver a product to the end user that meets the goal of being easy to use but at the same time can be feasibly engineered.

The user interface of the F.A.S.T is a key component to the overall ease of use for the consumer. It was important that the product could send data in real time to an android app in intervals of at least 5 minutes. On the engineering side the app had to be programmable in either C# or java. The product also had to be able to alert user via the app if any abnormal reading occurs. These abnormal reading may mean either the sensors need to be calibrated or disaster is imminent. Another key specification was the ability of the application to

record historical data and present it to the user. This was a key feature as it can be used to predict trends in climate that can be used for growing season. The application must also be able to poll the product and request current reading in real time. However we did not come up with a data summary program to summarize historical results.

It was essential that the microcontroller used for this product meets all the specifications required as it will be the key component to integrating all the components. The microcontroller for this product had a 12C address width of 7 bits or greater to ensure the processing of the amount of data required. The microcontroller also was low voltage as an independent power supply was also a goal for this project. It also had low power draw for the same reason. Bluetooth 5.2 with Mesh Network Capability was be ideal as farmers may utilize multiple stations within their crop land to create a more accurate climate profile. SPI support was also a requirement for the microcontroller that will be used in this project.

Sensors were the fundamental components for data retrieval for the F.A.S.T. It is important that all the sensors listed integrated with the microcontroller. A lux sensor was used to measure light intensity and was able to measure in the range of .1 to 30,000 lux. The air humidity senor was able to measure relative humidity from 10% to 90% accurately. The relative soil humidity was able to measure from around 10% to 90% relative humidity. Wind speed was able to be detected between 0 and 50 m/s. Along with lux sensor there was also be a UV light intensity with a range of 0 to 12 mW/cm^2 . Temperature sensor was able to measure between -40C to 125C.

The physical structure was able to support the weight of the photovoltaic module as well as house and components from the elements. The servo motors that were used to support the solar tracking mechanism had a specified torque of at least 21 to 25 kg/cm each. The physical structure also needed to be under 30 pounds.

The power system for the F.A.S.T. needed to be 100% independent and not rely on any grid power. The solar panel used to generate power had to have a max wattage of around 20 Watts. The solar panel also had an ideal condition efficiency that is at least 10%. The battery capacity powered around 24 hours of microcontroller run time. The charge controller and voltage regulation of the system were entirely independent and had at least 85% efficiency.

2.5 House of Quality

The house of quality was a pictorial representation of how the customer and engineering design specification need to compromise to create a final project. Since this project was not sponsored by a specific customer, we needed to use similar products and our own motivation to come up with the customer specifications. However, in most design scenarios the customer specifications will be outlined by a company and it is the job of the engineering team to factor in all the engineering requirements how those requirements will affect the customer specification. It also gives the company trying to develop and idea clear tradeoffs that will need to occur to make the project deliverable. A house of quality is a

very helpful diagram that provides guidance and helps a project come to fruition in such a way that the customer requirements and the engineering specifications are both satisfied in a reasonable manner. A house of quality diagram was constructed for this project in order to aid in the design and development on the F.A.S.T. project. The house of quality was designed after days of analyzing the requirements and the needs of this project as well as finding out what relationship these have with each other. The house of quality shown in **Figure 1** is the house of quality that was designed for this project.

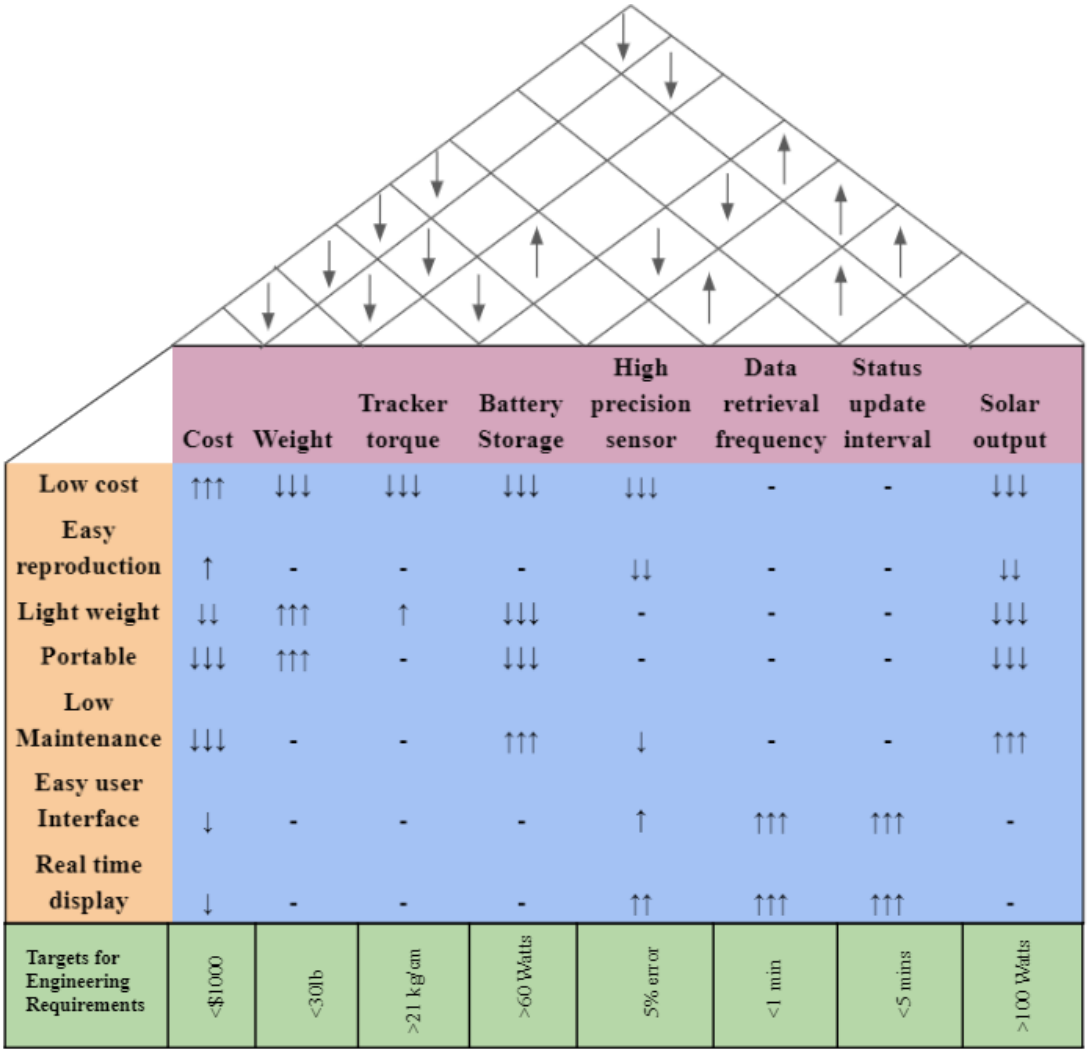

Figure 1. House of Quality

This weather station was targeted to help farmers at many different socio-economic levels, so it was important to keep our cost as low as possible. Another customer specification was that the product is easy to be reproduced so many can be used in a single area to create the most through climate profile. Light weight was also a customer specification because it

needs to be able to be shipped around the world as well as easy transported from area to area. Another key feature for customers was that the weather station is low maintenance and does not require a lot of labor to make it work. Easy user interface was another key component because if the interface was difficult to work with, they will be unable to make sense of the data. Some of the intended customers may not be very familiar with using a complicated weather station, so it was essential that user have a very easy-to-use and intuitive interface to interact with the weather station. Real time display will allow farmers to know what their current conditions are and was the final customer specifications we have listed.

The engineering specifications focused on are the cost, weight, tracker torque, battery storage, high precision sensors, data retrieval frequency, status update interval, and solar output. The house of quality summarized the relationship between the engineering specifications and customer specifications. The low-cost customer specification had a strong positive relationship with the engineering specification cost, no relationship with status update interval and data retrieval frequency, it had a strong negative relationship with everything else. The low-cost customer specification had a negative correlation with almost all engineering requirements because low-cost parts tend to be lower quality then more expensive parts. Most of these engineering requirements relied on high quality sensors and components which tend to not be available in low cost varieties. Easy reproduction had a weak positive relationship with cost as if it is easily reproduced the cost should be manageable, it had negative relationship with high precision sensors as it is difficult to ensure each senor reproduced is reporting values accurately, it also had a negative relationship to solar output as it is more expensive and difficult to get easily reproduceable high efficiency solar panels. Easy reproduction had a neutral relationship with all other engineering specifications. Keep the weather station lightweight had negative relationship with cost as lightweight materials tend to be more expensive, strong positive relationship with weight, weak positive relationship with tracker torque as the less weight the less strong of motor needed, strong negative relationship with battery storage because high storage capacity means more weight in terms of batteries, strong negative relationship with solar output as a larger solar panel means more weight as well, and no other relationship with the other engineering specifications. Portability had similar relationships except it has a stronger negative relationship with cost as the influence on other components besides weight is in play. Low maintenance was one of the most important consumer specifications and it had a strong negative relationship with cost as low maintenance means we need to invest in more robust components that can self-manage, there was a strong positive relationship with battery storage and solar output as higher independently produced and stored power supports the weather stations independence, there was a weak negative relationship with the high precision sensors as they may need to be calibrated, no other notable relationships however. Developing an easy user interface had a weak negative relationship with association to cost, it had a weak positive relationship with high precision sensors as more accurate data makes for a better interface, it had a strong positive relationship with data retrieval frequency and status update interval

as a more constant stream of information makes for a better interface, it did not have a notable relationship with the other engineering specifications however. Real time display had similar results except it had a stronger positive relationship with the high precision sensors as more precise sensors allow for a notable change in real time and provide a much more accurate view of what is going on over a long period of time.

The roof of the house of quality illustrated the relationship between engineering specification so it was known what limiting factors we are working with. Cost had a negative relationship with all other requirements except it has no relationship with status update interval and data retrieval frequency. The reason for why cost had no relationship with update interval and data retrieval frequency is because these two attributes rely solely on software implementation and require no cost to implement in addition to purchasing the sensors themselves. Weight had a negative relationship with tracker torque as the more weight the more torque need, a negative relationship with battery storage as battery storage increased the weight, negative relationship with solar output for similar reasons, and no other substantial relationships. Torque tracking had a negative relationship with battery storage as the battery gets large more torque is required to move the F.A.S.T., there was a positive relationship to high precision sensors and no other notable relationships other than the ones previously mentioned. Battery storage had a negative relationship with data retrieval frequency and status update intervals as these will take up more battery if they are done more frequently, there was a positive relationship with solar output, no other notable relationship beyond what has been mentioned. The reason no other attribute had a relationship with battery storage is because the effect they had on overall power consumption is negligible and will then have very little effect on battery storage. High precision sensor had a positive relationship with data retrieval frequency and solar output. Data retrieval frequency had a positive relationship with status update interval and solar output.

Overall, some engineering and customer specification took precedent over each other. For example, it was essential that the F.A.S.T. is able to power itself independently so battery storage and solar output engineering requirements took precedence over the majority of customer specifications. For instance, a battery that can sustain outdoor use and power the F.A.S.T. unit was a critical requirement and took precedence over minimizing cost. Engineering requirements such as these took priority over some customer specifications. If modifying an engineering requirement negatively impacted the quality of the project in favor of a customer requirement, then the engineering requirement was given precedent over the customer specification. A customer specification that will outweigh may of the engineering specification was making the F.A.S.T. low maintenance and offering easy user interface. The other specification, both customer and engineering, was sacrificed to ensure that the prioritize specifications are adhered to. All these attributes were balanced to ensure that the F.A.S.T. project had an overall successful implementation that did not heavily favor one attribute over another at a cost of overall quality.

3 Research

A significant amount of research was required to implement this project. Research ranging from looking into similar projects or products, to determining, which was the best microcontroller for this project, to even determining the best battery for our use case. Research is a considerable part of any project and this one was no exception. A large portion of time was dedicated to researching all aspects of this project. This section will cover all research that was performed for this project. This section will also include any decisions based on this research as well the rationale for that decision.

3.1 Similar Products & Projects

This section discusses similar product and projects related to the F.A.S.T. devices. They helped give us some ideas into how to design our own product. There are some similar existing technologies that assisted us when deciding parts, equipment, or design. Researching similar projects gave us additional insight into common pitfalls that were encountered when developing similar projects and what solutions, if any, where applied to these pitfalls. Studying similar products also provided us with a ballpark estimate into what the overall cost per unit should look like along with overall development time.

3.1.1 Arduino Weather Station

This example below in **Figure 2** of a simple similar weather station product using Arduino Uno Rev3. The Arduino Weather Station uses a DHT11 for humidity and temperature sensing. A BPM185 sensor for barometric pressure. A rainfall detection module, CP316 rain sensor module. Lastly, a YL-69 moisture sensor. Each of the devices in this product use analog to digital conversions to get the data from the sensor modules. [7]

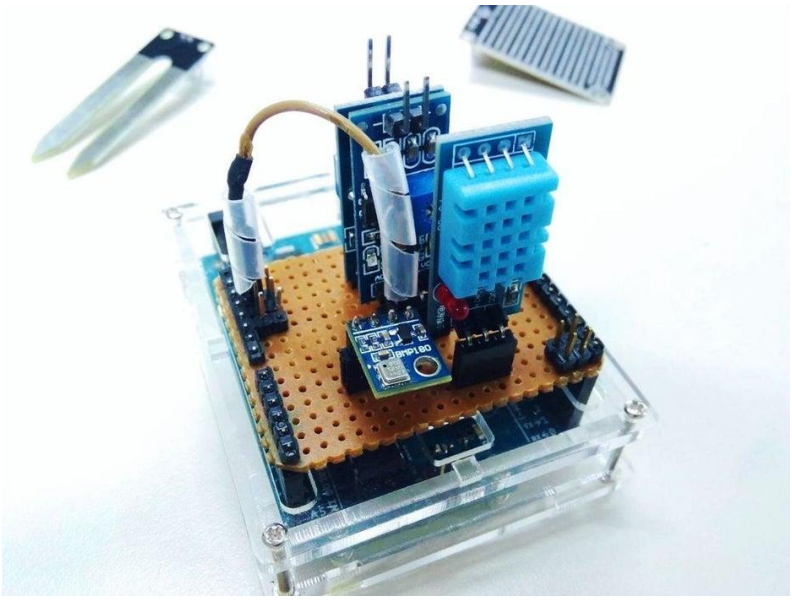

Figure 2. Simple Arduino weather station

3.1.2 Ambient Weather station WS-2902A

One similar product is the Ambient Weather though this product is not solar powered, this weather module has some similar feature in terms of its user interface and a few of its sensing modules. The weather module is shown below in **Figure 3**. The WS-2902A has sensing capabilities such as wind speed, wind direction, rainfall, UV, solar radiation, barometric pressure, temperature, humidity, dew point, heat index, and windchill. While we intended to have a few of these sensors, not all are possible. The user interface for the phone shown below in **Figure 3** is an excellent example of what our product intended to show via Bluetooth connection. This device is similar as it shows real time data to view on an android application. The WS-2902A has the capability to connect to the ambient weather website to upload real time data to the internet. As well the device can connect to multiple devices including, pc tablet or smart home. [8]

Figure 3. Ambient Weather Station WS-2902A

3.1.3 Solar Trackers

There are two types of racking systems for solar panels, there is the fixed tilt and the tracker. A fixed tilt system is having the photovoltaic panels at a fixed tilt orientation while the tracking technology is the ability to rotate photovoltaic panels toward the sun.

The fixed tilt system requires no extra components such as electrical motors or gears. This makes this system less susceptible to breaking as it does not require extra components. A fixed tilt system would also be cheaper and easier to maintain because of the absence of these extra components. **Figure 4** shows a fixed tilt system. [9]

Field-Fast™ Solar Mounting System

Figure 4. Fixed Tilt System

The solar tracker system is rising in popularity and there are many benefits in using this type of technology. In comparison to the fixed tilt system, the tracking technology increases the yield. This is because it collects solar energy for the longest period of day by constantly tracking the sun.

There are two main types of solar tracking systems, a single solar tracking system and a dual axis solar tracking system. The single axis tracker is the most common tracking system. Single axis trackers simply mean that there is one axis of rotation. This could mean that the axis is horizontal, which is the most common, tilted or vertical. **Figure 5** below is a table comparing the annual production of fixed tilt and single axis trackers. [10]

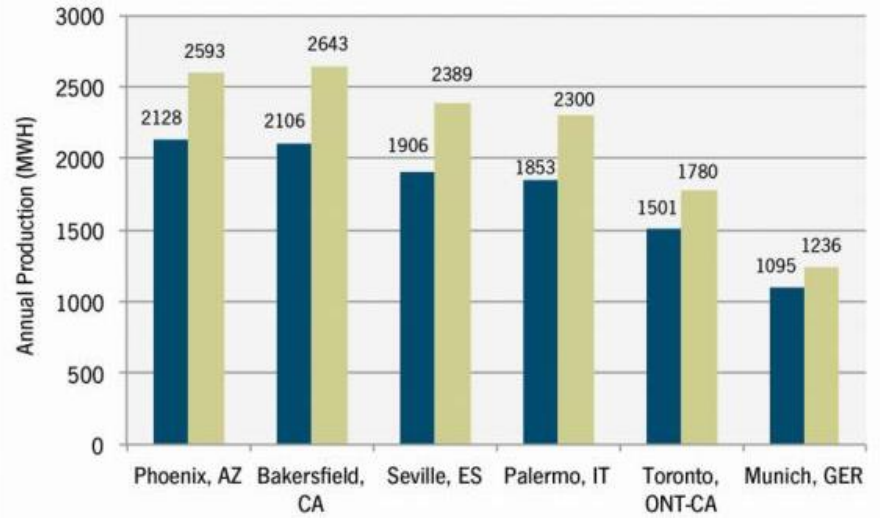

Figure 5. Fixed Tilt vs Single Axis

Dual axis trackers mean that the system moves on two axes. By moving on two axes, the system can have a wide range of position options. This type of tracker can optimize the

collection of solar energy when seasons and sun paths change. With all that said, the tracking system requires extra components such as motors or gears. Having these extra components also means that the tracking system will be more expensive. Also, the presence of these extra components can mean that maintenance will be more expensive as there are more things that could potentially break. From the left, **Figure 6** shows a single axis tracker followed by a dual axis tracker. [11]

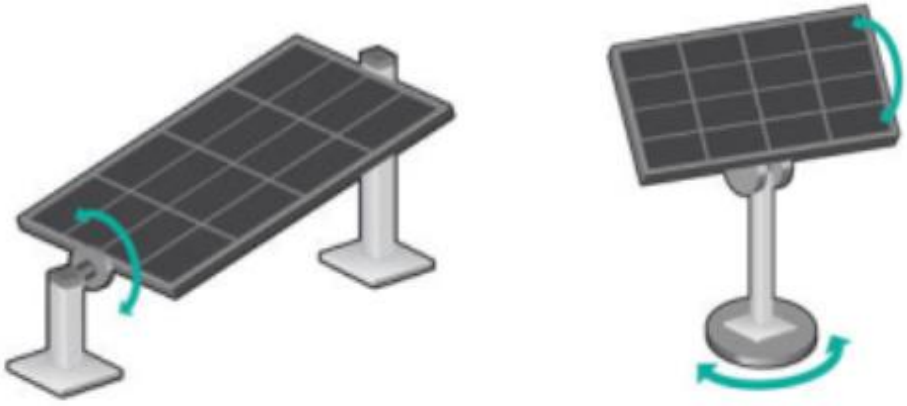

Figure 6. Single Axis (Left) and Dual Axis (Right) Tracking

3.2 Measurements of Interest

Farmers need to monitor many parameters in order to ensure that their fields will stay healthy and productive. Parameters such as temperature, humidity, UV index, and others affect the health of a field as well as the expected yield a farmer can expect from their crops. Having easy access to these previously mentioned parameters will allow a farmer the ability to make faster and more evidence-based decisions when it comes to the management of their fields. Being able to access historical data will also assist a farmer in planning future agricultural endeavors. This section will discuss the measurements that were be made by the F.A.S.T. network along with why these types of measurements were selected by the team.

3.2.1 Temperature

Temperature is one of the most important parameters that one needs to monitor when it comes to any kind of agriculture. Some crops may thrive in very high temperatures while others may wilt in very high temperatures. The same applies to low temperatures. Some plants may be able to endure long stretches of cold, while others can die from only a brief exposure to extreme cold. Knowing the temperature, a field is enduring will allow a farmer to determine which crops are best suited to that field as well as what is the current risk to the crops currently in that field. There are multiple ways to measure temperature in embedded applications such as for this project.

There are temperature sensors available that offer an I2C interface which would made development easier but has some overhead required. There are also temperature sensors are do not require a communication interface and simply output the temperature as a voltage or current. These analog sensors require some conversion to convert into useable data through the means of an ADC and some programming logic. There are also temperature sensors that function through SPI which is a communication interface that functions similarly to I2C. Some of the choices that were considered for the temperature sensor are discussed in this section along with their advantages and disadvantages.

3.2.1.1 TMP36 Temperature Sensor

The TMP36 temperature sensor, shown below in the T0-92 package in **Figure 7**, is a low voltage precision temperature sensor that does not require the use of a communication interface. As shown in the image below, the TMP36 has a very small profile and overall size. The measurement is outputted as a voltage which must then be translated with the use of an ADC to an actual temperature measurement. These kinds of sensors require very little overhead to implement but require additional hardware to process their outputs. These sensors are also very inexpensive which allows for many of these sensors to be implemented with very low overall cost.

Figure 7. Analog Devices TMP36 Temperature Sensor

3.2.1.2 Texas Instruments TMP100

The TMP100 is a temperature sensor that uses the I2C communication protocol to send its output back to a central location. The TMP100 comes in a SOT-23 package and its pinouts is shown below. As shown below, this sensor has a more complicated package than that of the TMP36. This package has a total of 6 pins while the TMP36 has only 3 pins in the T0-92 package. However, as the package shows, the TMP100 implements the I2C communication protocol which makes processing the output very simple. The TMP100 is

inexpensive, but complicated to wire and implement due to its size and pin requirements. **Figure 8** below shows the pinout for the TMP100.

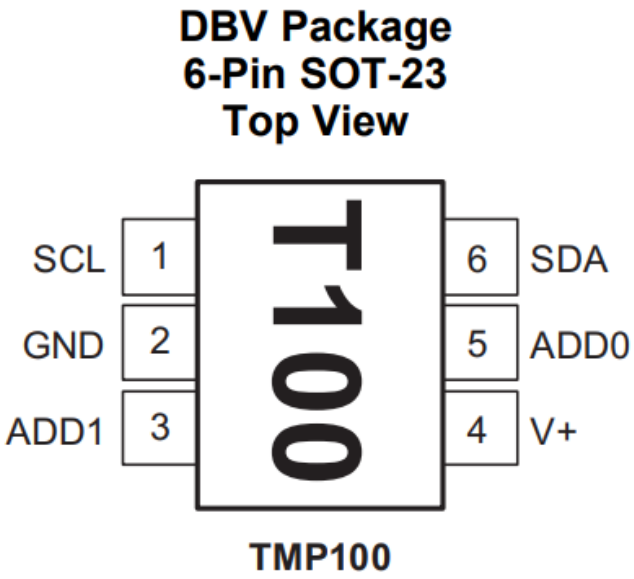

Figure 8. Texas Instruments TMP100 Pinout

3.2.2 UV index

The UV index is measure of the intensity of the ultra-violet radiation that is received from the sun. The higher the UV index at a given area, the more damaging UV radiation is being received at this spot. UV radiation has many damaging properties to humans as well as plants. Exposure to a large amount of UV radiation can cause DNA damage and negatively affect plant growth. Farmers may want to monitor the intensity of the UV radiation that their field receives to investigate if high UV radiation is a possible cause of low plant yields. Another reason to for why UV index is an important quantity to measure is that animals can also be negatively affected by a high UV index. Animals, just like us, can be burned by a constant high intensity UV radiation as well as be subject to negative health effects. A farmer can put one of the F.A.S.T. units in their grazing fields to ensure that the conditions their animals are under are adequate and conductive to a healthy lifestyle.

3.2.2.1 Vishay VEML6070

The VEML6070 is a UV light sensor manufactured by Vishay that uses the I2C protocol to communicate its results. This sensor is a true UV sensing unit that actually measures the UV radiation levels instead of just light intensity to sense what the UV intensity it is receiving. The sensor is I2C as previously stated and will be very easy to implement in this project if selected. The main difficulty associated with using this sensor is its package type. Shown below in **Figure 9**, is a standard VEML6070, which typically comes in surface mount packaging. Surface will make testing difficult when using the actual sensor. There

are breakout boards available for the VEML 6070, but the sensor itself is only available in surface mount which might make soldering much more difficult than standard packages.

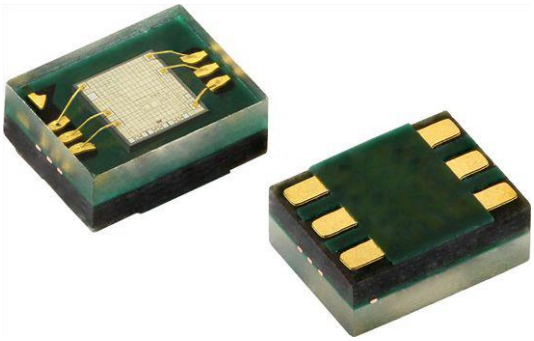

Figure 9. Vishay VEML6070

3.2.2.2 GUVA-S12SD

The GUVA-S12SD is an analog UV sensor that does not require the use of a communication interface such as I2C. This sensor operates by generating a current, through the user of a photodiode, that corresponds with the UV intensity that the sensor is receiving. This sensor is very simple to implement and requires very little overhead. To interpret the measurements from this sensor, one simply has to perform very basic operations on the measured output voltage. The datasheet of this sensor contains information on how to properly convert the voltage to a UV index reading. This sensor simply requires a power source and a connection to ground in order to operate. The downside of this sensor is if the project uses an I2C network to operate a variety of sensors, then this sensor will require its own separate procedure to extract and process data that is different from all the other sensors.

3.2.3 Humidity

Humidity is an important parameter when it comes to plant growth. Plants require water to thrive and humidity is simply how much water there is in the air. However, too much water can also be a bad thing when it comes to agriculture. A farmer may want to monitor the relative humidity in their fields to help them make decisions when it comes to watering and understanding how much water their crops are receiving from the environment. Having a historical record of humidity levels can help the user determine what periods of the year tend be dryer than others to help decide which crops will fare better in certain sections of the year.

3.2.3.1 SHT3x-DIS

The SHT3x-DIS sensor is a humidity and temperature sensor that operates with the use of an I2C interface. This sensor can easily be added to an I2C sensor network with almost no overhead. One of the main benefits of this sensor is the fact that it offers humidity as well as temperature sensing. This single sensor can be used to measure two different quantities

which would normally require a sensor each. This sensor offers very fast start-up and measurement time according to its datasheet. The sensor has a very small 8-pin DFN package which will save on board area. During standard operation, this sensor will return two measurements, one for temperature and the other for humidity. This particular sensor can measure between 0% relative humidity up to 100% relative humidity.

3.2.3.2 DHT-22

The DHT-22 sensor is a humidity and temperature sensor similar to the SHT3x-DIS. The main difference between this sensor the and the SHT3x-DIS is how they communicate their results. The SHT3x-DIS uses the I2C interface to communicate its results back to a central MCU whereas this sensor simple sends it data as a digital output through its data pin. This sensor is a very common sensor used in hobbyist projects and therefore has a large amount of community resources as well as support. This sensor is very easy to implement and requires no additional components.

3.2.3.3 BME280

The BME280 is small 2.5 mm × 2.5 mm × 0.93 mm humidity and pressure sensor. The emergence of home automation control, GPS tracking, and personal fitness give this module multiple modern day uses. This two-in-one sensor also offers temperature sensing but has lower accuracy and is not its intended purpose. This sensor operates using a standard I2C. This sensor offers several power saving features such as multiple sensing modes and low power consumption requirements. This sensor can also be operated in SPI if I2C is not preferred. This sensor is able to be used as both the humidity and the pressure sensor in order to reduce the total number of sensors used in this project by one. Typical uses for this sensor include, context awareness, e.g. skin protection, room change detection, fitness monitoring, home automation control, GPS enhancement and outdoor navigation, and weather forecast.

3.2.4 Wind speed

Wind speed is important factor to monitor for farmers or ranchers. Wind direction and velocity have a significant influence on the growth of plants. Wind increases the turbulence in the atmosphere, in turn, the supply of carbon dioxide for the plants to grow increases or decreases. Plants such as barley and rice are affected even more, wind increases the production of ethylene production as well as nitrogen concentration and decreases the gibberellic acid content of the roots. Strong wind can damage shoots, ruin flowering or seeding, and can cause possible uprooting. A combination of cold and strong wind can cause chilling injuries to all types of crops. Wind affects cattle in similar ways. In cold wind, cattle must be forced to shelter, or they are at risk of being injured by debris. An anemometer is a tool used to measure wind speed and will be used in this project. [12]

3.2.4.1 Adafruit Anemometer

The Adafruit anemometer is an analog sensor use to measure wind speed of the surrounding area. This anemometer functions by converting the captured into a DC voltage that corresponds with a specific wind velocity. The specific conversion is provided in the datasheet. The main drawback of this sensor is the large size of the sensor as well as the power requirements. An image of the sensor is shown below in **Figure 10** which illustrates the large size of the anemometer. The large size of this type of sensor is inherent in how the sensor functions. The cups seen in the image are pushed by the wind and an internal motor spin. The spinning of the motor generates power which then can be converted to a speed. The faster the wind is blowing, the faster the motor turns which in turn causes a larger electric signal. This sensor must be supplied with between 7V to 24V of DC voltage in order to function which is a much larger power requirements than an average sensor we are potentially going to use requires. The data output of this sensor is simple to convert, it simply is a voltage that must be read and digitally converted to windspeed which is a straightforward process.

Figure 10. Adafruit Anemometer

3.2.5 Luminosity

Light intensity or quality refers to the amount of light a plant receives or the brightness a plant is exposed to but does not consider wavelength or color. Meaning, the units are based on visual sensitivity [13]. Measured in lux, which is the illumination that a surface receives one meter away from the light source.

Light is a requirement for plant development though different species of plants differ in requirements for deficiency or excessive light intensities. High lux causes plants to rapidly photosynthesize this can likely reduce the number of hours that the plant must receive each

day. High intensity can scorch leaves leaving them dry and burnt. Excessive light is usually associated with an increase in temperature causing rapid water loss. Chlorophyll content is reduced which causes the plants to photosynthesis more slowly, so greater light intensity can actually reduce the plant growth. Finally, high leaf temperature inactivates some enzymes that change sugars to starch, sugars then accumulate in the plant and the photosynthesis slows down.

Deficient light can cause a plant to fall below its compensation point. Compensation point is the According to Chapman and Carter (1976) Low light intensity causes etiolation or a morphological manifestation of the adverse effect of inadequate light, which in turn causes it to “develop white spindly stems, elongated internodes, leaves that will not fully expand, and a stunted root system”. To measure luminosity, a lux sensor is required. There are many different types of lux sensors available that each operate in a different manner. A few of these options are discussed in the next sections.

3.2.5.1 TSL2591

The TSL2591 is a digital light sensor that measures lux levels in a given environment. The TSL2591 light sensor implements the I2C communication protocol to broadcast its measurements. One of the main advantages of this sensor, is the low power requirements that is incurs. This sensor operates through the user of photodiodes of which it has two. One photodiode is a full spectrum photodiode and the other is an IR photodiode. This sensor is capable of measuring IR intensity as well as full-spectrum and visible light intensities. This sensor also offers a very large range of measurement values up to 88kLux.

3.2.6 Barometric Pressure

Subjecting plants to changes in atmospheric pressure may shorten germination time, improve the growth rate of young plants, and cause more rapid root growth [14]. The effects are species dependent. When increasing the pressure photosynthesis efficiency decreases slowly and nonlinearly as well as irreversibly damages the leaf tissue induced by changes in the chlorophyll fluorescence. The previously discussed BME280 can be used to measure atmospheric pressure as well. There are not very many options for small form barometric pressure sensors besides the BME280, so no sensors will be discussed in this section.

3.2.7 Soil Moisture (Soil Humidity)

Moisture of soil can affect plants growth by altering the biochemistry and reproduction of said plants. It can affect plant height, stem diameter, number of leaves, and the overall biomass. As well, high or low moisture can affect the rate of growth in specific invasive weeds. In an experiment conducted by Aakansha Shadha et al. Plants were given different drought conditions, in the study, experiments were conducted to examine the influence of different soil moistures or water holding capacity (25%, 50%, 75%, or 100% WHC). Plants with 25% WHC had the highest root to shoot ratio while 100% WHC stresses and damages plants by reducing the amount of reactive oxygen. This can have a significant effect on

agricultural systems as some species can survive both under low and high moisture conditions. [15]

3.2.7.1 Adafruit STEMMA Soil Sensor

The Adafruit STEMMA soil sensor is a low-cost soil sensor that implements capacitive measurement to determine the moisture levels of the oil the sensor is inserted into. This sensor has no exposed metal and should be resistant to oxidation due to moisture. This sensor also has a temperature probe that can read the temperature of the soil which is desirable parameter to have readings of. The temperature probe is not very precise, but it is good enough to get an approximate reading of the soil temperature. This sensor implements the I2C communication protocol to transfer its readings and measurements to a central MCU. The I2C implementation will allow for very fast and easy data readings to be made. The I2C protocol will also allow this sensor to be added very easily to an I2C sensor network should this project end up implementing one. The sensor has a solid construction that will allow for the sensor to be inserted safely without risk of damaging the structure. The rugged construction of the sensor can be seen below as well as its small size. This sensor is partially "outdoor-proof" and will be able to sustain prolong periods of time in which it is inserted in the soil. However, area above the soil sensor will need to be prepared for outdoor use. This segment of the sensor can be seen in the image below. The area that must be protected is the area with the exposed traces on the PCB.

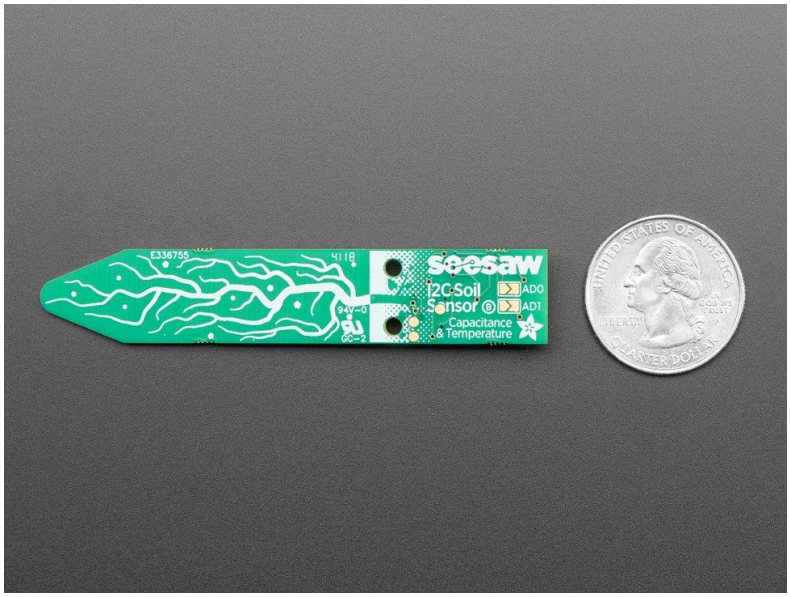

Figure 11. Adafruit STEMMA Soil Sensor

3.2.7.2 SparkFun Soil Moisture Sensor

The SparkFun Soil moisture sensor uses the conductive properties of moist soil to operate. The sensor has two conductive probes that behave as a variable resistor with the moist soil

between them. The higher the water content of the soil the more conductive the soil will be. Using this principle, the soil sensor operates by treating the soil as a variable resistor and simply outputting the voltage across the soil to an output signal. This sensor is completely analog, and the measurement is simply a voltage reading on the output terminal. The higher the voltage, the more humid the soil is. This sensor does not implement any sort of communication protocol and simply has its output as a raw voltage signal which will require further processing on the software side to interpret the readings correctly. Therefore, this sensor cannot be easily added to an existing I2C sensor network and will require its own procedure for measurements that is different from the rest of the sensors.

3.3 Microcontroller

This project required the use of a microcontroller in order to come to fruition. The project required a microcontroller to process the input of a large number of digital and analog sensors. The microcontroller also needed to send and receive data wirelessly to an android device through a mobile app. The project planned to incorporate a mesh network across multiple units which the microcontroller would have needed to implement in some way. In the end, a star network was implemented instead. Each individual unit had a microcontroller capable of doing all the previously mentioned functions. The same model of microcontroller was used in every unit. A capability to handle I2C devices was a necessity for whatever MCU is selected along with being relatively low power to allow for self-sufficient operation. The MCU selected needed to be able to process data quickly to allow the MCU to be in low power mode for the majority of the time. The MCU needed to have a relatively high amount of non-volatile storage to save measurements in case power is lost.

3.3.1 Options for Microcontroller

There was no obvious choice for which microcontroller should be used for this project so this section will compile a list of possible choices. Each microcontroller discussed in this section will contain its own advantages and disadvantages. At the end of this section, the microcontroller that was used for this project will be mentioned along with the reason for the selection. Among the options that will be discussed are microcontrollers such as the ones in the TI MSP430 line due to our exposure to MSP430 MCUs as well as microcontrollers found in Arduino hardware platform due to their ease of use and community support.

3.3.1.1 MSP430FR6989

The MSP430FR6989 immediately jumped out at us when considering options for the MCU to be used in this product. All of the team members were familiar with this specific model of the MSP430 because all members had taken a course in which the lab used this specific MSP430 to conduct the lab work. The MSP430FR6989 is in a 100-pin package which may cause some difficulties in prototyping and testing with a breadboard, but there are mounts

to allow for breadboard testing with a 100-pin package. Another disadvantage of the 100-pin package is that it leads to greater difficulty in soldering the MCU to the PCB.

The MSP430FR6989 has 128KB of FRAM along with 2 KB of SRAM. This MCU has a few dedicated timers that can be used with interrupts. This MCU allows up to 63 I/O pins or up to 83 I/O pins depending on the configuration of the MCU. This MCU consumes a very small amount of energy while in active mode. The active power mode of this MCU draws 3.0V at 210µA running at 1MHz. When running in low power mode, the MCU can draw as little as 3.0V at 0.6µA. This MCU hardware that allows an LCD to be very easily added to the controller. This MCU allows for the user of I2C sensors. A very important advantage that this MCU offers is the existence of the MSP430FR6989 LaunchPad Development Kit. The LaunchPad is an all-in-on dev kit that contains this MCU that allows for very easy breadboard prototyping. All members of this team already very familiar with this development kit due to the previously mentioned lab. This MCU allows for the user of I2C sensors and SPI. The cost of this MCU is $6.42.

3.3.1.2 MSP430G2553

The TI MSP430G2553 is another option that was considered as our choice for the MCU. This MSP430 module is smaller in scale than the previously discussed FR6989 but is still a viable option. The G2553 has up to 16KB of non-volatile Flash as well as 512B of RAM. The MCU has two timers. This microprocessor offers ultra-low power consumption. In active mode, this MCU draws 230µA at 1MHz while receiving 2.2V. In standby mode, this MCU draws 0.5µA at 2.2V. This MCU has an “off” mode which draws only 0.1µA while retaining the contents of RAM. This microcontroller allows the use of SPI and I2C. This MCU is offered in a 32-pin package.

An important advantage of this microcontroller is that it also has a LaunchPad development kit which allows for very fast and easy breadboard prototyping. The development kit will allow for a very large amount of this project to be completed and tested even without PCBs being ordered. This development kit is very inexpensive at $9.99. The MCU itself is also very inexpensive at $1.84 per unit.

3.3.1.3 ATmega328P

ATmega is an 8-bit MCU produced by Atmel. This MCU is relatively low power drawing 1.5mA at 3V while running at 4MHz. The MCU has a “Power-down” mode where it draws only 1µA at 3V. However, the ATmega328P can be ran as fast as 16MHz with a noticeable increase in power draw. This MCU offers 32KB of flash memory along with 1KB of EEPROM and 2KB of SRAM. The MCU offers three counters with varying bit widths ranging from 8-bits to 16-bits. This MCU allows the use of SPI and I2C protocols which are extremely useful and necessary for properly implementing sensors. This MCU offers 23 programmable I/O lines and is rated to operate between -40°C up to 125°C. This MCU comes in a 32-pin package and is rated for automotive use.

One of the main reasons that this MCU is being considered is because this MCU is used in the Arduino Uno Rev3 microcontroller board. The Arduino Uno dev kit will allow for very straight forward development and prototyping. Another very useful advantage for the Arduino Uno using this MCU is the wealth of community resources available for the Arduino and by extension the ATmega328p. There are countless tutorials on YouTube and on Arduino forums on how to make use of all the features available in the Arduino Uno. The Uno is completely open sourced which will provide very useful assistance when it comes time to schematic capture and PCB design because all of the schematic files for the Uno are publicly available along with the PCB files. The Arduino Uno Rev3 is a relatively inexpensive development costing $22.00 per development board. The ATmega328P MCU is inexpensive at $1.90 per unit. The MCU is available in a DIP (Dual In-line Package) style package which will allow for very easy breadboard prototyping without the use of the development board if need be.

3.3.1.4 Atmel AT91SAM3X8E

The Atmel AT91SAM3X8E is a 32-bit ARM processor. This ARM processor offers multiple low power modes which allow the power draw of this processor to got as low as 2.5µA. It is available in 144 pin packages. Specifically, it is available in LQFP (Low profile Quad Flat Package) and TFBGA (Thin profile Fine Pitch Ball Grid Array). The previously mentioned packages are not as straightforward to solder to a PCB as other more common package type which is a slight disadvantage. This processor has a total of 512 KB of available Flash and 96 KB of SRAM. The processor offers a 12-bit ADC along with a 12-bit DAC. A 32-bit timer with 9 total channels is also available in this processor. This ARM processor allows for the use of SPI and I2C protocols. This processor also implements a real time clock which may be useful for taking measurements and time stamping them.

Like the ATmega328P, the Atmel AT91SAM3X8E is also used in an Arduino. The Arduino Due uses the AT91SAM3X8E as its MCU. Because an Arduino product is available with this ARM processor the same advantages that the ATmega328P has for being available in an Arduino also apply to the ARM processor. The Due is a much newer Arduino board than the Uno which means there is less community support available than there is for the Uno. However, due to the popularity of the Arduino boards there is still an overwhelming about of resources available to us with regards to learning how to best make use of the ARM processor if it is chosen as the MCU for this project. The due is slightly expensive for each development board at $38.50 a unit and the ARM processor itself is also more expensive than the other options at a relatively very expensive $9.68 per unit. This ARM processor appears to be meant for use cases that require much more performance than this project demands, and it is the main reason behind the high cost. An advantage of using an ARM processor would be the extremely valuable experience that will be gained in developing a project using an ARM processor due to their prevalence in commercial electronics.

3.3.2 Microcontroller Comparisons

All four options for our microcontroller are going to be compared in this section. Each section will state which microcontroller is the best choice for the given parameter.

3.3.2.1 Power Consumption

All four microcontrollers have varying levels of power consumption, but they all offered a lower power mode which was useful given the self-sufficient nature of our project. The following table, **Table 1,** contains the data about the power consumption of each of the devices

Microcontroller	Active Mode	Low Power Mode
MSP430FR6989	3.0V with 210µA at 1MHz	3.0V with 0.6µA
MSP430G2553	2.2V with 230µA at 1MHz	2.2V with 0.5µA
ATmega328P	3V with 1.5mA at 4MHz	3V with 3µA
Atmel AT91SAM3X8E	1.8V with 21.37mA at 1MHz	1.31mA at 1MHz

Table 1. Microcontroller Power Consumption Comparison

From the previous table, it is apparent that the Atmel AT91SAM3X8E requires much more power to function than the other microcontrollers in the list. Out of the remaining MCUs, the MSP430G2553 is the one that requires the least amount of power to operate followed, by the MSP430FR6989 and lastly by the ATmega328P. These three MCUs consume about the same magnitude of power. The MSP430 MCU consume less than the ATmega, but are still within an order of magnitude of each other.

3.3.2.2 IO pins

IO pins/ports are pins/ports which can be used to send or receive signals through the MCU. Typically, they are referred to as GPIO (General Purpose Input/Output) pins/ports. The GPIO pins can be used as input and in that case will read if a high signal or a low signal is applied and return a 1 for high or a 0 for low. The GPIO pins can be used for output and in that case will send a high signal to represent a logical 1 and will send a low signal to represent a logical 0. This project required the use of a relatively large amount of these GPIO pins. There could have been cases where a simple I/O interaction was needed between the main MCU used in this project and external sensors or external interfaces. A very large number of IO pins may not be necessary but choosing an MCU with too little GPIO pins may lead to a shortage of GPIO pins as the project progresses. Below is a table, **Table 2**, showing how many GPIO pins are available in the available MCUs.

Microcontroller	GPIO pins/ports
MSP430FR6989	83
MSP430G2553	24
ATmega328P	23
Atmel AT91SAM3X8E	103

Table 2. Microcontroller GPIO Pin/Port Comparison

From the previous table, the Atmel AT91SAM3X8E has the largest amount of IO pins at 103 followed by the MSP430FR6989 at 83. The MSP430G2553 and the ATmega328P have relatively the same amount of GPIO at 24 and 23 respectively. This project should require less GPIO pins than any of the MCUs listed so any of the listed choices should suffice in this regard.

3.3.2.3 Cost

An important aspect of this project is to minimize the cost per unit. This project intends to be as low cost as possible, so it is important to minimize cost wherever possible. However, cost should not be minimized to the point where quality begins to suffer. Below is a table, **Table 3**, comparing the cost of one unit of each microcontroller choice.

Microcontroller	Cost per unit
MSP430FR6989	$6.42
MSP430G2553	$1.84
ATmega328P	$1.90
Atmel AT91SAM3X8E	$9.68

Table 3. Microcontroller Cost Per Unit Comparison

The most expensive option is the Atmel AT91SAM3X8E at $9.68 due to its very high performance and large number of features. The least expensive options are the MSP430G2553 and the ATmega328P at around two dollars a unit. Both of these previously mentioned MCUs would be the best choice if based solely on cost.

3.3.2.4 I2C

I2C (Inter-Integrated Circuit) is a synchronous serial computer bus that can operate in a variety of different configurations. I2C tends to be slower than other communication methods, but it is very simple to implement. This project will require the use of I2C extensively because of the large number of sensors that will be incorporated into each unit. Sensors typically use I2C to communicate and transmit their results to a central location, so the selected MCU must have I2C functionality in order for this project to succeed. Luckily, I2C functionality is available in the vast majority of MCUs. All of the possible choices for this project have I2C functionality and therefore any of them would have been an adequate choice for this project.

3.3.2.5 SPI

SPI (Serial Peripheral Interface) is a serial communication interface which is similar to I2C. SPI has similar uses to I2C; however, they are configured differently and have different wiring requirements. SPI requires much more wires than I2C but offers much higher speed and a larger throughput than I2C. Some sensors communicate through SPI instead of I2C, so this project will require that the chosen controller offers SPI functionality in the case that a desired sensor is only available with SPI communication. SPI can be

configured in a number of different ways. Like I2C, SPI is found in the vast majority of MCUs due to its usefulness in embedded application. All of the possible choices for a controller in this project have SPI functionality and therefore any of them would have fulfilled the SPI requirement for this project.

3.3.2.6 UART

UART (Universal Asynchronous Receiver-Transmitter) is an asynchronous serial communication protocol that is very commonly used in embedded applications. UART is commonly used to allow separate devices to communicate and send data to each other in a very simple and low-cost way. UART can be used to allow multiple different devices such as Bluetooth modules, GPS modules, among others to communicate with the MCU. UART is also indispensable when it comes to debugging and testing. UART can be configured to send data to a computer through a terminal application which will be very important in the prototyping phase of this project as well as the debugging phase. UART is very commonly found in most controllers. All of the choices for an MCU in this project have built-in UART and any of them will be a valid choice with regards to the UART requirement.

3.3.2.7 Community Support

A very important factor in deciding which controller to pick for this project is how much community support that controller has. Being able to easily research any issues that may be ran into during development or look up how a specific feature works in a controller would be extremely helpful during prototyping. TI products, especially the MS430 line, have a relatively large amount of community support. TI themselves provide sample programs and a large amount of resources to aid in development. The MSP430FR6989 and the MSP430G2553 both have a large amount of community support some of which comes directly from TI. The ATmega328P and the Atmel AT91SAM3X8E both have an extraordinary amount of community support due to being used in Arduino products. The ATmega328P has more community support than the Atmel AT91SAM3X8E due to its lower cost and the lower difficulty associated with its development. However, the vast majority of the community support comes from the hobbyist community and not from the manufacturer. The ATmega328P seemed to have the largest amount of support available out of all four choices.

3.3.2.8 Memory

Memory is a very important aspect to consider when selecting a controller for a project. There are several different types of memory and each of them have a different function. One of the most important types of memory is program memory. Program memory is the memory in which the program is stored in. Program memory limits how long the program that runs on the controller can be. The more program memory a controller has available to it the bigger and longer the code that can be ran on that program can be. Program memory is nonvolatile which is to say that the memory stored in program memory is not lost if power is lost. A second type of memory that must be considered is RAM. RAM (Random

Access Memory) is a type of memory that is used for temporary data in a program's execution. RAM has an effect on how complex a program can be and how much freedom it allows for dynamic memory allocation. Both of these memory types are very important when it comes to selecting an MCU. However, there is a point of diminish returns where getting a larger memory size costs more than it is worth. Below is a table, **Table 4,** with the RAM and program size available for all of the possible MCU choices.

Microcontroller	Program Memory Size	RAM/SRAM Size
MSP430FR6989	128kB	2kB
MSP430G2553	16kB	0.5kB
ATmega328P	32kB	2kB
Atmel AT91SAM3X8E	512kB	96kB

Table 4. Microcontroller Memory Comparison

Based on the previous table, the MSP430FR6989 and the Atmel AT91SAM3X8E have the most memory overall by a significant margin. However, this project will most likely not use that much memory. The MSP430G2553 and the ATmega328P should have enough memory for our needs.

3.3.3 Selected MCU : ATmega328P

The ATmega328P was chosen for this project. This MCU was chosen for a variety of reasons. The ATmega328P contains all of the communication protocols that may be used in this project. This specific MCU is very low-cost and has a relatively low power consumption. There is an almost overwhelming amount of community resources available for this product as well as a wealth of public knowledge on its capabilities. One of the main reasons that made this MCU so attractive is its inclusion in the Arduino UNO development kit. The UNO will allow for extremely rapid development and prototyping. The UNO has publicly available schematics which will be of benefit when schematic capture and PCB design needs to be performed for this project. The ATmega328P is available in several different packages. It is available in a 28-pin PDIP, a 32-lead TQFP, a 28-pad QFN/MLF, and a 32-pad QFN/MLF. The PDIP package will facilitate breadboard testing significantly when compared to the other form factors.

3.4 Wireless Communication

This project relied extensively on wireless communication. Each F.A.S.T. unit needs to communicate with other units to facilitate the mesh network capability of the project. The main unit in the star network needed to communicate with the user in some form. The unit-to-unit communication method needed to allow for long range communication due to nature of the project. Units will be spaced out at least several dozen meters from each other and the communication method selected needed to be able to tolerate such distances. The communication method between the units needed also to be as low powered as possible to ensure that the units can be self-powered. The communication method between the main

unit and the user did not have to be very low powered or have long range capabilities because the user must be required to be within close proximity to the main unit to interact with the F.A.S.T. mesh network as well as monitor the data gathered.

3.4.1 Options for Wireless Communication

There are several technologies that which could have been used for implementing wireless communication. Each of these technologies had their own sets of benefits and drawbacks. For this project, the team decided to consider Wi-Fi, Bluetooth, and LoRa for our wireless communication methods. This section will discuss of each of these technologies as well as which communication type they are best suited for. Afterwards, the technology selected for each kind of wireless communication will be stated as well as the reasoning behind this selection.

3.4.1.1 Wi-Fi

Wi-Fi is a type of wireless networking technology. Wi-Fi has been in use since 1999 to create wireless local area networks to facilitate internet access. The current generation of Wi-Fi is known as Wi-Fi 6 (802.11ax) which as a maximum performance in the gigabit range. Wi-Fi modules are available for embedded applications. Wi-Fi can be used for short range communication and medium range communication. Wi-Fi can transmit a large amount of data at once. However, Wi-Fi's performance suffers greatly if line of sight is lost and there are obstructions between the receiver and the transmitter. Obstructions such as walls, rain, and crops would greatly affect maximum range of this technology. Wi-Fi is not low powered and depending on the range required the power requirements go up significantly. There are many resources available for Wi-Fi implementations in embedded applications. Wi-Fi also has been successfully used to implement mesh networking and community resources are available that showcase these implementations.

3.4.1.2 Bluetooth

Bluetooth is a short-range wireless communication technology that is commonly found in mobile devices. Bluetooth is a wireless technology standard that is used to send and transmit data between a receiver and transmitter that within close proximity of each other. Bluetooth networking is typically used to implement PANs (Personal Area Networks) which are simply networks that span a very small area. This technology is very commonly used in smart phones as well as laptops, computers, media centers, and general smart devices. The current Bluetooth standard is known as Bluetooth 5.2 which has built in mesh networking capabilities. Bluetooth is relatively low power, but it has a drastic range limitation. Bluetooth is typically limited to around 10 meters. However, Bluetooth can be made to transmit up to a much larger range, but it has a much larger power requirement when it does. Bluetooth has a large amount of community support and many embedded modules are available. This technology excels in short-range communication but suffers greatly at longer ranges.

3.4.1.3 LoRa

LoRa (Long Range) is a wireless communication technology that is very low powered as well as long range. LoRa offers up to 10km range while maintaining very low power consumption. LoRa is very frequently used in embedded applications, specifically in applications that have low power requirements. LoRa has been successfully used to implement star networks and there are several community resources available detailing this process. LoRa has also been used to monitor agriculture in commercial farms as well as monitoring water in these same farms. However, LoRa cannot send a large amount of data at once and is better suited to send smaller payloads. This technology has been extremely popular for IoT (Internet of Things) devices due to its low energy cost and long range. This technology is best suited for medium to long range communication with small payload requirements.

3.4.2 Short-Range Communication Selection: Bluetooth

The team decided to select Bluetooth for the short-range communication section of this project due to its relatively low power draw and its large payload capabilities. The short-range communication technology will be used to facilitate communication between the main F.A.S.T. unit and the user. One of the main reasons Bluetooth was selected for the short-range communication is that it must be a technology that the user will be able to easily interface with. Bluetooth is commonly found in smartphones which allowed the user to interface with the F.A.S.T. network through their mobile device. Another reason that Bluetooth was selected was its relatively low power consumption. There are more low power options available, but the short-range communication will not require a large amount of power. The power draw only increases when it is data is being transmitted to the user which happens somewhat infrequently. The convenience granted by using Bluetooth outweighs any power consumption increase that is associated with its use. A user not having to purchase a separate peripheral and being able to directly interface with the main F.A.S.T. unit with a device they already understand and own is invaluable in terms of user friendliness and accessibility.

3.4.3 Long-range Communication Selection: LoRa

LoRa was the long-range technology that was selected by the team for the long-range needs of this project. LoRa was selected for its very low power consumption and its long-range capabilities. This technology has been thoroughly tested in embedded applications with long-range needs that have similar requirements to this project. Another reason that LoRa was selected was the large amount of community support available for this technology. There were many open-sourced projects that use LoRa as their communication method which greatly helped the team gain a practical understanding of how to implement LoRa in the F.A.S.T. network. Although, LoRa cannot send a large amount of data at the same time as other technologies are able to, this capability will not be needed. The data sent

through the long-range system will be minimal and should be small enough to not be limited by the bandwidth constraints of this technology. LoRa is used exclusively to implement the star network and facilitate communication between the units in the F.A.S.T. network. The long-range communication is used to communicate directly to the user and is used as a backend tool.

3.4.4 Options for Wireless Antennas

The selected communication methods required the use of some sort of antenna in order to broadcast their transmissions. This section will discuss the different kinds of antennas that are available along with which antenna is better suited for each of the selected communication methods. The LoRa communication method will fare best if paired with a powerful antenna while the Bluetooth communication method will be able to suffice with a small antenna.

3.4.4.1 PCB Trace Antenna

PCB trace antennas are a type of antenna which is in the actual PCB of a project. The antenna is drawn in the PCB layout in such a way that the signals are broadcasted as they travel through the "antenna trace" section of the PCB. These types of antennas are difficult to implement, but they are very cheap to implement [16]. The main caveat of these types of antennas is that they cannot transmit very far and are usually limited to a few dozen meters at most. This type of antenna would be best suited for the Bluetooth communication portion of this project and most Bluetooth modules come with a PCB trace antenna already included. There are also many resources online which cover how to design a basic PCB trace antenna if this project where to require its own custom antenna for the Bluetooth communication.

3.4.4.2 Wire Monopole Antenna

A simple antenna can be made using a standard piece of wire. This is a very inexpensive solution that is very easy to implement. However, results vary wildly, and precise results are very difficult to attain. There are many online resources showing how to create wire antennas, but many are from non-academic sources with little theory. These types of antennas are good for short to medium range, but they are very fragile and very sensitive to environmental conditions such as line of sight and interference. The length of the wire is dependent on the frequency at which the transmission is going to made. This type of antenna should not be used for this project due to its fragility, relatively limited range, and difficulty in reproduction.

3.4.4.3 External Whip Antenna

Below in **Figure 12** shows and example of the External whip Antenna. External whip antennas are very similar to wire monopole antennas. The main difference between external whip antennas and wire monopole antennas is that whip antennas tend to be encased in a protective housing. Whip antennas are simply wire antennas which are placed in a metal

or plastic protective housing. These antennas must be bought and cannot be made by the team like the wire monopole antennas. The main benefit of these external whip antennas is that they are very reliable and sturdy. The main drawback to these antennas is their size. As shown below, these antennas are of considerable size when compared to other electronics or even other types of antennas. These antennas need to be placed externally in order to get the best reception. These antennas are ideal for medium to long range applications and will be the best choice for use with our LoRa communication tasks.

Figure 12. External Omni-directional 433MHz Whip Antenna

These antennas can be calibrated and configured to extend the range of the LoRa transceiver by several kilometers. This increased range will greatly improve the overall usefulness of the F.A.S.T. network because the users will be able to space the F.A.S.T. units much farther apart and cover much more of their fields with fewer units. One of these antennas will be required for each F.A.S.T. unit in the F.A.S.T. network in order to actually improve the overall range of the system.

3.5 Power System

The use of photovoltaic solar modules was only power source and gave the weather station the ability to be self-sufficient in terms of power supply. The downsides of using solar was the lack of consistency in day to day charge. To account for this, we used a battery with a large capacity than required to power the microcontroller in its most intensive moments. Because this system required a battery along with a solar module to supplement power there was a means to regulate voltage supplied to the battery and microcontroller to ensure these devices were not damaged. A charge controller was also needed to ensure that the battery is not overcharged and, in the case, that it is fully charge then the PV module was able to directly power the weather station. It was very important that the power system was always robust enough to supply power even in the events where the solar module goes multiple day without producing power. There was also power loss associated with the efficiency of the battery, photovoltaic module, charge controller and voltage regulator. This meant that the majority of our calculations regarding total amount of power required to be

supplied were worse case scenario to ensure reliability of the weather station. **Figure 13** outlines the flow diagram of the entire system. [17]

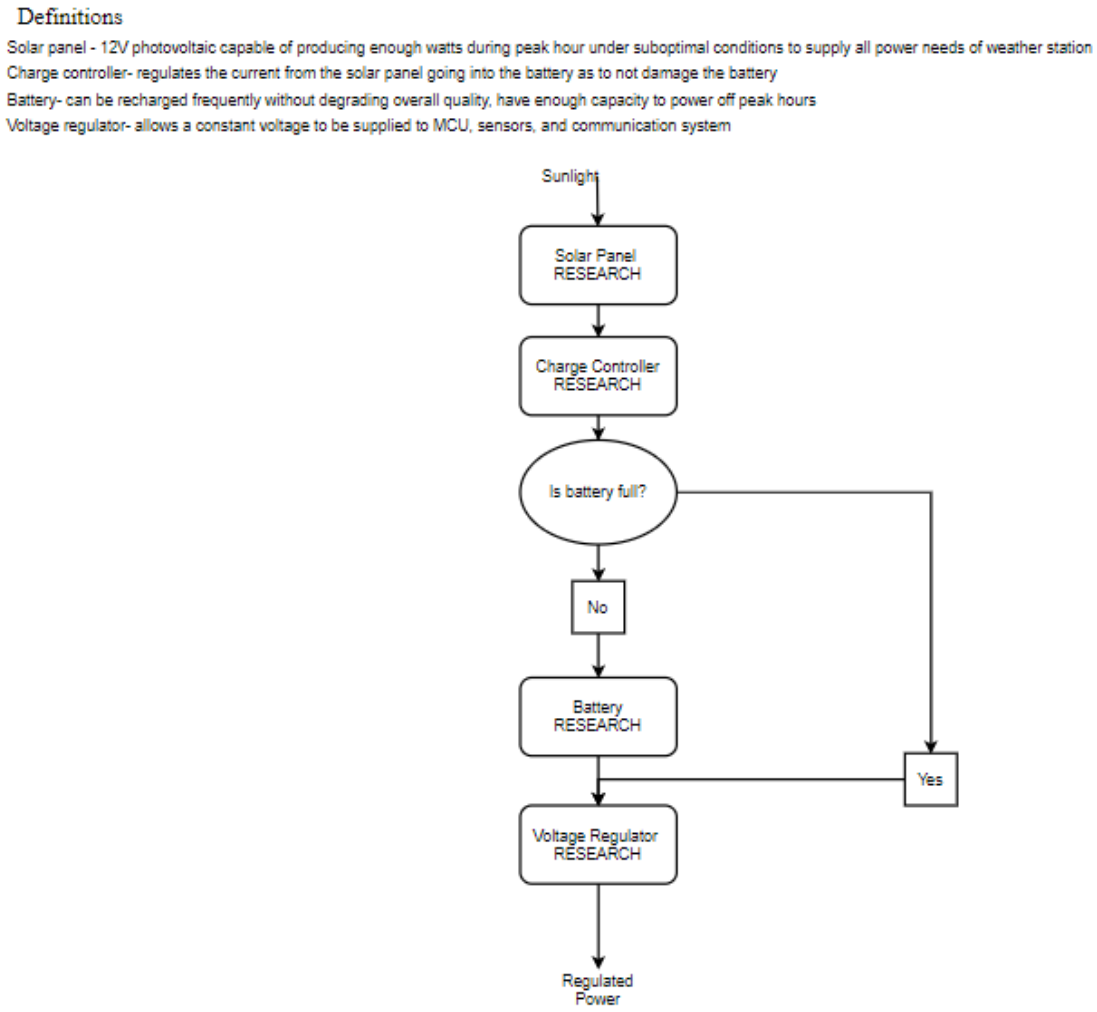

Figure 13. Power System Diagram

3.5.1 Photovoltaic Module

From a purely physical standpoint there were three commercially available types of photovoltaics. These included monocrystalline, polycrystalline, and those that utilize thin film technology. The main drawback that surrounded these different types are varying cost, performance, and reliability. Monocrystalline solar panels are made from one single silicon crystal. This single crystal is produced after a silicon ingot is heated and then is slowly pulled and cools into one monocrystal. A polycrystalline solar panel is less arduous as creating a single crystal. The crystals are still made from a silicon, but not as high quality and they are not meticulously pulled to reduce in a single crystal but instead are just cooled

into a cast where multiple crystal form. Thin film panels produced like the way their name states, with thin layers of photovoltaic material stacked on top of each other.

Price of each respective panel comes from the cost of materials and the ease of replications. Of the three, most monocrystalline panels are more expensive than polycrystalline panel and thin film panels. Thin film produces the best price per panel but not necessarily price for are output.

There is also a large difference in the sustainability of each panel. Since one of our main motivations for this product is to collect data due to the influence of climate change it is important to consider the ability to recycle and reuse of materials

Thin films strengths are mostly in its flexibility, heat resistance, and lack of shadowing effect. Monocrystalline and polycrystalline solar panels efficiency coefficient will be reduced significantly depending on the temperature outdoors and in systems with multiple rows of panels are subject to a reduce in production due to shadowing effect. However, the advantageous of Thin films strengths are mostly in its flexibility, heat resistance, and lack of shadowing effect. Monocrystalline and polycrystalline solar panels efficiency coefficient is reduced significantly depending on the temperature outdoors and in systems with multiple rows of panels are subject to a reduce in production due to shadowing effect. However, the advantageous of Thin Film are not advantageous enough for this project as we will be only implementing on photovoltaic per weather station. The efficiency different between monocrystalline and polycrystalline is around 10% favoring monocrystalline. Because of how the efficient the manufacturing process has become for monocrystalline solar panels it is worth paying a little extra for a monocrystalline power to take advantage of the efficiency gain. However, finding a monocrystalline solar panel that is not considered overkill may be difficult and if that is the case, we will settle for a polycrystalline panel that better suits our needs. [18]

Thin Film were not advantageous enough for this project as we will be only implementing on photovoltaic per weather station. The efficiency different between monocrystalline and polycrystalline is around 10% favoring monocrystalline. Because of how the efficient the manufacturing process has become for monocrystalline solar panels it was worth paying a little extra for a monocrystalline power to take advantage of the efficiency gain. However, finding a monocrystalline solar panel that is not considered overkill was too difficult and we settled for a polycrystalline panel that better suits our needs. [19]

Before figuring out the most efficient photovoltaic in idea conditions we needed to summarize what standard conditions were for the weather station out in the climate. Assuming the station was placed in a variety of hot and cold climate we needed to observe the degradation in efficiency for each panel in different conditions. [20]To figure out the size of monocrystalline panel purely depends on the power needed to supply the microcontroller and sensors. Once this was known we used solar equations to calculate the amount of area per low end efficiency needed to supply the microcontroller.

A direct comparison between a polycrystalline and monocrystalline module to formulate the decision between the two. Both panels are produced by the same company but the monocrystalline photovoltaic is rated at 20W and the polycrystalline is rated at 10W. Both modules have a similar temperature degradation fact at around .5% per degree Celsius. The difference in weight is 10 watts per pound for the polycrystalline and 13.33 watts per pound for the monocrystalline. The polycrystalline cost about 2 dollars per watt and the monocrystalline cost about 1.6 dollar per watt. The current versus the voltage is listed in **Figure 14** and **Figure 15**. Between the price and weight, the monocrystalline is the clear choice and will be used for this project even though the efficiency at different temperatures based on the IV curves on the product specification sheet does not produce much of a difference.

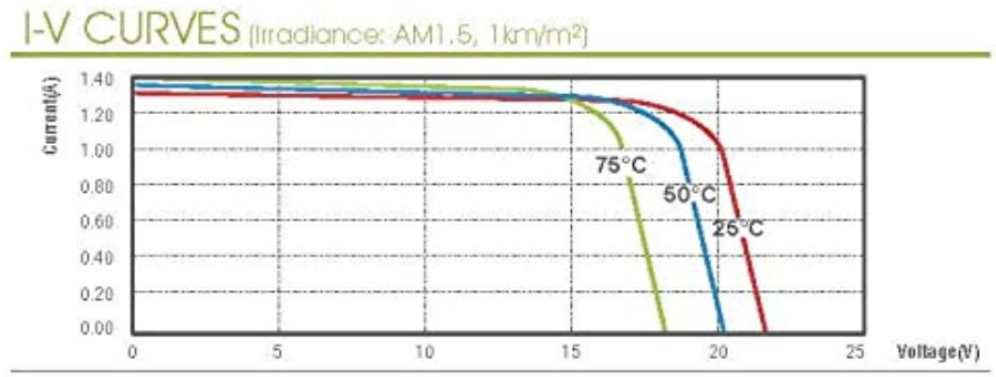

Figure 14. IV Curve for 20W Monocrystalline Photovoltaic Module [21]

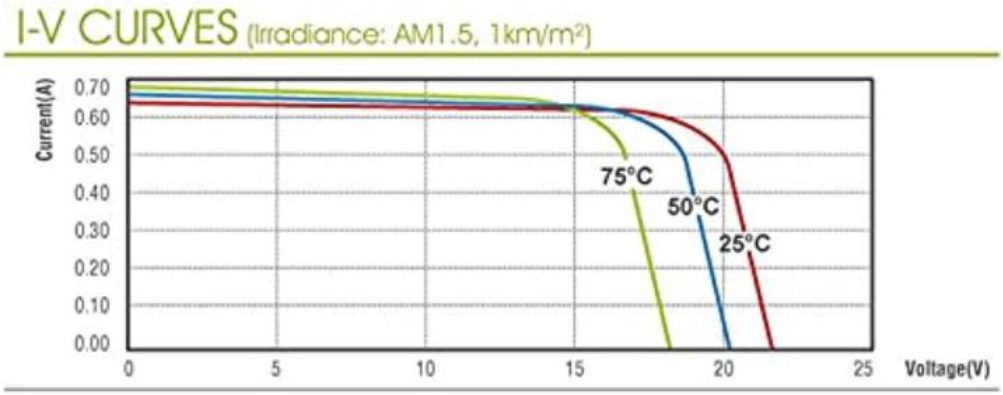

Figure 15. IV curve for 10W Polycrystalline Photovoltaic Module [22]

3.5.2 Battery

According to **Figure 1** it was important for the battery to be the minimum weight it needs to be to adequately power the weather station for at least 24 hours with no sunlight. Assuming we will use the most power intensive **Atmel AT91SAM3X8E** and assuming it will be in active mode for the majority of its duration on we can calculated the amount of power it will use as seen below. We can then use **Equation 1** to calculate the amount of energy it will use in joules. We can then use this figure to estimate that for a 48-hour period needed a battery with a capacity equivalent to 6646.9248 joules or around 3000 for a 24 hour period.

After a battery that met the requirements for supplying the weather station with the necessary joules to last 24 hours without sunlight, we then can explore the possibility of different kinds of batteries. A few different options for batteries would be nickel cadmium, nickel metal hydride, lithium ion, and sealed lead acid batteries. Nickle cadmium batteries are particularly resilient to cold temperature could be advantageous in unique farming scenarios but overall is not a compelling quality. Nickle cadmium batteries are also particularly useful for situations where a higher current output is required. However, nickel cadmium batteries need a safety valve and overall, the benefits do not outweigh the hurdles so nickel cadmium would be an unlikely choice for this application. Nickle metal hydride batteries are similar to nickel cadmium batteries, but they have a higher energy density per weight however for the same reason it is an unlikely choice for this project. [23] Lithium ion batteries are well known for their high energy density, lifespan, and discharge efficiency. They are frequently chosen for solar project because of their life span as with solar systems there is frequent charging and discharging. For most batteries this constant charging and discharging negatively impacts the batteries capacity and efficiency over time. The other battery option that is also used for solar project is lead acid battery. Lead acid batteries are a better choice than nickel batteries for solar project but its only advantage over lithium ion is its cost. Since our project will require a relatively small battery and weight it's a constraint, it makes sense to use a lithium ion battery of lead acid. [24]There are a few lithium ion battery packs specifically meant be integrated with microcontroller that will sufficiently meet our needs. [25]

$$P = I * V$$

Equations 1. Power Calculation

E=VI*H ==> 1.8V * 21.37mhA = 38.466 mW

$$E = P * t$$

3.5.3 Charge Controller

A charge controller is voltage/current regulator specifically for batteries. In power systems that utilize photovoltaics as their main power source they are essential. This is because photovoltaic modules act as a current source and many batteries have limitation on the

current, they can be injected with. A charge controller also helps to makes sure the battery is no inject with power above its capacity which can do a lot of damage to the battery. The right charge controller was specifically tailored to the photovoltaic module chosen and the battery. Some requirements we wanted of charge controller is that it had decent efficiency, so we didn't experience a high amount of conversion losses, it was also important that it had a small form factor. It iwas possible to design a charge controller instead of buying one, but it was advantageous to purchase one based on the complexity of our design.

3.5.4 Voltage Regulator

To create a safe and reliable power system for the weather station we needed multiple voltage regulators. We needed one from battery to microcontroller directly we and we also needed one in the even the battery is fully charge and the photovoltaic module delivers power to the microcontroller directly. Both regulators needed to step the voltage and current to a level that the microcontroller can safely.

There are two types of regulators that could be useful for this project. A linear regulator is mostly used when the voltage source is higher than that of the load. In general, linear regulators are cheap but they are not necessarily very efficient. Another downside to linear regulators is that they do not dissipate power very well and have the potential to overheat. In **Figure 16** it is easy to observe why the power dissipated needs to be minimized or the regulator will overheat. The photovlatic module acts as a current source so a linear egulator is not the ideal choice as it is likely to see the regulator overheat with a solar panel source.

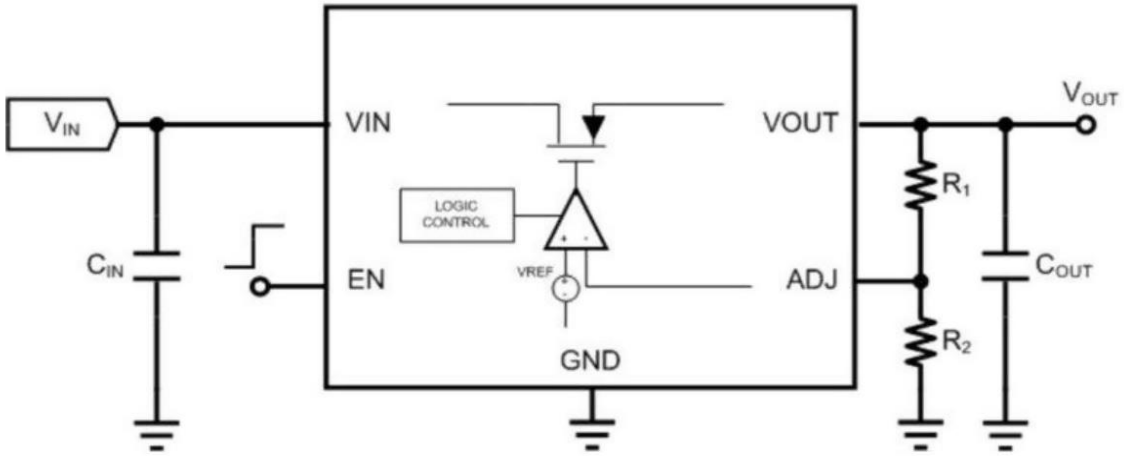

Figure 16. Linear regulator design

The other voltage regular we need to consider is a switch regulator. Switch regulator are used when linear regulators have too high of power dissipation. However, switching generator do not produce a very clean output which is preferred for this design. Because of this it was best to use a switch regulator then followed by a linear regulator so that efficiency is maintained while producing a clean regulated output. [26]

3.5.5 Tracking

Another requirement of our power system specifically related to the photovoltaic module was the ability for it to track solar. While it is known that maximum solar output is produced when the panel is 45 degrees from the sun in some situations it may not be efficient to use power to move the solar panel to its maximum efficiency positions. There are a few different methods to implement a solar tracking device. The most intuitive is "Perturb and Observe" when the photovoltaic module is moved slightly and if the voltage increase then the module is continued to be moved in that direction. Another method is Incremental Conductance and is used in conjunction with perturb and observe but makes sure that when the panel moves towards its favored direction it stop on the most optimal place. Building off the previous describes method is the Parasitic Capacitance method where the capacitance of the modules is factored in. For our specific case since we are only using on photovoltaic module per weather station it is not advantageous to use this method. The final method researched is the Voltage control maximum point tracker. This method assumes the max output is around .75 the amount of the photovoltaic modules open circuit voltage and tracks till the output is around this figure.

For our weather station not all of these methods were advantageous. The Perturb and Observe was appealing because we do have two axis solar tracking, meaning the module can be moves on both the x and y plane. Perturb and Observe would let positive feedback in either direction that we move it to. This also make using incremental Conductance with it will be advantageous to ensure our module does not end up pushed too far in any directions. The Parasitic Capacitance does not appeal as there is only on module and the capacitance will not matter enough to waste energy based of its algorithm. Voltage control maximum point tracker would be advantageous if the module is not exposed to extreme weather conditions but since it is one of our goals to have a working product in extreme weather condition, the max output will change frequently with the weather conditions. [27]

3.6 Display

To make F.A.S.T more user-friendly the team considered adding a display component onto the device which would serve to output data gathered from the different sensors. In this section we will first discuss the three different display modules we were considering. This will be followed by comparing all three ultimately we did not use a display in our final implementation.

3.6.1 Display Choices

There were a few different options of displays to use. In this section we explored three different kinds of popular compatible displays that we could have used to help us determine which one could have been the best fit for our project. Factors such as ease of use, ease of development, cost, and sustainability were considered in the selection of the display.

3.6.1.1 TFT LCD Display

Thin Film Transistor LCDs have a transistor for each pixel on the screen. TFT LCDs are also called active-matrix LCDs and they allow for a faster refresh rate than a regular LCD which results in a smoother motion. TFT displays are much higher quality than regular LCDs but they also use more power than a regular LCD thus making it more expensive to operate.

There are two types of compatible TFT LCD's to choose from, one is a 3.5-inch 480x320 with a touch panel and the other one with the same exact dimensions but without a touch panel. **Figure 17** shows an example of a TFT LCD. [28] This TFT LCD has an input voltage of 2.8 to 3.3V and has a 28-pin module. On Amazon, these modules cost about $20 and some oversees websites they could be bought for about $10.

Figure 17. TFT LCD

These displays tend to be more expensive monetarily as well when covered to other displays. These displays do not withstand prolong direct sunlight well and will need to be protected if deployed in our F.A.S.T. units. These displays are also not as common as other types of displays in the hobbyist community and will, as a consequence, have much less community support than other types of screens. These screens tend to harder to develop for and the code tends to be more complicated.

As for sustainability, a touch panel was not ideal as the environment which the display will be placed was outdoors. Having a touch panel also meant that there were more things that could break. These are some of the things that were considered when choosing a display.

3.6.1.2 White OLED Display

OLED stands for organic light-emitting diode. The organic plastic layers of the OLED are thinner, lighter, and more flexible. OLEDs can produce both light and color from a single diode which means that they do not need a backlight. Each individual pixel could turn on and off. Each pixel has its own color and light. Not needing a backlight results in consuming much less power. Compared to a regular LED, OLEDs are brighter.

The available compatible OLED displays are smaller than the TFT LCDs. One option is a .96-inch with 128x64 pixels and the other one is a .91-inch with 128 x 32 pixels. **Figure 18** shows an example of a white OLED Display. [28]

Figure 18. White OLED Display

The OLED display had a voltage of 3V to 5V and has a 4-pin module. On Amazon, the .96-inch white OLEDs costed about $6 a piece. Some of these displays such as the one pictured above implement the I2C communication interface. Displays that make use of the I2C protocol would have been very easy to add to our project and very straightforward to operate.

OLED displays are not as common as other displays, but they still have a significant amount of community support that would have made development using this display easier. This kind of display is not restricted to ASCII characters and can display images. This display allows for the user to control what is displayed on it on a pixel by pixel basis. These displays tend to be somewhat fragile and do not fare well under direct sunlight so measures to protect the display will need to be taken if selected for use in this project. This display would have been more than adequate to provide information on how the F.A.S.T. unit is operating as well as additional information that the user might be interested it. This display could have been useful for debugging and diagnostic purposes as well.

3.6.1.3 16 x 2 LCD Display

LCD stands for Liquid Crystal Display and is the most common of the three. LCDs are made of two pieces of polarized glass and these glasses contain a liquid crystal material in between. Unlike the OLED, LCD needs a backlight to produce images. LCDs produce very bright images and are suitable for brightly lit environments. LCD's power consumption can be controlled and is based on its backlight settings. With a low backlight setting, an LCD could draw less power than OLED. One LCD display we can use for our project is a 16x2 character LCD Display Module. As the name suggests, it displays 16 characters in 2 rows. They come in either a blue or green background. This LCD is popular to people who are just getting into using an Arduino.

There are other options for sizes other than a 16x2 such as a 20x4, but for the purposes of narrowing our options, we will only consider the 16x2 LCD display. A 20x4 display is also bigger but a smaller option is more ideal for our project. **Figure 19** shows an example of a 16x2 character LCD display.

Figure 19. 16x2 Character LCD Display

This LCD display has a voltage of 5V and has a 16-pin module. On Amazon, these components are cheap and average about $3 a piece. For manufacturability purposes, this type of display is ideal.

LCD displays such as the one pictured above tend to be limited in their output capacity. These LCD screens can only display standard ASCII characters and are limited to two rows of characters each with a maximum length of 16 characters. These displays are not capable of displaying images or large amounts of data. However, these displays could have been immensely useful in debugging and displaying concise operational data such as onboard temperature or solar output. These screens could have been used in the final unit for diagnostic purposes and during development for debugging purposes.

Another advantage of these kinds of displays was that they are extremely common and a favorite of hobbyist. An overwhelming number of projects make use of these displays and for this reason this kind of display had an extreme amount of community support. There are pre-made libraries used to interface an MCU with one of these screens in a very simple and straightforward way.

3.6.2 Display Comparisons

In this section compared all three of the LCD displays. Comparing these four factors help us determine that an LCD was not prudent for the first iteration of the F.A.S.T. The following table, **Table 5**, contains a comparison between the previously mentioned factors.

	TFT Display	OLED Display	LCD Display
Cost	$20	$6	$3
Screen Size	3.5 inches	.96 inch	16 characters x 2 lines
Resolution	320 x 480	128 x 64	5 x 7
Module Size (in)	2.17 x 3.35	2.6 x 2.6	4.3 x 1.4

Table 5. LCD Comparisons

When it comes to cost, the 16 x 2 Character LCD display beats both the TFT and OLED display. However, when it comes to the size and quality of display, the TFT is the most attractive at 3.5 inches and 320 x 480 resolution. If we choose the TFT, we can have a more sophisticated interface because of its vivid colors and clear images. It boasts a higher resolution than both the OLED and LCD display and it is highly customizable. The quality of the 16x2 LCD Character display simply does not compare to the TFT. This display is limited to 16 characters by 2 lines and each character is displayed in a 5 x 7-pixel matrix. Unlike the TFT, the LCD will be less customizable. The OLED display stands in between the TFT and LCD when it comes to cost, size, and resolution.

3.6.3 Selected Display

The display component for F.A.S.T was a requirement, but it was not a top priority. Having to spend less on the display allowed us to invest more on the sensors which was of more importance. Ultimately it was decided to go with no display as the user was able to see the output on their own device.

3.7 Housing Design

An essential part of the project is to design a housing that will be able to fit every component. This includes the sensors and the printed circuit board. Also needed is a housing strong enough and big enough to hold a specific photovoltaic module. Since the device needs to be able to track the sun and the team has decided on a dual axis tracker, the housing must be capable of doing a dual axis rotation. This means that the housing must be able to rotate in both x and y axes.

There are many ways the team could design the housing. Since the group consists of three computer engineers, one electrical engineer and no mechanical engineer, it is vital that enough research is done to figure out what will be best for the device. This section will discuss the different options and possibilities for a housing design. This section will be divided into three parts. In the first part we three different possible materials will be discussed. The next part will discuss the type of manual labor that is going to be involved with all three materials. The final part will discuss the cost and time for each material.

3.7.1 Housing Material

The first element that will be discussed is choosing the material. To pick the right material to build the device the team must do some good research ensuring that the material chosen would meet the device's performance requirements. The team must also ensure that the material is easily accessible. There are many available options out there, however, the options will be narrowed down to three. These three options consist of metal, wood, and plastic. These are among the most popular materials used for building. Since these materials are popular, they are easily accessible. First that will be discussed is metal, then wood, and finally plastic. With each material the team will discuss the physical labor that will be involved, the cost to build with each material since the team is working under a budget and the time it will take to build with each material.

3.7.1.1 Metal

Metal conducts heat and electricity relatively well and they are usually malleable. This material is strong and will last longer than many other materials such as wood. If the housing is correctly built with metal, it can withstand strong weather conditions which is important because device will be used outdoors. Most of the time, metal do not need complex maintenance. Painting metal with durable paint can also boost its life cycle significantly. Another attractive aspect of using metal is that it is a hundred percent recyclable.

3.7.1.1.1 Labor

There are three main categories when working with metal and these are forming, cutting, and joining. To work with metal, need various tools and specific skills will be needed. After forming a design for the housing, the team will mark out the different patterns on the metal. A scriber and a centerpunch are two common ways people mark metal, however a simple black felt tip pen can also be used.

The next step is to cut the metal after it has been marked. Cutting can be tricky and there are many ways we can go about cutting and shaping our metal. The team can either use a hack saw or a power saw. Using a hacksaw requires a lot more physical work than a power saw. A power saw will cut the metal faster, but it would at times be harder to control. It would be easier get more control and precision with a hacksaw. If control and precision are wanted, then the hacksaw would be the better option. However, if saving time and using less physical energy is the priority then the power saw would be a better option. To properly cut the metal the team must ensure using the right blade, for example, a coarse-toot blade would be ideal for heavy cutting.

After cutting the metal, they must be joined. To join the different parts, the metal must be weld. Welding metal is the process of using high heat to melt the parts and this causes fusion when the parts are cooled down. To weld metal, proper equipment is needed. For a beginner, an MIG welder is recommended, and since no one in the team has any experience with welding, this would be the better option. There are also several materials needed to weld properly. These are things such as MIG wires, welding tips, flux, and many more.

3.7.1.1.2 Cost

To build the housing using steel, a ¼ inch by 4 inches by 12 inches plain steel plate would be needed and this would cost about $12 dollars. A couple steel tubes and steel angles for the stand are also needed. A ½ inch by 72 inches plain steel square tube would cost about $13 dollars and a 1 inch by 72 inches plain steel angle would cost about $15. A hacksaw to cut pieces must also be acquired and this would cost about $20. In total, the steel pieces and a hacksaw would cost about $60. To join the steel pieces, they would need to be welded together. The most expensive component would be the welder. An MIG welder costs about $300 which is more than half of the team's budget. An extra $50 would need to be added for additional miscellaneous costs such as the MIG wires. Spending another $300 on a welder is very unlikely for this project. Unless the team could manage to get a cheaper way to weld steel, choosing to work with steel is going to be very costly and most likely out of budget.

3.7.1.1.3 Time

Since no one in the team has any experience with welding, the team must either learn to weld or pay a professional. Learning to weld would require extra time, effort, and material. There is also an option of spending less time on welding and paying a professional, but that entails spending more money. Cutting the metal while using a hacksaw would also take a lot of time especially for the team who have very little experience.

3.7.1.2 Wood

Wood is one of the most popular and most used building materials in the world. It has a low thermal conductivity and relatively high strength. Obviously when it comes to being more environmentally friendly, wood beats metal and plastic. Wood is raw material and is renewable. Like metal, wood can last a long time and using the light laminate can also improve its lifetime. Wood is recyclable as well, but wood that is painted renders it not a good candidate for recycling. There are many types of wood and when it comes to picking the best one for our device, further research must be done. For outdoors, cedar and redwood must be considered as they are resistant to moisture and therefore more ideal for outdoors.

3.7.1.2.1 Labor

Like working with metal, the three categories that are involved with working with wood are cutting, shaping, and joining. After gathering the right amount of wood in the right sizes and thickness, first mark out the different patterns needed on the wood. The next step after marking the wood is to cut it. There are many tools and options for cutting wood. The bigger and more serious options are a lathe and a table saw. These make wood working as easy as possible, but they are also costly. Unless already owned, these options are not ideal. Like with metal, a hacksaw could be used to cut wood. A crosscut saw is also a popular choice and is good for cutting through tough pieces of wood. Cuts can be made efficiently with a crosscut saw. To cut curves, a jigsaw can be used.

The next step is to join the wood pieces together. There are many ways to join wood together and in this section two will be discussed. One way to join wood together is using the Basic Butt method. This just simply means that one piece of wood butts into another.

This method also requires using glue to improve strength. The next one to consider is to drill pocket holes at corner joints. This requires proper planning on where to drill the holes. This method requires drilling and the use of screws to hold everything together. The team must also ensure that everything is properly sanded. For safety purposes when working with metal and wood, it is very important to wear thick gloves to protect hands from the tools and any sharp edges. It is also important to wear safety glasses to protect eyes from any flying debris while cutting.

3.7.1.2.2 Cost

To build the house with wood a couple 2 inches by 4 inches by 4 inches will be needed. These usually are cheap and would cost about $3.50. Some plywood will also be needed, and a decent size would cost about $6. To cut the wood, a crosscut saw can be used which would cost about $10. To join the pieces together some glue and or some screws can be used. These two should cost less than $15 combined. Buying some sandpaper would also be needed and this should cost less than $5. If the team decides to choose wood, altogether it should cost about $45.

3.7.1.2.3 Time

Working with wood would require less time compared to working with metal since welding is not needed. Wood would also be relatively easier to cut than metal. One of the most time-consuming aspect of working with wood is making sure that everything is cut correctly and that joining pieces together will not be an issue.

3.7.1.3 Plastic

The final material is plastic. Building with plastic by means of a 3D printer. One of the most popular type of 3D printing plastic we can use is Polylactic Acid or PLA. It is plant based and is therefore a biodegradable plastic. When considering the environment, it would be the better choice. PLA is one of the easiest materials to print with and is usually what beginners use when first starting to 3D print. Compared to ABS, which will also be discussed, it has a lower printing temperature. It is also generally odorless unlike some type of plastic. There are countless of colors and styles available. A concern about the PLA is its poor resistance to temperature. Acrylonitrile Butadiene Styrene plastic or ABS filament is another popular 3D printer filament. It is a high strength and durable material and will withstand usage and wear. It can also withstand high temperatures. A couple concerns about this material is the heavy warping and the odor it produces. Another one to consider is PTEG and it is a Glycol modified version of Polyethylene Terephthalate. It is most known for being used in water bottles. This material has a good impact resistance but is also prone to wear. This filament is supposed to be clear and less brittle to print with. It is water resistant which is ideal for our device.

Table 6 will compare different all three materials based on strength, stiffness, durability, cost, and whether it is impact resistant, UV resistant and water resistant. The information gathered is from Simplify 3D. [29] The strength of the material is the maximum stress the material can take before breaking. Stiffness measures how difficult it is to bend the material while durability is a combination of heat, fatigue, water, UV, and chemical resistance. Both

stiffness and durability are based on a 1 to 10 rating scale with 10 being the highest. All these properties help determine which would be the best fit for our design. The following, **Table 6**, contains a comparison between all the three plastic types on all the previously mentioned aspects.

	PLA	ABS	PETG
Strength (MPa)	65	40	53
Stiffness	7.5/10	5/10	5/10
Durability	4/10	8/10	8/10
Cost (per kg)	$10 - $40	$10 -$40	$20 - $60
Impact Resistant	No	Yes	No
UV Resistant	No	No	No
Water Resistant	No	No	Yes

Table 6. Filament Comparisons

3.7.1.3.1 Labor

Unlike metal working and wood working, using plastic by means of 3D printing will require very minimal physical work. To start, a 3D model would need to be built. To build a 3D model, a 3D modeling software or a scanner could be used. There are many available options for a 3D software online. After the 3D model is done, the next step is called slicing. What slicing does is divide the model into many different layers. After the model is sliced, it is then fed into the 3D printer. The 3D printer prints the model, and the team must ensure to have enough plastic to finish the build. Sometimes a printed part will need to be sanded.

3.7.1.3.2 Cost

Finally, 3D printers are easily accessible to the group, so purchasing or paying to have our model printed is needed. There are also many options for free 3D modeling software that could be installed. Blender 3D and Autodesk Tinker CAD are two examples of free software we could download to work on the 3D model. Running out of filament could be a possibility and to replace one would cost about $30 for the design. Printing the 3D model could either be free or cost about $30 depending on the situation.

Building the design with a 3D printer would be the cheapest option because of free access to a 3D printer. The next cheapest option would be building with wood which would cost about $45. Building with steel could get quite expensive if we decide to get a welder. Even

if the team finds a way to weld for free, out of the three materials, working with steel would cost the most.

3.7.1.3.3 Time

After designing the model, printing the model would take a fixed amount of time. While printing, the print must constantly be checked to ensure the process is going smoothly. However, situations such as the printer breaking is a possibility which will result in costing more time.

3.7.2 Chosen Housing Material

For the housing design, the team has chosen to build it with plastic using a 3D printer. The filament chosen was PETG because of its strength, durability, and water-resistant capabilities.

3.8 Servo

Servo is short for servomechanism or servo motor and it is essentially a motor driven system with a feedback element. There are three core servo components, a motor, a controller circuit or amplifier, and a feedback mechanism. One of the goals for this project is to have a system that is capable of a dual axis movement. For the team to achieve this, servos must be implemented to our housing. Two servos working together to achieve a dual axis movement will be needed. One servo will be responsible for rotating the system in the horizontal axis and the other servo will be responsible for rotating the system in the vertical axis. There are many different options of servos available in the market. In selecting the right servo, considering elements such as the servo load, torque, speed, size, and weight is a must. In this section different elements need to be considered to select the best servo for the project will be discussed. Four different kinds of servos will be researched and compared.

3.8.1 Servo Choices

The first thing to consider when choosing a servo is the type of servo motor. There are two types of servo motors, a DC servo motor, and an AC servo motor. Both motors include three wires, power, ground, and the control wire. The difference between the two is how electricity works. When it comes to DC, it has both a negative and a positive terminal and the current flows between each terminal in the same direction. Depending on the region, AC current alternates a certain number of times per second. AC and DC motors can be further divided into two categories. For AC motors, there is synchronous and asynchronous. Synchronous AC motors have a high torque density and does not have a slip. They initially require a DC power source to rotate the rotor. Asynchronous AC motors do not usually incorporate feedback and they have a slip. Unlike synchronous motors, they do not need the initial DC power to get started. When it comes to cost, synchronous motors are more costly than asynchronous. Although more costly, synchronous motors are more efficient than asynchronous motors. As for DC motors, there is brushed and brushless. Although brushed motors have an overall lower cost, they are known for wearing and thus they require preventive maintenance. Since preventive

maintenance will be needed for a DC brushed motor, this type of servo will not be considered for the project. On the other hand, brushless DC motors are known for high efficiency and reliability. They require less overall maintenance compared to brush motors and is therefore more ideal.

The other factor to consider when choosing a type of servo is the load. A servo strong enough to hold the photovoltaic module and each component that will be attached to the servos must be chosen. Every component that the motor is responsible for moving will contribute to the total load inertia. The team must ensure to know the limit load of the servos and not exceed its limit to prevent issues such as vibration and in extreme cases breaking of the servos. Another important factor to consider is speed or velocity. The team must ensure that the servo can go as far and as fast as needed while carrying its load. Checking the torque speed curve for the servos which includes the two operating zones, continuous duty and intermittent duty is also important. The team must ensure that the application runs within the continuous duty zone during normal operation. The team also need to consider the amount of voltage the servos need to operate since there is a maximum voltage limit. The weatherproof aspect of the servo must also be considered. Since the device must be outdoors, considering a servo that is waterproof to prevent them from breaking and getting damaged is a must. Waterproof servos are usually more expensive than a regular servo and since the team must work under a budget, regular servos must be considered.

3.8.1.1 HiTec D840WP Servo

The first servo is the D840WP servo. It is also one of their larger sized servos and its dimensions are 2.59 by 1.26 by 2.44 inches. It has an IP67 rating ranking in which IP stands for International Protection Code, 6 stands for Dust Tight and 7 stands for Immersion up to 1M. HiTec claims that this servo has an unprecedented programmability and flexibility as well as delivering significantly smoother movements. [30] It has an operating voltage range of 4.8V to 8.4V. The microcontroller should be able to operate this servo. The features include a programmable high response digital amplifier with MOSFET drive and durable steel gears with dual ball bearing. The output torque at 4.8V is 264 oz/in which would be way more than enough to support the system's components. The IP67 ranking sounds ideal for F.A.S.T as we need our system to be weatherproof and this servo will do the job. However, this servo is very expensive and cost around $99.99. Deciding to choose this servo will put the team out of budget since a total of two servos is needed. Although it is higher quality than the other three servos and the features sound very enticing for the project, it most likely will not be the best choice due to its high cost.

3.8.1.2 HiTec HS-645MG Servo

The HS-645MG servo is one of HiTec's most popular servos. It is one of their standard sized servos and its dimensions are 1.59 by 0.77 by 1.48 inches. HiTec claims that this servo offers one of the strongest gear trains available in any servos. [31] This servo is ideal for projects that require durable high torque servos. This servo is not weatherproof unlike the D840WP which means that if we decide to choose this servo it will potentially need to be covered up to be protected. This servo has an operating voltage of 4.8V to 6V so operating this servo with the microcontroller should not be an issue. The output torque at

4.8V is 106.93oz/in which is lower than D840WP but should also be enough to support the components. This servo comes with 3 metal gears and 1 resin metal gear. This servo is much cheaper than D840WP and cost around $29.99.

3.8.1.3 DSSERVO DS3225 Servo

DSSERVO is not as popular as HiTec but they offer a servo called DS3224. This waterproof servo is much cheaper than HiTec's D840WP and costs $25.88. It is much more reasonable than D840WP's price tag of $99.99 and is more ideal for the team's budget. It is a standard sized servo like the HS-645MG, and its dimensions are 1.58 by 0.79 by 1.60 inches. This servo is commonly used for robots, mechanical arms, and remote-control toys. It features high-precision metal gears, CNC aluminum middle shell and dual ball bearings. This servo has an operating voltage of 4.8V to 6.8V and the team will have no problem powering it with the microcontroller. The output torque at 5V is 291.5oz/in which is more than the HS-645MG Servo and surprisingly more than the D840WP Servo.

3.8.1.4 SPT 5525LV

Unlike HiTec and TowerPro this servo is not as popular. It is also not waterproof. However, this servo offers 360-degree continuous rotation. This servo has an operating voltage of 4.8V to 6.0V and the team will have no problem powering it with the microcontroller. Its dimensions are 1.6 by 0.8 by 1.6 inches and weighs approximately 68 grams. The output torque at 4.8 V is 333.29 oz/in. It costs $18.99 for one servo.

3.8.1.5 Tower Pro SG90 Micro Servo

The SG90 servo is one of the most popular servos. It is included in many kinds of Arduino kits and is very popular to use as a beginner. As seen from the title it is a micro servo and is the smallest of the four servos we are considering. Its dimensions are 1.15 by 0.9 by 0.48 inches. This servo is also very light and weigh only 9 grams. The output torque at 4.8V is 25oz./in which is the lowest of the four but with the right housing design this servo can work. This servo has an operating voltage of 4.8V to 6V. Our microcontroller should have no problem powering this servo. The SG90 servo costs $13.95 for a set of two and is the most budget friendly out of the four servos.

3.8.2 Servo Comparisons

In this section all four servos will be compared. For each servo, **Table 7** includes the cost, max torque in oz/in, speed in sec/60 degrees, weight in g, and whether the servo is waterproof. Comparing these factors are important and they will help us choose which servo will best suit the team's need. Green is for good, yellow is for decent and red is poor.

	D840WP	HS-645MG	DS3225	5525LV	SG90 Micro
Manufacturer	HiTec	HiTec	DSSERVO	SPT	Tower Pro
Cost	$99.99	$29.99	$15	$18	< $7
Max Torque (oz/in)	417	133.31	347.1846	333.29	25
Speed (sec/60^0)	.11	.20	.13	.12	.12
Weight (g)	227	55.2	67	68	9
Waterproof	Yes	No	Yes	No	No

Table 7. Servo Overall Comparison

Although the D840WP servo is the most ideal option for the project due to its dust tight and waterproof IP rating, the $99.99 price tag is not reasonable given our situation. Its max torque of 417 oz/in with a speed of .11 sec/60^0 is impressive but because two servos will be needed, it will cost a total of $200 for the D840WP. If cost were not a factor, D850WP would be the selected choice. A much cheaper option for a waterproof servo, although less popular and arguably not as good quality, is the DSSERVO DS3225. DSSERVO's waterproof servo cost a sixth of HiTec's and it does not fall that far behind. It has a max torque of 347 oz/in with a speed of .13. For two servos it will cost a total of $52 which makes it a more reasonable choice than the D840WP. Because waterproof servos are generally more expensive, the team is also considering servos that are non-waterproof. The options we are considering is the HS-645MG, 5525LV the SG90 Micro. The HS-645MG servo standard size and its max torque is greater than the SG90 micro servo. The 5525LV has an impressive torque of 333.29 oz/in. SG90's and 5525LV's speed is faster at .12 sec/60^0 compared to HS-645MG's .20 sec/60^0. When it comes to price, the SG90 and 5525LV beats the HS-645MG at less than $7 per one servo. Because its max torque is only 25oz/in, making it the least out of the four options, it may not be able handle all the components.

3.8.3 Selected Servo: SG90 Micro

The team found that the SG90 Micro Servo with a stall torque of 1.8 kg/cm at 4.8V would suffice for the horizontal movement. The load for the vertical servo is greater as it is responsible for rotating the solar panels up and down. The SG90 servo was not strong enough and thus decided on the AGFrc Sub-Micro Servo with a stall torque of 3.0kg/cm at 4.8V.

3.9 Standards and Design Constraints

In this section, we first discuss applicable standards that pertain to F.A.S.T. After standards, we discuss the realistic design constraints of F.A.S.T. Some of the strands discussed include were PCB standards, coding standards and formatting standards.

3.10 Standards

Standards are a very important part of any engineering project and are integral to the success of a project. This project made use of several standards some of which are industry standards and some of which were implemented by the team for this project. Following these standards ensured that development was as easy, fast, and safe as possible. Themes standards ensured that health risks during development and operation were minimized and overall compatibility was ensured.

3.10.1 PCB (Printed Circuit Board) Standard

In order to have a high-quality PCB, we abided by the IPC Standards. IPC produces PCB-related standards and is the trade association for the electronic interconnection industry. They are accredited by the American National Standards Institute as a standard-developing organization. Adhering to IPC standards helps improve communication, helps reduce costs and gives a product instant credibility. IPC defines three classes of electronic products they are defined below. [32]

Class 1 products or general electronic products are for applications in which the primary requirements are the function of the completed assembly. This will include typical everyday products.

Class 2 products are dedicated service electronics, which have high reliability and extended life, uninterrupted service is preferable but is not critical.

Class 3 products are high-performance electronics which provide continued performance or performance on demand. There can be no equipment downtime and the end-use environment may be harsh. This equipment must function when required such as life support products.

Our PCB design was a class one product, it was a general electronic product that doesn't necessarily require high reliability, but still adhered to IPC standards all the same. There are IPC standards for most phases of PCB production. Below we list some of the IPC standards required to ensure consistent production results for our product.

IPC-2581: This is a generic standard used for printed circuit board and assembly manufacturing description data and transfer methodology. This standard ensures manufacturability, quality, reliability, and consistency in electronic assemblies for products.

IPC-2221: Is another Generic Standard for designed PCB that define flexibility of the board, design layout, parts lists, materials, and other physical design standards.

IPC-41401C: covers requirements for base materials such as laminate or prepreg, these standards are used for multilayered PCB design dealing with the materials used.

IPC-6012B: Printed boards may be, single sided, double-sided, or with or without plated-through holes. They can also be multilayered and can contain vias. The purpose of this standard is to provide requirements for qualification and performance of rigid printed boards.

IPC-A-600F: Describes preferred, acceptable, and non-conforming conditions that are externally or internally observable on the printed board. This requirement provides proper tools for evaluation of visual anomalies.

J-STD-001: Requirements for soldered electrical and electronic assemblies. They describe materials, methods and verification criteria for producing high quality soldered interconnections.

IPC-A-610: The most widely used standard for IPC, this is the source for end product acceptance criteria for consumer and high reliability printed wiring assemblies. This requirement demonstrates your commitment to customer requirements.

IPC-A-620: This standard is similar to IPC-A-610 in that is describes acceptability criteria for producing crimped, mechanically secured, or soldered interconnections. This standard defines wire diameter and the overall diameter of conductor plus insulation.

IPC standards are essential for all companies involved in the production of PCB and other electronic products to ensure quality, reduce cost, simplify communication, and improve credibility.

3.10.2 Coding Standards

As a team we must write code that adhere to certain coding standards to ensure the ease of programming. Programmers spend a lot of time debugging code and using coding standards would help the team detect problems earlier and thus preventing further headaches. By using coding standards, we have improved the readability and maintainability of our code. Having a uniform and consistent code results in less errors and becomes easier for any programmer to come in and understand the code without much difficulty. This is especially useful for any new future feature additions as well as updates. Following coding standards makes reading, checking, and maintaining code easier and less time consuming.

For our design project, we are using the Arduino IDE for the ATmega328P. The Arduino IDE is written in function from C and C++. We are also be using JAVA as the programming language for the software application feature. Below are some of the coding standards we will follow.

3.10.2.1 Coding Documentation Standards

Proper documentation and code commenting are a must. By properly documenting our code, team and other developers have an easier time trying to understand and follow the code logic. With proper documentation, any programmer looking at our code should be able to tell the functionality and utilize the code by reading the comments. At the top of the

program, we must include what is called a Header Comment which includes all the information about the authors as well as the dates. This Header Comment must also include a description of the program's functionality.

Every method, routine, or function is commented stating their purpose and their functionality. The description of both the purpose and function is in a block comment on top of the method, routine, or function. A block comment is a comment that includes multiple lines and is usually a paragraph or more. This description includes the return values, errors, exceptions if present as well as the algorithms used. Lines of code that do not have a clear purpose or are complex are commented as well. Comments within an algorithm should be as short as possible in order to not clutter up the overall code. Good comments are extremely important in any kind of software engineering. Commenting helps with catching bugs and logic errors early in the development process before they become harder to track down. A balance is found between having enough comments to properly document the program as well as explain to another developer how a program functions, but not be so long that the comments behave like a tutorial and end up being longer than the actual program itself.

3.10.2.2 Coding Modularity Standards

Standard headers for different modules have been used. These include naming and dating the modules, putting the author's name on the module, writing the modification history and a description of the module's functionality. By including all this information, everyone working on the code can easily understand the modules and look through the history. Any statement that has a functionality that is not easily understood just buy reading it includes an in-line comment either above the line or on the same line of it. Comments on the code do not be repetitive and anything that is obvious should not be commented.

Software is designed to be as modular as possible. Definitions and function signatures are outsourced to a header file. Placing all definitions and function signatures in a header file allows for very easy referencing during development. Any custom-defined datatypes also reside in a header file to also allow for easy referencing. Header files keep comments to a minimum, but they still have concise statements that explain what they are. However, datatype definitions are commented heavily in the header file to allow for any developer to quickly familiarize themselves with any and all custom datatypes that the software may have.

Program files are as modular as possible as well. Logic is broken down into its base logic and split it into functions. No single function should be too long. If any function is more than around a hundred lines, then it is split up into multiple shorter functions, if possible. Having many short to medium length functions allow for much easier debugging and development, especially when compared to having few long functions. By splitting up logic, into its base behaviors and creating functions to replicate these base behaviors any logic errors or programming bug can be caught quickly and easily. Splitting functions up also simplifies testing dramatically, by allowing every individual part of the logic to be tested individually and thoroughly. Another benefit of splitting logic up into multiple simple functions is that it allows one to improve parts of the program easily without having

to do massive rewrites. This project strives to minimize the number of long functions and maximize the number of shorter more digestible functions.

3.10.2.3 Naming and Formatting Standards

Global variables are visible to the entire program. By using too many global variables, we risk the increase of bugs, so minimizing the use of global variables is very important. They can also cause tight coupling of code and it is hard to decouple them when testing. Code that uses a lot of global variables will be harder to maintain especially as the program size grows. When using global variables, their name is stylized in all uppercase letters to denote that this variable is a global variable. As previously stated, they are used sparingly and only when absolutely necessary to ensure proper functionality of code.

Local variables, global variables, constants, and functions should have a consistent naming convention. We must ensure to use meaningful names that help anyone reading the code understand why the name is being used. Function names should be telling of the function's purpose. The function names usually include a verb and an object. When naming we must also avoid the use of digits. A very commonly used naming convention is known as "CamelCase" and it requires that variable names follow a specific pattern. If a variable name or a function name requires more than one word in its name, then each word will have its first letter written in uppercase after the first word in the variable name. Following the "CamelCase" convention has greatly improved code readability and allowed for much easier debugging. Names are descriptive, but concise. No variable or function are longer than a few words at the most, but they should not be so short that their meaning is lost. A balance has been struck between names that are short enough that they are quick to read and type, but also long enough that one can quickly tell what the name represents.

Consistent and proper indentation has been used throughout the code. A consistent indentation makes the code easier to read and thus easier to detect bugs. As a team we have used white spaces properly. Deep nesting which means too many nesting structures have been avoided when possible as this makes the code harder to read. The braces start from a new line as well as the end of the braces. Nested blocks also have been consistently spaced and indented. Lines of code is also terminated by a newline character and does not contain hanging spaces at the end of each line.

3.10.2.4 Version Control Standards

The software is maintained using some sort of version control to ensure all developers can work simultaneously and effectively. A very common form of version control is called Git and will be used to maintain this software. Git is a very powerful, but simple version control software. Git runs on command line Linux and is very easy to learn and has a tremendous amount of community support. The team has used GitHub to manage the software's git repository. GitHub is among the most used git server services available and offers desktop applications to further simplify version control operation in this project. GitHub also provides cloud storage to code that is uploaded to their website and allows for easy remote access which allows the developers of this project to work on the code easily from anywhere. GitHub is used exclusive as the version control server service for this project's software development.

3.10.3 UUID Standard and Bluetooth

Bluetooth standards are divided into two categories, the “Core specifications”, which describe details of the protocol layers and their relationships between each other. The “profile specifications” are the uses of Bluetooth technology to support different types of applications. The profile specifications define which protocols in the core specifications are implemented with a particular usage model. Bluetooth consists of a five layered protocol architecture which consists of core protocols, cable replacement, telephony control protocols and adopted protocols. The five-layer protocols are Radio, Baseband, Link manager protocols, Logical link control and adaption protocol.

For the F.A.S.T. device Bluetooth specifies a telephony control protocol. F.A.S.T. implemented Radio protocol, which specified details of the air interface, including frequency, modulation scheme, and transmission power. A baseband, which is defines connection establishment to the application. Link management protocol, which is responsible for link setup between Bluetooth devices and ongoing link management. The link management protocol also includes security aspects, though our device does not need much security, just a connection directly to the device. The Logical link control and adaptation protocol adapts the upper layer protocols to the baseband layer, which provided both connection and connectionless oriented services. Finally, Bluetooth standard contains a service discovery protocol which controls device information, services and characteristics of the services can be queried to enable the connection between two or more Bluetooth devices. [33]

Bluetooth standard as well is defined by usage models which describe the core protocols used in the specific application. In the case of the F.A.S.T. device, each profile defines the set of protocols that are implemented for a unique Bluetooth-based application The usage models that are implemented in our devices are, File Transfer which supports the transfer of files, directories, documents, images or streaming media. We have implemented file transfer to send data from the main F.A.S.T. device to the android application. LAN access which defines local area network or that a device function as if it were directly connected to the main device.

Finally, with Bluetooth standards within the android application are implemented a UUID or Universally Unique Identifier which are used to identify the layout of a device. In the case of the HC-05 device connected to the android app, we must use the 128bit UUID to open the Bluetooth connection within the Arduino software. Bluetooth’s unique UUID is “00001101-0000-100-8000-00805F9B34FB” that can be set with the UUID import library. This UUID will be used within the android application for Bluetooth standard connectivity

3.10.4 Battery Standards

There are multiple industry standards that make sure batteries are safe and reliable for consumers. Specifically, for lithium ion batteries, that batteries that we utilized in this project, there are specific set of standards that certify them safe. The UN/DOT 38.3

standard is a standard that needs to be pasted in order for batteries for lithium ion batteries to be transported via air, vessel, rail, or truck. This standard consists of 8 different tests completed in order. The first test is an altitude simulation and it simulates the batteries at different atmospheric pressure. The next test is thermal test which makes sure the batteries are stable at different temperatures. The third test is one of the most rigorous and is a three-hour vibrating test. The fourth and fifth test are electrical test and the test how the battery responds to shock and being connected to a short circuit. The sixths test makes sure the battery can withstand impact without damage or puncture. The seventh and eighth test observe the battery in different charge scenarios. The seventh is an overcharge the eight is a forced discharge, these test all ensure that batteries can withstand conditions such that they are safe for transport.

Another international safety standard for lithium ion batteries is IEC 62133. The previous standard is must be passed in addition to four more test to receive for a battery to be in compliance with IEC 62133. The first of the four is a molded case stress. The next is an external short circuit to ensure the battery does not exacerbated short circuit situation. The next test is a free fall where the battery is dropped and must receive minimal structural damage. The next test which is the hardest of the relatively easy four is an overcharging test. This standard is not very difficult to obtain if the battery was able to pass the UN/DOT 38.3 with ease.

Another standard that is relevant to this project tis the UL 2054. This standard is mandated in the United States and consist of an extensive series of test. There are seven different kinds of electrical test which are considered the hardest to pass. There four different kinds of mechanical tests, four different battery enclosure test, 1 fire exposure test, and 2 environmental tests. Since we are purchased a battery in the United States and our not manufacturing our own battery, we assumed the battery we ended up acquiring adheres to all of the standards listed. [34]

3.11 Constraints

Designs constraints were limitations on a design. For our project to be successful we had to define constraints and abide by them. These constraints were based on the project's goals and requirement specification. This section goes over different realistic design constraints and the impact the had on our design project. This section goes over economic constraints, environmental constraints, social constraints, political constraints, ethical constraints, health and safety constraints, manufacturability constraints, sustainability constraints, time constraints and testing/presentation constraints. It was important to view the constraints head on and address the different constraints individual ensuring the project was able to be completed to the specification in a reasonable timeline.

3.11.1 Economic Constraints

F.A.S.T was not a sponsored project; this means that the budget for designing this project was limited and funding came entirely from the group members. The goal was to complete this project with a cost of about five hundred dollars however we went about three time

over this amount. In an ideal situation, every component on the device would have been of the highest quality, however these parts were not as budget friendly. Because of the budget constraint, the team spent time researching and comparing different components such as sensors and microcontrollers. It was important for the team to decide between cost and quality with every component.

There was also a high chance that some components would fail. This failure could have been due to the manufacturer's fault or the team's fault. This meant that there were many times when extra parts were needed to be bought which also was allotted into the budget.

Another important factor considered while researching which component to buy is the location of the component. Depending on where the component came from, it could have taken weeks or months for the company to get it shipped and delivered. This also meant that shipping back a faulty component could have taken double the amount of time it took to receive it. With parts failing and delivery time as factors to be considered, ordering an extra part or two was worth the cost.

The PCB design was also carefully considered as it was one of the main sources where we can cut back on spending just through design optimization. Team members had to weigh the efficiency, BOM, and area that each part takes up on the PCB to ensure that it is the overall best design. It was also important that the PCB is tested as extensively through simulation as possible such that faulty PCB were not paid to be shipped and printed.

All these factors were considered while researching and deciding which components to buy in order to stay under the budget.

3.11.2 Environmental Constraints

Environmental constraints were the surroundings and conditions that might affect how the device performs. Since F.A.S.T is a solar tracker, the environment for our device is outdoors. We also ensured that our device is not harming the environment in any way. Having a device outdoors was more complicated than indoors as the surroundings and conditions are not controlled. The device had to withstand different outdoor weather conditions including rain, direct sunlight, cold weather, hot weather, wind and light hale.

Different components of the device were not waterproof such as some of the sensors and the printed circuit board. Different measures were taken in order to weatherproof these components to prevent them from failing. The stand must have to be weatherproof to prevent rusting and breaking.

One of our goals was to be environment friendly. As for non renewable energy, F.A.S.T does not use large amounts of energy as it was using solar which is a renewable energy source.

3.11.3 Social Constraints

Social impact must also be considered for this project. Our goal is to have a positive impact on communities by helping farmers. The device can be used to assist farmers in their daily tasks. Farmers are very important in our society as they provide us access to food and cloth

and they also strengthen our economy. This device can help make their tasks easier resulting in a more enjoyable experience.

Since our device's goal is to help farmers, the device must be user friendly and shall not cause any more stress. It shall require no more than five minutes to set up and use. It is also important that the F.A.S.T. is accessible in terms of price to groups of all socio-economic background. It is likely that the target group of individuals who will benefit the most from this product will need the product to be offered at the lowest cost possible without sacrificing data accuracy.

3.11.4 Political Constraints

Political impact must also be considered. The government develops agriculture to raise the gross national product (GNP). Agriculture is the cultivation of soil, growing crops, and raising livestock. F.A.S.T is a device that is designed to help farmers with their daily tasks. On top of using renewable energy source that is considered environmentally friendly, the device is capable of conveniently providing important data for our farmers. These characteristics help farmers and in doing so could potentially help the agriculture. However, since the project is mainly for academics, political impact is minimized.

3.11.5 Ethical Constraints

Considering ethical constraints for F.A.S.T is also very important. The team put fort their best effort into citing every source used as well as giving credit where credit is due. The team ensured that they are adhering to the principles of ethical conduct. Anything that may violate the code was not considered. The team ensured to prioritize the health and safety of anyone that will use the device. This means abiding by certain standards to ensure that the device is safe to use. If there are any risks at all while using the device, proper labeling and warning will be put in places that is easily visible. The team ensured that the device provides data as accurately as possible and took steps to ensure that incorrect data never reaches the user. The user is intended to use the information gathered by the F.A.S.T. devices to make decisions about their fields. If the F.A.S.T. devices are providing the user with improper data, then the user can be negatively affected such as by having reduced crop yields or even total field devastation. If the user of the F.A.S.T. device suffers any negative effects due to improper data being generated, then the team would be ethically responsible for any consequences the user faces.

3.11.6 Health and Safety Constraints

For health and safety reasons, the team's goal is to make this device as safe as possible to the users. To prevent any possibility or electrocution, all electronical connection points are covered. The team ensured that all connections are properly grounded, and all components are properly placed. As the device will be placed on the ground, the device is big enough to be easily seen, this will be to reduce the possibility of a person accidentally getting knocked down by the device. The device should always also implement high visibility markers such as orange cones or reflective tape on the device's body to ensure high visibility. Any material used to make the device was well inspected and there are no sharp

edges present that may pose any possibilities of the users getting cut. The 3D printed components were properly sanded.

Since the device will be used outdoor and will be exposed to different elements, the goal is to ensure that there would be very little chance of anything rusting. Rust encourages the growth of a bacteria that causes tetanus. To ensure the safety of the users from rust, the team ensures that materials will cause little to no rust. The battery used in this project must be protected such that it will not leak or create any dangerous emissions that may harm the user if they inspect the device.

3.11.7 Manufacturability Constraints

In every design, an important factor to be considered is manufacturability. Manufacturability is the extent to which a product can be manufactured with relative ease at minimum cost and maximum reliability. When designing the project, the team took into consideration different factors when it came to choosing parts. Because ease of obtainability and availability are important, the team considered how long the part has been in the market for and how easily this part can be obtained in addition to considering cost over quality. The team chose parts that are going to be available for a significant amount of time and not parts that are going to be obsolete. Since repair and maintenance are very important, our goal is to have every component in the project be easily replaceable by ordering through third party manufacturers.

One of the main manufacturing constraints will be due to the ability to print and test many PCB. For this project PCB was designed but was not be able to manufacture a PCB reasonably for enough devices to test the mesh network scalability. The team was also not able to mass manufacture a device with this type of electrical design or at least it would make more sense if the was only a single VCC line in terms of mass production and area that the board will take up. COVID-19 has also impacted the ability to have PCBs reliably shipped at a low price to the United States, some supplier for individual components have been impacted as well as supply chain are disrupted.

The team ensured to abide by the manufacturability constraints due to the fact that multiple units need to be produced to ensure this project’s success. The mesh network will require multiple units to be produced so by abiding by these constraints the construction of additional units will be as straightforward and simple as possible. These manufacturability constraints affect reproduction of the entire product as a whole and of the individual units that make up the final project.

3.11.8 Sustainability Constraints

Sustainability constraints are important to F.A.S.T as the device is meant to be placed and used outdoors. The device is exposed to different elements and weather conditions. It is a known fact that water is specially devastating for electronics. With this into consideration, the team must weatherproof every electrical component that is exposed, such as the electric board and sensors, to minimize damage. Protective cases help protect the components from dust and debris as these elements also have the potential of damaging the device and limiting its life span.

The electrical components of the F.A.S.T. have long enough life spans that they do not have a significant negative impact form a sustainability prospect. The two components that can have the most impact by having longer life spans is the photovoltaic module and the batteries. The photovoltaic module is not often recycled so if it did not work and had to be deposed would be a bigger impact than desired. Ideally, the team would want the batteries to have a usable life span of at least over 5 years. Anything under 5 years would make the battery choice unacceptable and that is one of the reasons the team has chosen lithium ion batteries. The photovoltaic module should work as a stable power supply for at least 10 years.

The stand for F.A.S.T is also robustly designed. The stand is very important as it holds everything in place and if it is easily knocked down, the team risk the potential of damaging the components.

3.11.9 Time Constraints

Possibly the most important constraint in this design project was time. It was very important that the team completed this project by the end of the fall semester which was in December 2020. Two semesters are given in order to complete the project. This was roughly eight months to have every single part of the design project implemented and working. The design, prototyping, implementing, and testing must be done by the end of December 2020.

Most of the first semester was spent doing research and writing the design document. The second semester was mostly implementing and testing the design. Because of the timing constraint, the team made a schedule and adhered to the schedule in order to make each of the project milestone’s deadlines.

As mentioned in the Economic Constraints section, considering the time to get a hold of components is very important. Since the project must be done at a certain time, the arrival time of these components combined with the possibility of them failing was taken into consideration. Getting a hold of these components early on was beneficial and the team allotted enough time to get a hold of the product, test the product, and the possibility of having to return and order a new batch. Since the recent unexpected impact of the Corona Virus everywhere, there were even more limited options.

Also because of the time constraint, the device’s features were limited as each addition of a feature cost time. Certain desired features, such as implementing another sensor, was put on hold. There were also situations where programming one feature can cause desire to add another feature for enhancement. The team ensured to not get distracted and only produce necessary features. The team also ensured to avoid feature creep which is the tendency of increasing product requirements. The goal is to be able to have enough features to satisfy the requirements. By using the Scrum methodology, we have control of releasing features right away. Once the main features are implemented and tested, and if there is enough extra time available, only then will the team implement these extra features.

3.11.10 Testing/Presentation Constraints

F.A.S.T is solar tracker that is meant to be used outdoors. This means that testing and showcasing the product must be done outdoors. Testing outdoors can be challenging and more complicated than indoors. In contrast to being able to control the condition indoors, the team could not control the condition outdoors. Planning when to test the device was necessary as there may be specific conditions needed to be met to test certain features. As an example, if the team wanted to test the weather station functionality, the team needed to be able to see how accurate it was by testing it in different weather conditions. If the team wanted to check how accurate our tracking was, we had to test it throughout the entire day. We tested it during sunny and cloudy days.

Due to the Covid-19 pandemic, it became difficult for the team to meet up and work on the project together during the first semester. Face to face meetings came to a halt and all the meetings were to be conducted online. This results in issues such as problems with full regressions when it comes to our software development. Being apart and only working online makes it more difficult to communicate. The most difficult part is working on the hardware design and mechanical design and testing them individually. During the second semester the team was able to consistently meet up twice a week and made significant process.

When showcasing, the device must be placed in an open space and preferably not in the shade. There may be days when it might be impossible to test certain features because of the rain. This means the team ensured to get enough videos and pictures of the device properly working in case of severe weather conditions.

4 Hardware Design Details

The following section discusses the hardware design of the F.A.S.T. mesh network devices including, sensors, hardware prototype design, and PCB design. Each of the devices have their own sensors and their own ATmega328 to process the data of each sensor module. The data is then sent via the LORA modules to the main master module. The main device includes a Bluetooth module which can send data to be view with the android application. Subsequently we will be using LORA to connect the devices across long distances (up to 1 mile). The master device will contain a self-charging battery using a rotating solar panel. The Solar panel will rotate based on the location of the brightest light data collected from the light intensity sensors. This is the direction that provides the maximum light energy for the solar panel to charge the battery.

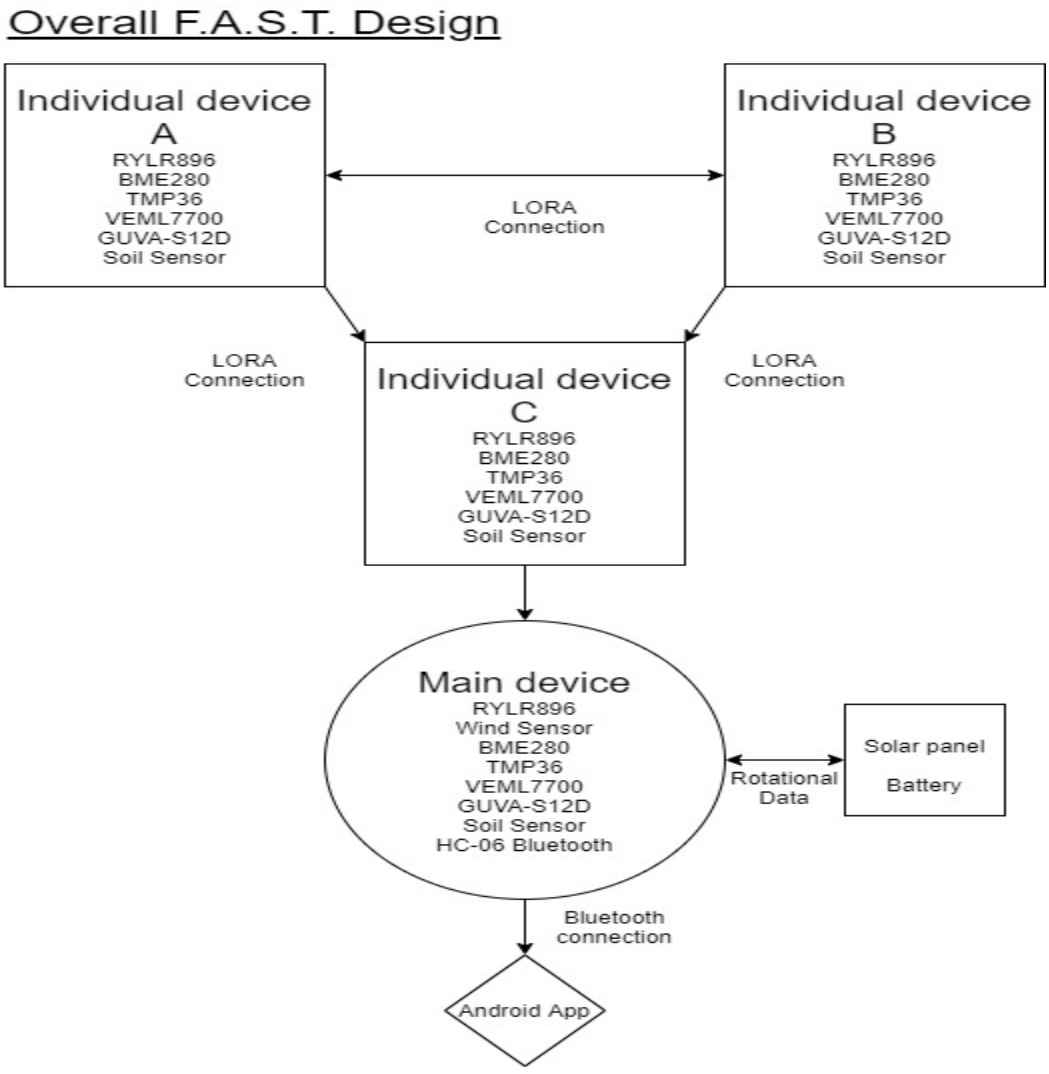

Figure 20. Overall Hardware Design

Above in **Figure 20** is a visual description of the device connections to each other using Bluetooth and LORA connections. Each device has its own LORA device for connection between them. Seen in **Figure 20**, devices A, B and C all have connection to each other but only device C can send data back to the main device. Devices A and B can send all their data to device C to be sent to the Main device. Once the main device collects the intended data only then will it be displayed on the android application user interface which is described in the software design section. The design of the PCB will be discussed in the later section 5.4 Master Device Design Details.

In retrospect, we created a star network, that instead would send data from each individual device to the main device. That way each individual device was able to send data back to the main unit without going through other units taking up most of the memory on the ATMEGA328P chip.

4.1 Sensors

As a constraint for the project we were required to use multiple sensor modules. These modules shown below are the sensors we as a group agreed to use. Each F.A.S.T. device has their own individual sensors with specific hardware design inputs to their own ATmega328. This data is transmitted via LORA to the main device then sent out via Bluetooth the android application. Some sensors modules required multiple lines, for example the HC-05 Bluetooth module uses one line each for input Rx and output TX. Or others, such as the VEML7700 lux sensor, which is an I2C device, required a line for serial clock (SCL) and serial data (SDA). Obviously, each sensor required an input voltage to power on, and most are connected to the 3.3V line. Finally, they will require a connection to a ground node to complete the system cycle.

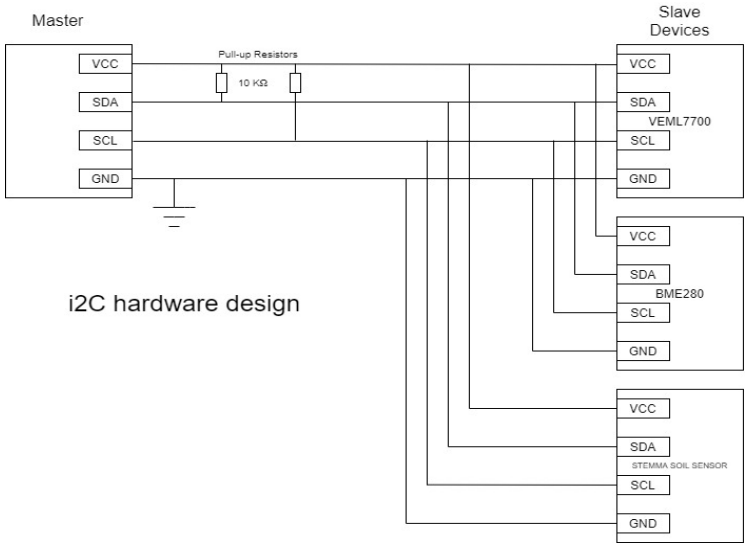

Figure 21. I2C Design

As described above some of the sensors including BME280 pressure sensor, VEML7700 lux sensor, Adafruit STEMMA soil sensors require a I2C connection. In the **Figure 21** above we show the hardware connection of the I2C lines for each of these sensors. The use of Pull-Up resistors is required for the correct timing from the ATMEGA328P. A pullup helps convert the open/closed connection of the switch inside the processor to a low or high voltage that the microcontroller can then sense. Each sensor has a specific address that are read to ensure that the data the ATMEGA is receiving is for the intended device. Since we will be using multiple VEML7700 devices we will have to ensure that the read address of each is different. We discuss more on this connection and addressing in the software design section 5.1.

4.1.1 TMP36 Temperature sensor

The TMP36 temperature module allowed the reading of ambient temperature via an analog to digital converter. Below in **Figure 22** shows the functional block diagram with pinouts for the module. Ground and 2.7V or 5.0V input lines are required to power the module. Using the 5.0V will allow the module to produce larger bits per voltage values and can therefore be more accurate with its signal detection. This data will then be modified to process the output analog signal sent back to the processor. Once power is received by the module the analog signal is sent back to the ATmega328 and converted into usable data via analog digital converter built into the ATmega328. The output data is stored together with data from the other sensors on the main device to be read by the Bluetooth device. [35]

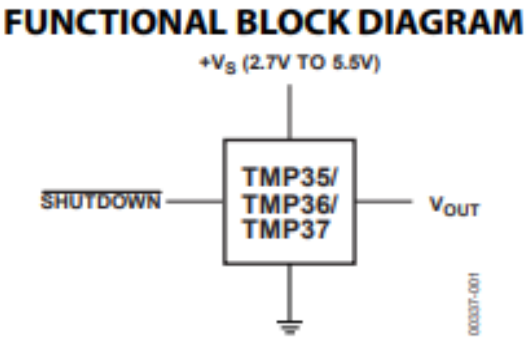

Figure 22. TMP36

The BME280 sensor had ambient temperature sensors built into them, therefore this TMP36 Temperature sensor was never used in the development of the F.A.S.T. devices.

4.1.2 GUVA-S12SD UV intensity sensor

The GUVA-S12SD UV intensity sensor must be supplied 3.3 V to 5.5v to power on. The module uses an analog to digital converter. Therefore, the output line out gives us an analog output of 0 V to 1.0 V that is connected to one of the analog inputs on the Atmega328. Once the data is received to the analog input pin it is processed with the analog digital converter to show its intended data and stored with the rest of that devices output data. The

details of how we use the ADC to gives us the intended output is discussed later in section 6.1.7. Once the data is collected from the module, the software takes responsibility for storing the data to be sent via LORA to the master device. Below in **Figure 23** is an image of the GUVA-S12SD UV sensor, the green wire is voltage input, the red wire is connected to the analog input and the yellow wire is the connection to the ground to complete the circuit. The device also contains an analog digital converter, we have added a similar set up to each PCB for UV index detection. [36]

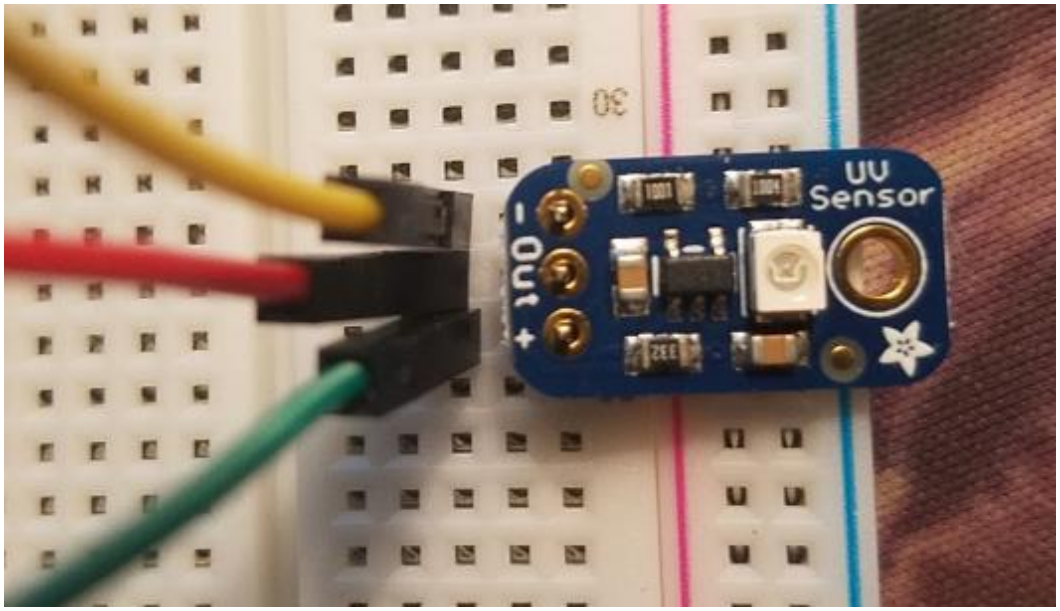

Figure 23. GUVA-S12SD

4.1.3 BME280 Ambient Humidity and Pressure

The BME280 is used to sense ambient humidity, temperature, and barometric pressure of the surrounding area. This device uses I2C, meaning the device has a line for the serial clock and the serial data. The voltage power input of the device ranges from 1.71 V – 3.6 V, so the device will be provided a 3.3 V input. The sensor module will also be connected to ground to complete the circuit cycle. Each of the devices contain their own humidity temperature and pressure sensors. For pressure sensing, noise of pressure is no concern, and a low data rate is needed. As well as for humidity sensing, power consumption is minimal. We use forced mode on the module to minimize the total power consumption when synchronizing the read outs for both pressure and humidity. The pin out of the module is shown above in **Figure 24**. Pins 1 and 7 are used as ground reference pins. Pin 2 is the chip select, which is used if SPI is utilized to read data from the device, for our device we will only be accessing the data with I2C. Pins 3, 4, and 5 are the I2C lines (SDI data in; SDO data Out) Finally pins 6 and pin 8 are voltage input pins, we used pin 8 with an analog input supply. We will have similar wiring connection to the PCB in the final design. [37]

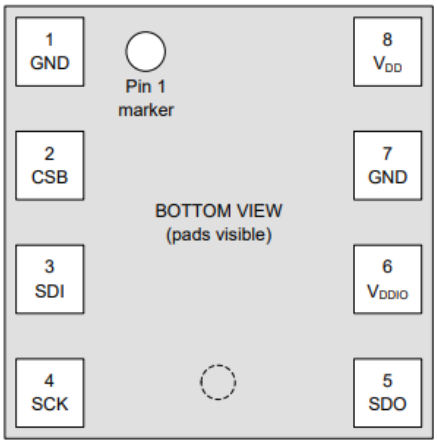

Figure 24. BME280 Pinout

4.1.4 VEML 7700 High Accuracy Light Sensor

Using I2C we connect the VEML770 module which is a high accuracy light resolution sensor or a high sensitivity photo diode to each of the devices. The module is contained in a 6.8 nm × 2.35nm × 3.0nm plastic packaging that could have been surface mounted to the PCB. Each module contains a low noise amplifier, a 16-bit analog to digital converter, and low power modes. Each F.A.S.T. device has its own lux sensor, but the main device contains an array of VEML7700 modules that track the direction of the sun, or most light in the vicinity of the main device. Then, the solar panel module rotates via servo device accordingly. Below in **Figure 25** from the VEML7700 data sheet is a view of inputs wiring for the module. Pin 1 is the serial clock line (SCL). Pin 2 is the input voltage (V_{in}) which ranges from 2.5 V to 3.6V. Pin 3 is the ground and pin 4 is the serial data line (SDA). [38]

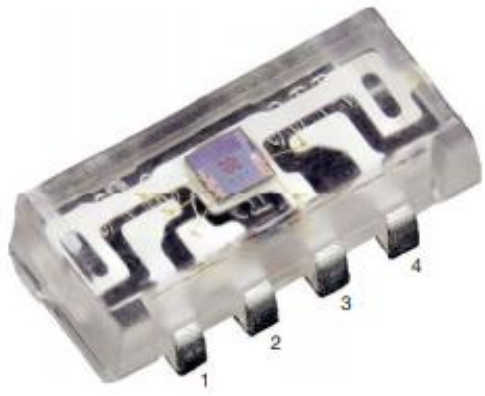

Figure 25. VEML 7700 Pinout

4.1.5 Adafruit STEMMA Soil Sensor

The soil humidity and soil moisture sensor we used is the Adafruit STEMMA Soil Sensor, this is an I2C device with four selectable I2C address and four wires. Two being V_{in} which requires 3V to 5V to power (we use the same power voltage as used for the I2C logic), and the ground to power the device. The other two wires are the I2C serial clock and serial data lines. The I2C lines within this device each contain a 10k pullup resistor to V_{in}. The lines of this device will need to be soldered separately to each device. The device needs external wire attachment for proper placement in the soil, therefore it is not soldered to the PCB.

4.1.6 Adafruit Anemometer

The Adafruit Anemometer uses an analog digital converter to convert wind speed into a readable voltage that ranges from 0.4 V to 2.0 volts. The input voltage required to power the device ranges from 7 V to 24 V, but 9 V is recommended. We need two wire connections to the ATMEGA328P to complete the power circuit, as well as one wire to connect the output voltage to an analog to digital convert. This device requires a relatively large voltage to operate and special care must be taken to ensure that powering this device is not hugely detrimental to overall power efficiency. This device could have possibly needed to have a sleep/wake process implemented on the software side. Implementing a sleep/wake process may ensure that the power consumption of this device does not become detrimental to the overall function of the F.A.S.T. unit. Once the device is properly connected to the processor, the sensor is tested to ensure the output is correct. Then, the device would have been attached to a hollowed metal pole. The pole also contained a spike on the bottom to hold the device steady in the ground as well as a mount on the top to place the Anemometer. The wiring of the device could have run though the hollow metal pole to be completely invisible. This would have provided a steady, easy to maneuver weather resistance support to mount the device. Only one anemometer is used in this project due to cost along with redundancy. The windspeed will not vary significantly across the distance that the F.A.S.T. network is intended to cover. The average windspeed measured at one F.A.S.T. unit can be considered to be the same windspeed that would have been measured at every F.A.S.T. unit. However, future iterations of this project may want to consider expanding the range between F.A.S.T. units and at that point the distance between units may be significant enough to consider adding anemometers to each F.A.S.T. unit.

This Anemometer did not need a sleep or awake function as the consumption we thought would draw too much did not necessarily take up that much power. As well, the anemometer was not placed on a pole as it would have made the device less portable.

4.2 Wireless Communication

The hardware development portion of the wireless communication functionality that this project requires is discussed in this section. The main communication methods implemented in this project are Bluetooth and LoRa. Each are used individually for their ideal purpose. Bluetooth has been developed for short-range communication and LoRa has

been used for medium to long-range communication. This section discusses the challenges and design decisions that were made with regards to wiring and hardware implementation when developing the wireless communication this project required. This section also discusses any decisions with regards to ensuring proper reception and whether or not to implement any external antennas to further amplify signal strength.

4.2.1 HC-05 Bluetooth Module

Connection to the HC-05 is required to receive data from the main device to the android application. **Figure 26** below shows the connection bus to the HC-05. Four wires are needed for connection of the module to the ATmega328. First, one wire is needed for ground connection, and another wire as a 5V input voltage to power on the device this is located a pin 14 in **Figure 26** below. This once powered on supplies the Bluetooth signal connection which can be connected to via android application. The next wire is connected from the TX out (transmit) or any serial output pin of the ATmega328, to the Tx input of the HC-05 to send data from the processor to the device. Finally, the RX input pin of the ATmega328 is connected to the RX output pin of the HC-05 to receive output data to the processor. Data is transmitted across these lines to and from the Bluetooth module sending the data needed to be displayed sent from each child device. Finally, this Bluetooth module is to be protected from the environment, we can in-case it inside of a 3D printed piece of clear plastic for protection. [39]

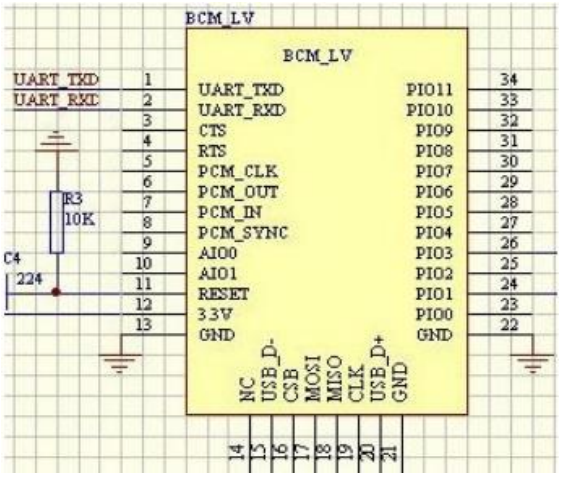

Figure 26. HC-05 Bluetooth Module

Though we were originally going to use the HC-05 module, it turned out that HC-06 was as cheap and functioned exactly the same as the HC-05 Module.

4.2.2 LoRa RFM95W Module

The LoRa module that was used in this project is the RFM95W. This LoRa module transmits in the 915MHz frequency which ideal for very long distance. This frequency is not ideal when fast data transfer is required, but this project had no need for high speed transfer. The data transmissions that was performed in this project are not very time

sensitive and a low transmission speed did not negatively affect performance in any measurable way. The LoRa module selected required an antenna. The selected antenna was a 915MHz omni-directional whip antenna. This was to be mounted externally with regards to the housing in order to maximize the range each transmission can be sent. This specific module uses the SPI communication protocol to be implemented and as such will require the wiring and hardware that all SPI devices require. This module required a voltage source of 3.3V. The input voltage had to be 3.3.V and was not able deviate much from this value or else the unit may have been damaged. In order to ensure that the input voltage is 3.3V, a 3.3V regulator was used. The regulator's output was then be wired up to the input voltage of the module. A linear voltage regulator sufficed for this purpose. The SPI protocol required the implementation of various pins such as the MISO pin, the CS, pin, the MOSI pin, and several others. Each of these pins had to be connected to the MCU in a specific way and to other SPI devices, had there been any, in a very specific way. Other SPI devices were not used in this project. The SPI pins had to be supplied with a 3.3V logic or else the chip may have damaged during SPI transactions. This module is commonly used in Arduino projects and as such is available in a breakout board form factor. The breakout board form factor, which is pictured below in **Figure 27**, aided greatly during the prototyping phase and was of some assistance during the PCB design phase. The actual unit used implement the RFM95W board directly with the PCB.

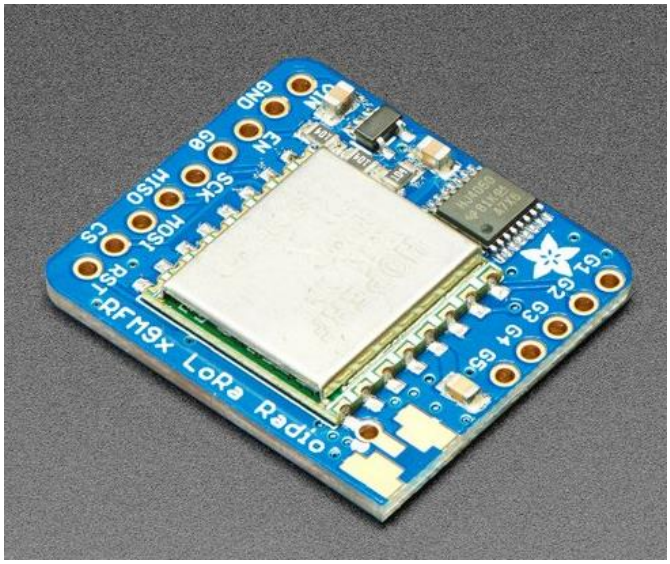

Figure 27. Adafruit LoRa RFM95W Breakout board

The LoRa module itself was housed safely within the protective housing, and the antenna was mounted externally to the housing. The antenna was then wired directly to the PCB inside the housing, while the antenna remained outside in order to maximize the range the module was able to transmit to. Each F.A.S.T. unit contained a LoRa module in order to

become part of the F.A.S.T. star network. Not all units will required a Bluetooth module or all of the sensors, but every unit had to contain a RFM95W LoRa module.

4.3 Servo Hardware Design

The servos motors have three wires, and they consist of power, ground, and signal. The power wires will be connected to the power pin of the MCU. The team ensured to not power it with a voltage that is too high as it might damage the servo. The ground wires were connected to the ground pin of the MCU. Finally, each of the servo's digital pins were connected to two different digital pins on the MCU.

Figure 28 is a block diagram showing how the servos are related to the lux sensors, microcontroller, and the panel. The microcontroller receives values from the sensors. The sensors are constantly feeding data to the MCU. Calculations will be made based on the sensor values and these calculations will determine which way the servos must turn. These calculations are based on the tracking algorithm. The microcontroller will then send signals to the servomotors. The vertical servo will rotate the panel vertically and the horizontal servo will rotate the servo horizontally covering both axes.

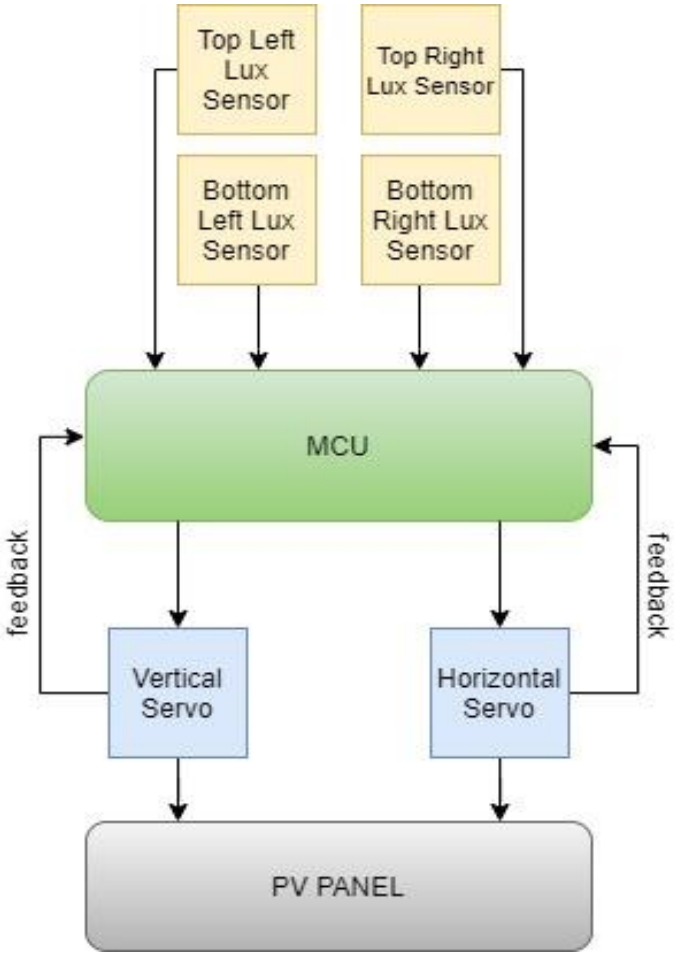

Figure 28. Servo Block Diagram

The F.A.S.T tracker has dual axis rotation. This means that the system will have two independent axes that will be layered on top of each other. As discussed before dual axis trackers allows for a higher energy output on sunny days. **Figure 29** shows where the servos are placed in relation to the housing. The tilt servo was placed closest to the photovoltaic mount and is responsible for tilting the photovoltaic module up and down. The rotation servo was placed on the bottom of the housing and is be responsible for rotating the whole system horizontally. The servo motors require a solid mounted position so we must ensure that they are properly mounted. The team ensured that all the wires from the tilt servo and rotation servo were properly tucked so that they will not be in the way of anything. The team ensured that they did not get stuck on any cervices to avoid breaking.

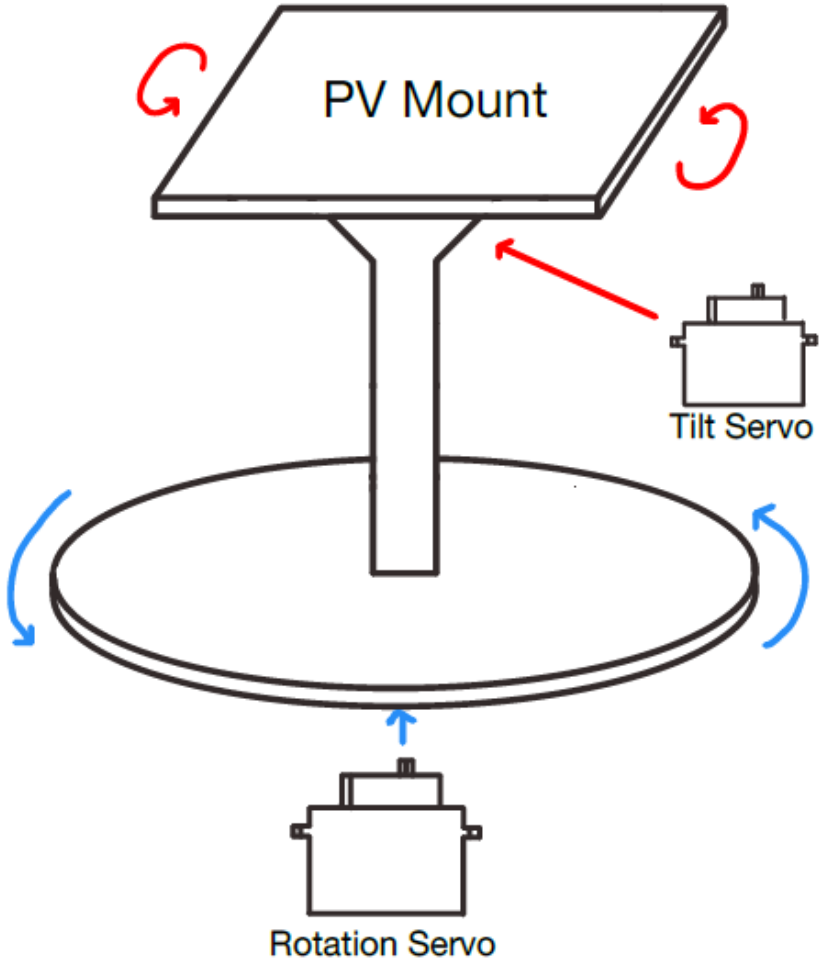

Figure 29. Servo Placement

4.4 LCD Module Hardware Design

The LCD in F.A.S.T is used to continually output data coming from the sensors. This is to ensure that users have access to information while close to the device or even perhaps in an instance where the software application. The application can fail due to issues such as the servers being down or simply because of bad connection. The LCD serves as a backup so that the users are not completely in the dark. The LCD we are using is the LMB162ABC which is 16x2 characters. The LMB162ABC is very popular among beginners and the team has experience with it from previous projects. It has three LCD display modes, STN, Positive and Translative. To change the display color, we set the Display Data to 1 for a Deep Blue color and set Display Data to 0 for a Yellow Green Color. The backlight is a Yellow-Green LED backlight. This LCD displays 32 ASCII characters in two rows with 16 characters in each row. Each character is displayed in a grid of 5x8 pixel. When it comes to pinout, the LMB162ABC comes with a row of 16 pins. Pin 1 and Pin 2 are both source pins which will be connected to Ground and VCC, respectively. Pin 3 is the control pin which will be used to adjust the contrast of the LCD. The register select Pin 4 will be used to toggle between command and data register and this is connected to the MCU. What toggles the LCD between the Read and Write operation is Pin 5 and this is connected to ground. To perform the Read and Write operation we have connected Pin 6 to the MCU. The pins that are used to send command or data to the LCD are Pins 7 to 14. For our project we only need four of these pins and connect it to the MCU. The minimum operating voltage for the LCD module is 4.7V and the maximum operating voltage is 5.3V. Since the LCD is placed outdoors, we need to ensure that it is bright enough for users to be able to see characters on the screen. With this type of LCD, it can be difficult to clearly see the text on the screen when the sun is shining directly on it. To accomplish this, we have tinkered with the backlight settings during the brightest part of the day and make sure that the screen is well lit and visible throughout the day. We have ensured that we place the LCD on the device very securely. Since it is not waterproof, we must also ensure that it is properly covered by a transparent case.

4.5 Power System Hardware Design

This section discusses all the design details that relate to the actual hardware implementation of this project. The photovoltaic panel that was selected is discussed in this section as well as the reasoning for selecting this panel. The battery selection is discussed in this section as well along with its reasoning. This section also discusses the voltage regulator required to ensure that the power system operates efficiently and safely. The tracking hardware is also discussed in this section because it is considered a vital part of the power system. All design choices and the reasoning behind them are thoroughly discussed in this section.

4.5.1 Photovoltaic

The panel selected is the Allpower 2.5-Watt monocrystalline photovoltaic module. This module is readily available on amazon and is relatively low cost as it is only around 6

dollars per module. This module is below the weight constraint of 3 pounds as it only weighs 92 grams. This should minimize the strain on the servo motors used for solar tracking and may allow us to choose a lower power servo. The panel outputs at around 5 volts and a current of 500mA. The charge controller we have chosen accepts 5 volts so it will work perfectly in conjunction with this panel with no modifications need. The panel also has a relatively high conversion rate of 17%. The dimensions of the photovoltaic module are also important as they will be used to decide how to house the unit. **Figure 30** shows the specific dimension which we will take into account when building the housing. It should also be noted that the photovoltaic does not include any mounting holes so that will have to be factored into our housing design.

Figure 30. Dimension of Allpower 2.5W [25]

4.5.2 Battery

The battery chosen for this project is a two 3000mAh 18650 battery and has a 3.7 Volt output. Based on the calculations calculated in the research section this battery should sufficiently power the weather station when the photovoltaic is not able to due to low light levels or any possible malfunction. The battery is lithium ion, so it allows for many charges without significant reduction in total capacity or significant degradation. The price of the battery was around 6 dollars per battery and the batteries were readily available on Amazon which offers prompt shipping times. The battery itself had a small form factor and relatively lightweight meaning it was easily secured within the housing to provide protection from the elements and protection to the environment. When the battery is fully

charge it outputs around 4.2 volts and when the battery is running low it outputs under 3 Volts. This means utilized the board along with an on off switch such that the F.A.S.T. has the capacity to kill power if the load is still trying to take when the battery is too low. Because we soldered the battery holder onto the PCB the batteries can be easily replaced if need be along with adding them if one single battery is not enough to sufficiently power the F.A.S.T. The battery was also protected through the construction of a protective housing. The housing ensured that the battery is not damaged due to prolonged exposure to direct sunlight or general exposure to inclement weather. If the battery happens to become damaged the housing served as a safe container for the battery until it can be replaced and prevented the environment surrounding the F.AS.T. unit to suffer any pollution from the breakdown of the battery.

4.5.3 Voltage Regulator Design

Originally, we needed two different voltage regulators for this project. The input will need to be stepped down from 3.7 volts to 3.3 Volts and we will also need a regulator to boost the 3.7 Volts to 5 Volts. This is because we are using the **Atmega 328p** chip with an external clock of 16mhz. When this cheap works at this increase clock frequency the lower range dvoltage of 3.3 Volts is insufficient to power the chip. We need another VCC line however for the sensors. While many of the sensor can accept a range of voltage from around 5 Volts to 3.3 Volts there are certain add on like Bluetooth for example that run the risk of being damaged if they run at 5 Volts. Both regulators will use the battery as input. To design the regulator, we will use Texas Instruments converter design tool to create two DC to DC converts one that convert 4.2-3 Volts to 5 Volts and another that steps it down to 3.3 Volts. **Figure 31** was chosen as the DC to Dc converted for 4.2 Volts to 5 Volts. It had decent efficiency as well as a relatively low BOM, the component estimated cost was less than a dollar. **Figure 32** is the same overall design but is altered to produce 3.3 Volt output. However this design described was not implemented because we need up requiring a 5 volt and 9 volt line so two buck boost converters were used to realize this. [26]

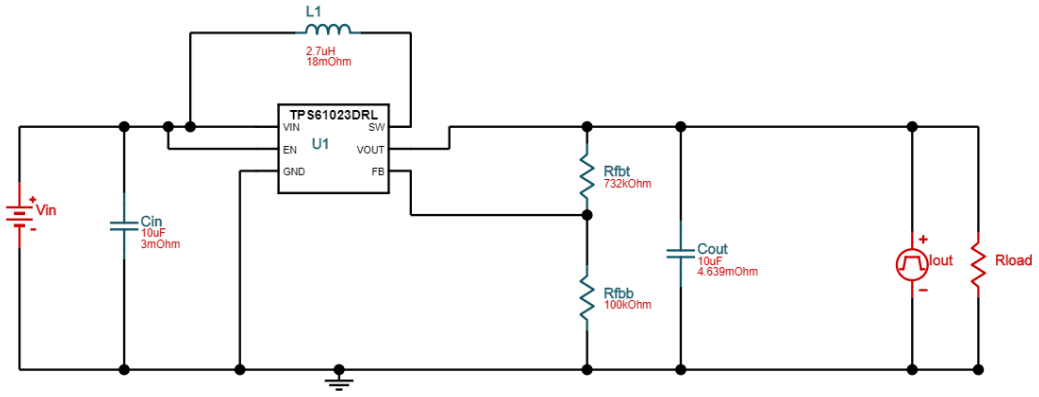

Figure 31. **4.2V-2.4V to 5V Regulator**

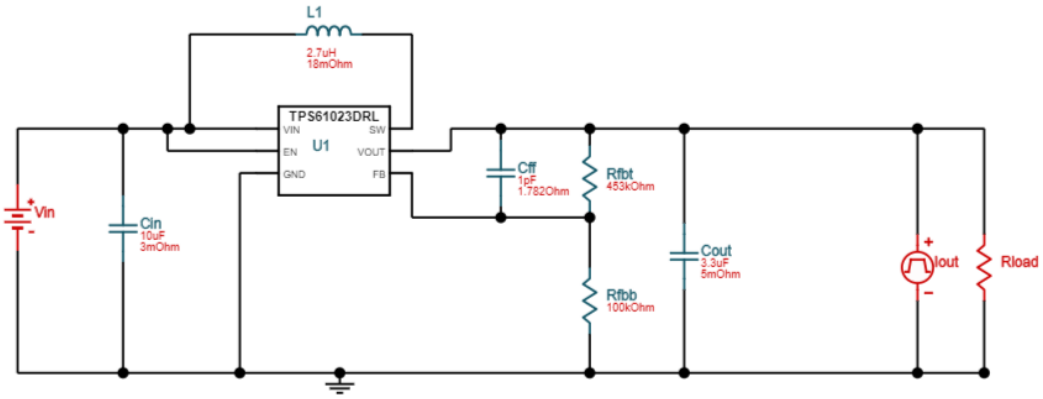

Figure 32. 4.2V-2.4V to 3.3V Regulator

4.5.4 Solar Tracking Hardware

To accelerate development time, it was decided to use a predesigned 3D model. The model used was found on a website called Thing Verse which provides primarily free open-source hardware designs. The design chosen is capable of dual axis rotation, however, to fit four of the photovoltaic panels, the mount had to be modified. The four VEML7700 lux sensors are mounted in the middle of the four panels and are shielded for directionality.

4.6 PCB Design

One of the requirements for this project was to implement a fully functioning printed circuit board. Although we were not required to physically build our own PCB, we did design our own schematic and then got it manufactured. In order to do this, a schematic was implemented in in a CAD software. Once a working schematic was devised, an actual PCB design was created. While this was being performed, the design had to be thoroughly tested on a breadboard prototype to ensure that the final PCB would work as intended. Once a final PCB design was designed and the design was verified, then it was sent to be manufactured and further tested.

4.6.1 PCB Schematic and Software

The first step in designing our PCB was choosing which core components we wanted to use. Once the components were selected, we made a system block diagram. The next step was creating the circuit design and this step took the longest. The schematic showed how every component connects. While building the schematic we ensured that we got the specifications for each of our selected components. To ensure that our design was effective we built the same design on a breadboard in order to test every component. By doing this we saved time and money.

To create the schematic, we choose a software tool that is used to design PCBs for manufacturing. Since we have all been introduced to Autodesk's EAGLE software, we used it to design our schematic. EAGLE electronic design automation software and it promises as an easy-to-use schematic editor. The software has a free student version available which should suffice for this project's needs and requirements. Since EAGLE is a popular software there are many textual and video tutorials available online which would be very helpful. Another popular CAD software that was considered for this project is known as KiCad. KiCad is a CAD software that is also used to designing schematics and producing PCB designs. The software is very similar to EAGLE. However, EAGLE and KiCad differ in the fact that KiCad is completely open source and entirely free unlike Autodesk's EAGLE. KiCad offers extensive add-on support and has many official tutorials available. However, KiCad is much newer than EAGLE and is not used as much as EAGLE is. It is currently undergoing widespread adoption by enthusiast and young engineers, but due to this it does not have as much widespread support and large community support like EAGLE does. For these previously mentioned reasons, the team selected EAGLE for the design and creating of all the schematics and PCBs that this project would require.

4.6.2 PCB Layout and Manufacturing

After designing the schematic, we then designed the PCB layout. We ensured that our components are not tightly packed otherwise this will take much longer to layout. We also ensured that we grouped our components in sections accordingly to reduce any noise and interference. To get started with manufacturing our PCB we first ensured to gather all the necessary files. The three files we needed to send our manufactures are the Gerber, Drill, and Netlist file and these were obtained through our software tool. The Gerber file includes information about the copper lays, solder mask and silkscreen. The Drill file helps with the sizes and placement of the drill holes and the Netlist file includes information about how all the components are connected.

Picking a good manufacturer for printing our PCB was important. We ensured that the manufacturer we choose would print a good quality PCB. This means that the board would be in the right dimensions, all holes are drilled perfectly, there are no warping and so on. Cost was also an important factor when choosing a manufacturer but if the cost is too low, we risk having a bad quality PCB. In choosing which manufacturer to use we had to have a good balance of cost and quality.

A very important factor considered, especially with the COVID-19 outbreak occurring as this project was being designed and implemented, was availability and shipping time. A manufacturer that can provide relatively quick turnaround and was available to manufacture during these very difficult times was very difficult to obtain. Many manufacturers ceased operations and many who have not are experiencing extreme delays due to the sudden increases in orders and extreme decrease in productivity due to the COVID-19 outbreak. Selecting a PCB manufacturer that could provide a balance between cost and quality and also be available and able to provide a manufactured PCB within a reasonable time frame was crucial to the success and competition of this project within a reasonable timeframe.

4.7 PCB Schematic Design Details

Using EAGLE software, we designed the schematic layout for each of the F.A.S.T. devices. Each of the sensors listed above in section 4.1 was implemented into a singular PCB design board to create a portable easy-to-deploy mesh networked weather station. The power PCB design was developed separately to ensure time and budget requirements were met.

4.7.1 Master Device Design Details

Using the modules discussed in the previous sections we created a schematic for the overall design of the main device PCB. Below in **Figure 34** is the schematic for the overall design of the main master device of the F.A.S.T. system. The following systems were connected to the master device, the power system connection, a LORA device to receive incoming data, a Bluetooth HC-05 module to send data to the android application, and its own individual sensors for receiving data it its vicinity. The device also included a 16 MHz crystal timer, which is designed to handle off chip crystals.

The main power supply system will provided 5.0V input for the devices. The 5.0V was be needed to power the HC-05 Bluetooth module. As well, the Bluetooth module will need to be connection to the microcontroller via UART. A LED diode will be connected to the Bluetooth module to ensure power is being received correctly by the device and we can establish a connection to the device via the android application.

Through the use of some logic converters and power converters, the I2C lines SDA, and SCL were connected to the 3.3V VCC line, with a 10 kΩ pull-up resistor to ensure correct timing with the ATEMGA328P. Each of the VEML7700 sensors was connected via I2C. Other sensors such as the BME280 and STEMMA soil moisture sensor were also connected via the I2C lines, we show an example of the connection for each device above in **Figure 21**. Each device was connected to the same SDA and SCL lines and will be delimited in software via its own device address.

The TMP36, and the GUVA-S12SD output an analog voltage to be read by one of the analog inputs read pins. Some of the analog sensors such as the GUVA-S12SD required a bit of design. Using an MCP6001 op amp to step up the very low nA current output of the UV sensor and provides a 0V up to VCC output that was to be read via one of the analog input lines located on the microcontroller. The TMP36's analog output was to be connected to an analog input pin of the microcontroller, more on the discussion of how we converted this output voltage from the device is discussed in the software design details section.

The RFM69HCW LoRa transceiver was be the device used to implement the LoRa network. The IC functions through the use of the SPI communication protocol and because of this has a MISO pin, a MOSI pin, and a SCLK pin. These three pins were required to implement the SPI protocol and had to be connected to the corresponding pins in the ATmega328 IC. The MISO pin on the transceiver connected to the MISO pin on the ATmega328. The same is done for the MOSI pin and the SCLK pin on the transceiver and the ATmega328. The RFM69HCW has a VCC line that can handle between 1.8V and 3.6V.

However, it is intended to be operated at 3.3V and as such the VCC pin of the transceiver is connected to a 3.3V VCC line which is shared with other devices in the schematic that also require a 3.3V VCC.

The transceiver also makes use of a reset pin which is connected to one of the digital pins on the ATmega328p. However, the Atmega328p digital pins, are designed to output to 5V by default. This is a problem as the transceiver is not rated to handle 5V values. Therefore, the all signals that are connected between the ATmega328p and the RFM69HCW that are known to exceed this safe range were passed through a 74HC4050D Hex high-to-low level shifter. The shifter will brought all the 5V levels down to a safe 3.3V before being passed to the transceiver for further processing. The transceiver also uses a hardware interrupt pin which is connected to a hardware-interrupt capable digital pin on the ATmega328p. Perhaps, the most important part of the RFM69HCW was the antenna. The transceiver has an antenna pinout to which an uFL/SMA connector is attached. This connector is the standard connector used for wireless antennas. A 915MHz attuned antenna was affixed to this connector on the physical device to maximize the effective range at which one node can communicate with another node in the network.

4.7.2 Master Connection to Voltage regulators

The ATMEGA329P microcontroller, each sensor module, and the communications devices require specific voltages across their systems to ensure their proper function. Using the voltage regulators designed in section 4.5.3 we are able to power these devices. Some of these devices such as the HC-05 Bluetooth module require supply output voltage from the 5.0V regulator for power. In **Figure 34** below, we can see that there is a 5V_VCC connection to the microcontroller and the HC-06 Bluetooth module. This connection is where we have placed a line from the voltage regulator. The same 5V_VCC line has as well been connected to the ATMEGA microcontroller that powers the device. This 5V_VCC is the output voltage from the top regulator shown in **Figure 35.**

Most of the sensors including VEML7700, GUVA-S12SD, BME280, STEMMA soil humidity sensor, and the TMP36 temperature sensor require a 3.3V input to provide power to the each of the modules. While some of these sensors can accept a wide range of voltage in terms of overall power consumption and protecting the device the majority of the sensor utilize the 3.3V VCC line. Below in in the schematic shown in **Figure 34** the 3.3V line is labeled as 3.3V_VCC. This voltage is output from the regular on the bottom of **Figure 35.** In the main device schematic this line has a connection to the sensors listed above. With these voltage regulators we are able to power all of the sensor modules, microcontrollers directly from the solar panel or the battery

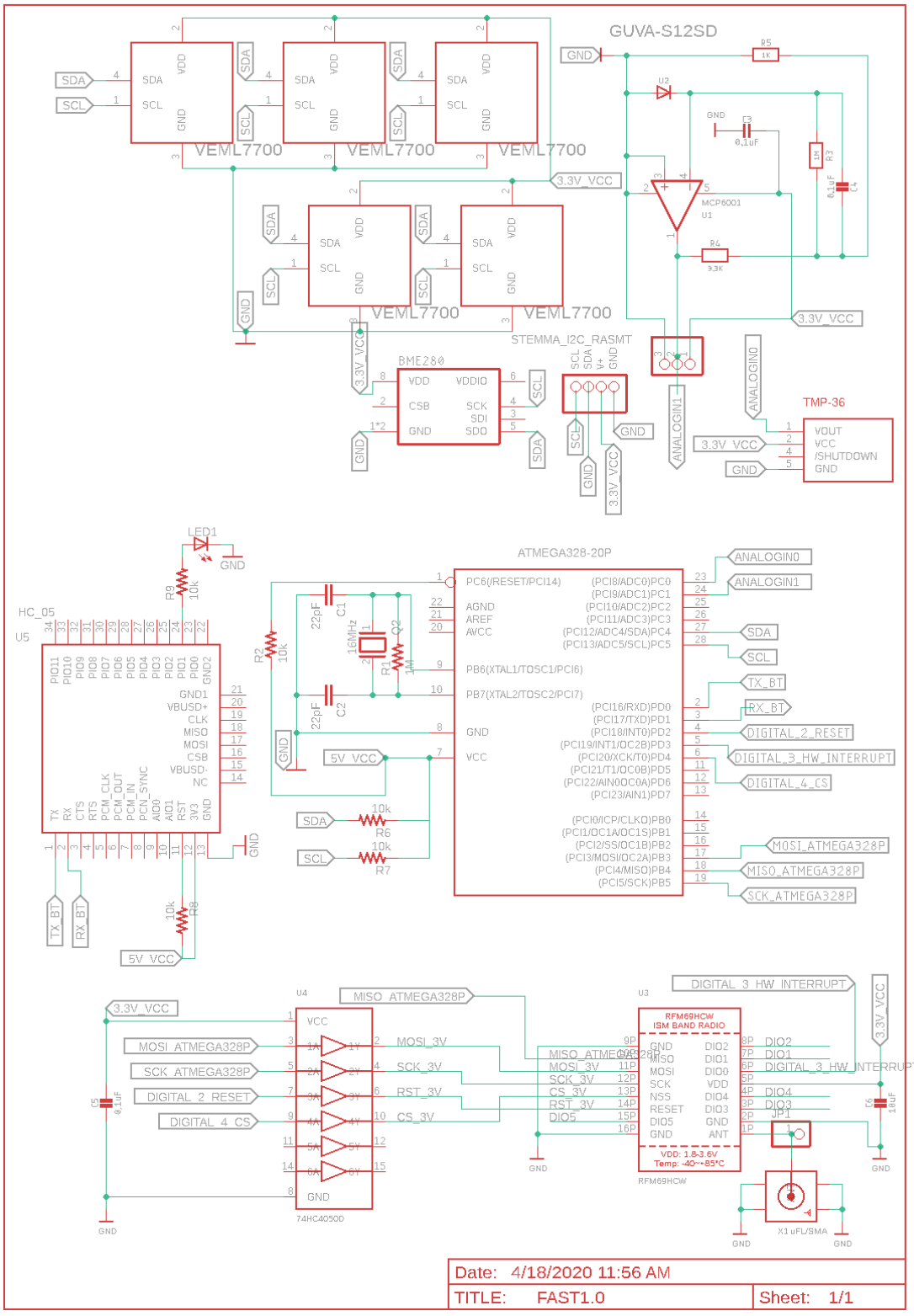

Figure 33. Master Device Schematic

Figure 34 depicts the overall device schematic without the power system. From the overall device schematic, it was clear that two different VCC line would be needed, on 5 Volt for the Atmega328p and another VCC line of 3.3 Volts for the sensors. **Figure 35** is a rudimentary schematic of how the photovoltaic is connected to a charge controller, TP4056, which charges the battery from the module safety. The battery is then connected to two different regulators, on which steps up the voltage to create a 5 Volt VCC line for the Atmega328p and then another regulator which steps down and stabilize the voltage to 3.3 Volts for the sensors. The TP4056 allows the temperature and voltage level of the battery to be monitored such that if the battery becomes too hot or if the voltage of the battery is too low to supply power the F.A.S.T. able to be switched off in an emergency measure.

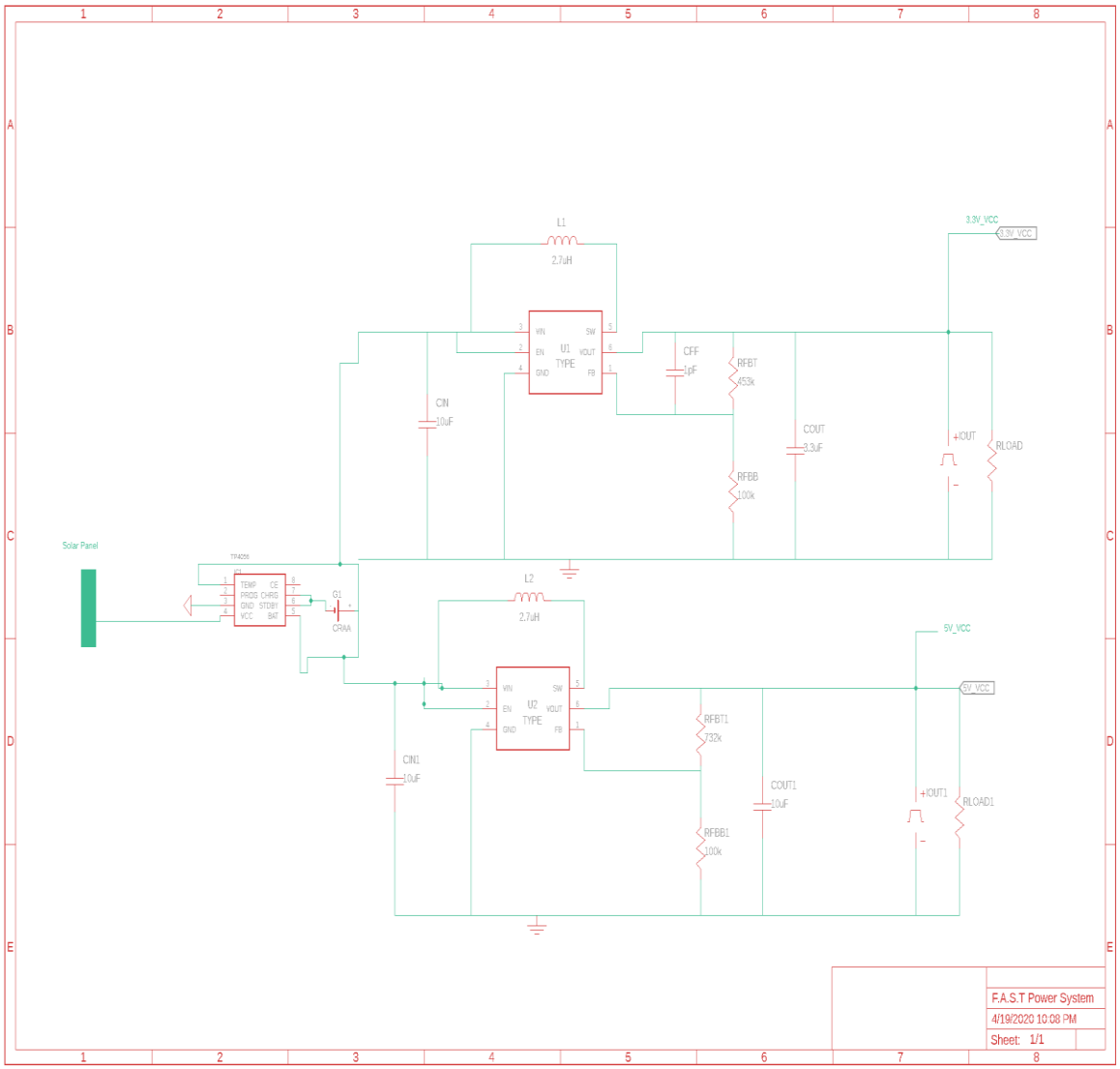

Figure 34. F.A.S.T Power System Schematic

5 Software Design Details

To design the F.A.S.T. devices, software was developed to control multiple different functions such as, board communications between each other using LORA, the control of sensor modules and the data they collect, timing with the master clock, control of the flow of data between devices and correct output display for each device. Software was designed for the external Android applications user interface and Bluetooth connection to the main device. Software is as well designed to send signals to control the power sources such as the solar panel rotation. I2C, analog to digital conversions, MOSI and MISO, and UART are all taken advantage of in our final project to send and receive data to the different device modules, collect data we intend to display from different sensor modules, and transfer data to our different display devices for testing and for the final product. The data eventually reaches its destination of the main master device which sends all this information via Bluetooth. This in turn which takes advantage of UART to send the data to the external android application. In the sections below we discuss our different design technologies used to create the F.A.S.T. device software.

5.1 Software Design Methodology

To produce reliable and high-quality software we have added structure to our software development workflow. There are many methodologies available when it comes to software development and choosing the right one for a project is dependent on factors such as team size and goals. For F.A.S.T we have decided to use the Agile methodology because not only are we familiar with it, but it also fits in with our goals. The Agile methodology is based on iterative development and it promotes a disciplined project management. As a team we have practiced continuous iteration of development and testing throughout the software development cycle. By constantly testing in each iteration, we get to have usable features right away. Under the Agile umbrella is the Scrum framework which is what we have used. This consists of 1-3 weeks Sprints where the team has worked as a unit to reach a common goal. There have been consistent meetings to talk about progress as well as give each other feedback. We have prioritized features by order of importance. By following Scrum, we have ensured higher quality, productivity and been able to incorporate changes as they occur.

5.2 Development Environment

The development environment that a project uses is as extremely important part of a project like this one. The development environments have been carefully selected such that it greatly improves the overall development process. Incorporating the best development environment for a specific project has greatly sped up development time and minimized the number of headaches. The right development environment is largely project-specific, and each project must ensure that their development environment is tailored to that project’s needs. This section will discuss the development environment that has been used for the implementation of this project as well as the reasoning for implementing such a development environment.

5.2.1 Arduino 1.8.12 IDE

The Arduino 1.8.12 IDE is the main IDE used for embedded development using Arduino products. The ATmega328p is the MCU that is used in the Arduino UNO R3 and therefore can be developed on using the Arduino IDE. A main reason for selecting this IDE is the vast amount of community resources available. There are countless hours of tutorials available online on this IDE as well as countless pages of community and official documentation. The IDE is very lightweight and allows one to easily import external libraries for use in projects. There are many built-in libraries that this project will make use of and are very well documented. The IDE also has bootloader burning functionality which will be very useful when burning the Arduino bootloader on ATmega328p ICs when it comes time to move from an Arduino UNO onto a PCB. The IDE also has many built-in example programs that are very well documented. These built-in sample programs can be used to further understand how certain protocols must be coded and how certain sensors must be implemented. This software is open-sourced and allows for the development of software plugins to add additional features to the IDE. There are many plug-ins publicly available which may help with development on this project and having the option to make use of them is a very good benefit. The IDE also has a built-in serial monitor which will be very useful during prototyping and troubleshooting. Having a built-in has allowed the team to debug and troubleshoot through UART all from the same software without having to switch over to an external serial monitor.

5.2.2 Android Studio 3.6.3

Android Studio 3.6.3 is an android development IDE that has been used for the Android app development portion of this project. This IDE is the official IDE provided by Google for development on their Android OS. Therefore, this IDE is extremely popular in the Android development community. There are many tutorials available publicly for this software. Many of the tutorials available are official tutorials published by Google and many others are unofficial community tutorials. There is a wealth of knowledge available online in community documentation as well as official documentation published by Google. This IDE has a built-in Android emulator which is very useful for quick prototyping and troubleshooting. The built-in emulator has allowed for quick changes in code to be swiftly verified without having to load the app onto a physical device. The IDE includes a built-in UI editor with a very intuitive drag-and-drop system. The UI editor has proved extremely helpful in the design and creation of the app because it has ensured that the user interface is easy to use and initiative to operate. The Android Studio IDE allows for the use and development of plug-ins much like the Arduino IDE. There are many community and official plug-ins available that had made developing the Android application much easier and has really speed up development time. The IDE also allows for the quick deployment of the application to an actual android phone. The ability to quickly install a new build of the application on a physical phone is necessary to ensure that the application functionality is working properly. There were moments when quick

changes to the code had to be made and they needed to be verified using a physical F.A.S.T. unit for which this quick application deployment which has been extremely helpful.

5.2.3 Visual Studio Code

Visual Studio Code is an open-sourced text editor developed directly by Microsoft. It is an extremely powerful text-editor is very popular in the software development community. This text editor has a huge number of features that will be useful during the development of this project. This text editor can be used for quick changes in the code used for the Arduino portion of this project as well as the Android portion of this project. The text editor can also be used to analyze output files and run any verification programs that may be written to ensure the functionality of the overall project. Visual Studio Code has a very active extension marketplace which act as plug-ins for the text editor. There are extensions available for almost any language as well as any software development purpose. There are even extensions available to do Arduino development on Visual Studio Code as well as deploy Arduino sketches onto an Arduino board. These Arduino extensions are not as feature packed as the Arduino IDE, but they have enough features with which to perform light development and small code modification. There are also extensions available for Android development. Like the Arduino extensions, the Android extensions are not as feature filled as the Android Studio IDE, but they have enough features to allow for light development and slight code modification which may be needed at times. Android Studio is a relative hardware intensive program and Visual Studio Code is very lightweight which will allow for coding on more low-end computers or on the go. Visual Studio Code also implements terminal management and allows the user to create multiple instances of terminals running command line operating systems such as Ubuntu. Being able to have many Linux terminals available is very useful for software development and specifically any programming done in C.

Although we mentioned using visual studio, it was used in a limited capacity during the development of the software for the F.A.S.T. project.

5.3 Arduino software

Since we are utilizing the ATMEGA328P processor the code is mostly designed and written in C which contains multiple different libraries to easily access different pins or elements built into the processor. **Figure 36** below shows the pin layout for the ATMega328P. Each pin can be accessed and is represented by its integer value. [40] We have taken advantage of a majority of the output pins on the ATMEGA328P for MOSI and MISO communication between the devices with the LORA mesh network, UART communication with the HC-05 Bluetooth connection. It takes advantage of serial input and output which uses i2C to communicate from master and slave devices such as the VEML7700 light sensors, and other i2C sensors. As well as analog to digital conversions from analog output devices to provide relevant data. All this data from each device will be stored to a JSON file that can be delimited at the main fast device.

The JSON file aspect of the project was thrown out because using serial outputs we were simply able to send messages between LoRa devices.

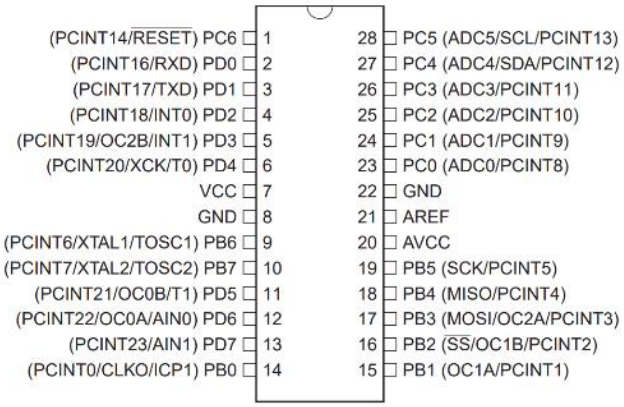

Figure 35. ATmega328P PDIP pinout

The ATMega328P contains digital pins as input and output variables, clock rates, baud rates, pin mode or specific libraries. In the software for the processors each contains two functions to run the file on the system, the void setup function, and the void loop function. Setup will only run once right after starting up, it is used to initialize variables, start timers, or know what pins to check for data during the main loop. Once the setup function is finished the code will run a loop function which runs continuously allowing your program to react or respond as intended. ATmega48P processor has a maximum clock frequency of 20 MHz but will run at 16MHz by default. ATmega48P uses multiple functions to send or receive different types of signals from pins using the serial functions. Serial allows us to begin, read, write, and check if we are receiving data through desired input and output pins and display the intended data to the console output screen.

5.3.1 Digital Pins as Input and Output

Pins for Arduino have been configured as input or output with pinMode() command. The pins default as input and do not need to be explicitly declared as input pins. Therefore, they are initially set in a high-impedance state when configured as input (INPUT). It takes very little current to move the input pin from one state to another. The Atmega348P chip contains a 20k pullup resistors built-in that are accessed via software (set pinMode() as INPUT_PULLUP) but this resistor value differs per microcontroller and can be found in the processor's datasheet. A HIGH reading means the sensor is off, a LOW reading means the sensor is on by default. If a pin is set to INPUT_PULLUP and we intend to connect a sensor, the other end should be connected "When connecting a senor to a pin configured to INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed" The pullup resistors are controlled by the same registers that control

whether a pin is read as HIGH or LOW. A pin configured to have pullup resistors turned ON with the pin set as INPUT will configure the pin as HIGH if set to OUTPUT, this works in both directions. An output pin left in a HIGH state will have pullup resistors turn ON if switch to an input using pinMode() Function along with the reading parameter. Digital write and read will allow us to send and receive values from digital pins. Analog input can be affected by electrical noise therefore we must be careful with wire placement on the PCB board when designing. [41]

5.3.2 Analog input

There are a few ways to write programs to read analog input with the analog digital converter. The ATmega48P pins A0-A5 are used as analog read inputs using an analog input. The analogRead() command converts the input voltage range, 0Vmin to 5Vmax (or 3.3Vmax) , to a digital integer value between 0 and 1023 or 0.0049 volts per unit (4.9mV) for 10-bit Arduino processor. If very larger input values are needed, we can use RCtime, which can be used to read resistive sensors. Using the internal 32k resistors we can create a voltage divider with an external resistor are use large input values. Input ranges are configured using analogReference() function which allows ATmega48P users to change the reference voltage to different built in reference values that will be converted to the values output by the sensor module. Using analog reference and an external 5k ohm resistor, we can also apply our own reference voltage using the AREF input pin. Doing this you can connect an external reference voltage allowing you to switch between internal or external reference voltages. The map command will allow you to map ranges of your analog signal. Value will store the read value from the incoming analog read function. The reference voltages must be changed. Using the **Equation 2** below the map function allows our code to change reference voltages FromLow and fromHigh to new reference values set by the user. ToLow and toHigh are the new intended values for the output range.

Equations 2. Map function: Map(value, fromLow, fromHigh, toLow, toHigh)

$$map = \frac{(value - fromLow)(toHigh - toLow)}{(fromHigh - fromLow) + toLow}$$

It is important that the analog reference (AREF) input is greater than 0V but less than 5V. There is potential to short together the internally generated reference voltage and the analog reference pin which could damage the processor. [42]

Although map was discussed within this section of the project, it is not used during our development process. We do however take advantage of analog read, for each of the analog sensors such as the GUVA-S12SD UV sensor and the anemometer wind speed sensors.

The reference used for these sensors is the default 5V for the UV sensor and 9V for the anemometer.

5.3.3 Using I2C with Arduino

Inter-integrated Circuit allows multiple devices that are connected to be programmed to communicate with each other. As opposed to UART, I2C can support multiple master devices and connect to multiple slaves via the same connection line. I2C uses two half-duplex lines, a serial clock line (SCL), which pulses at controlled rate for timing and sampling of data transfer across the serial data line (SDA) between devices. The F.A.S.T. accesses each different device by first sending the location in memory of the device sensor to the device that is read. Once the master has received an acknowledge signal from the slave device, it can read or write the intended data. Below in **Figure 37** shows a sample of how the two lines work together. As the clock line changes from low to high, a single bit of information is transmitted. The called upon device executes its request and transmits back to the master device the intended data. Using the Wire library, the processor can easily write the initialization addresses of each of the intended devices. Once the address is found we can use the requestFrom(deviceAddress) command to get the reading.

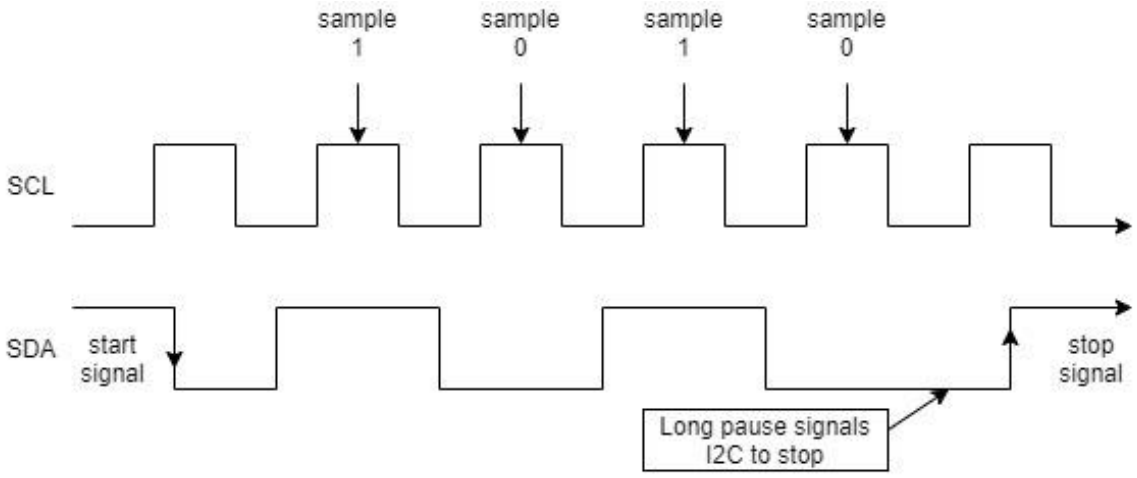

Figure 36. Waveform of Serial Clock and Serial data

The wire library is primarily used to access I2C devices. ATmega48P I2C contains two different addresses to access data, a 7-bit version that is used to identify the current device, and an 8-bit address determines if the device is being written to or read from. The wire library uses a 7-bit throughput address, this can be changed to 8-bit by dropping the lowest bit; or shifting the value to the right. This will allow us access to addresses from 0 to 127 (addresses 0 to 7 are reserved). A pull-up resistor is required for connection of SDA and SCL wires. The library also uses a 32-bit buffer and any device transmission that exceeds this will be cutoff.

5.3.4 Servo with Arduino

The solar panel is to be rotated with a 180 degree servos. Using the values from each of the lux VMEL7700 sensors, the panel and a simple compare function control the servo device to turn the direction with the highest lux value. ATmega328P and other Arduino processors on runtime can include a servo library with built in functions in the Servo.h library. Functions such as read and write are familiar, simply writing a value to the servo but the value is an angle between 0° - 180° will rotate the panel to that angle, and read will read the current degree that the servo is set to. As well it is possible to set write the direction using writeMicroseconds function with the input parameter values of 1000 µs to 2000 µs. 1000 being fully counterclockwise, and 2000 being fully clockwise.

5.3.5 Wireless Communication

This section discusses the different types of wireless technologies that this project utilizes and what has been implemented in terms of software development. This project requires concise, but also feature rich code in order to function. This section discusses all design decisions that were made in order to implement the Bluetooth connectivity along with the LoRa mesh network implementation. This section also makes use of descriptive flowcharts to breakdown complex algorithm code into easy-to-digest figures which aid the reader in understanding the operation of any and all algorithms described in this section.

5.3.5.1 Bluetooth with HC-05

Bluetooth with HC-05 accesses data via UART connection from the main device microcontroller. The various functions of each pin can be found in the variant.cpp file of the board [43]. RX (receive) buffers inputs from Bluetooth device and writes with TX (Transmit) buffers to Bluetooth device to send data to and from the device. Bluetooth uses a PAN or personal area network with a maximum data rate of 1Mb per second, with a nominal range of 100 meters using 2.4G frequency. Supplying 5 volts to the module can cause damage to the module (HC-05 has an internal 3.3v regulator). Once the master and slave are connected, the blue and red LEDs on the Arduino blink every two seconds. Our hardware design includes an LED to signal that the connection has been correctly established.

Serial baud rate is set to 9600 to read data from the HC-05. The transfer baud rate is set to 9600 by default for the HC-05 module. The Bluetooth module sends data through the TX pin (RX output) of the Bluetooth modules, to the RX pin (TX input) of the Arduino. **Figure 38** shows a flowchart for the Arduino code that receives which device data we would like to view (taken via serial read), then sends the corresponding data via TX line to the Bluetooth device.

The connection first must be established with the module by connecting the android device to the Bluetooth connection of the HC-05. Once the boards continuous loop starts, the serial value sent via the android application through the Bluetooth module represent what device number has been selected. The data is read from the master device once the data is

confirmed to be legitimate. This data is then sent to the android device via the TX buffer with a serial.print() function. All of the output data from each sensor will be stored to a structure that is created from a JSON file for easy access on the main device software.

During development of the F.A.S.T. it turned out we were able to get HC-06 modules for just as cheap as the HC-05. Data was easily sent via serial output from the HC-06, therefore the format of a JSON file was not needed for the development.

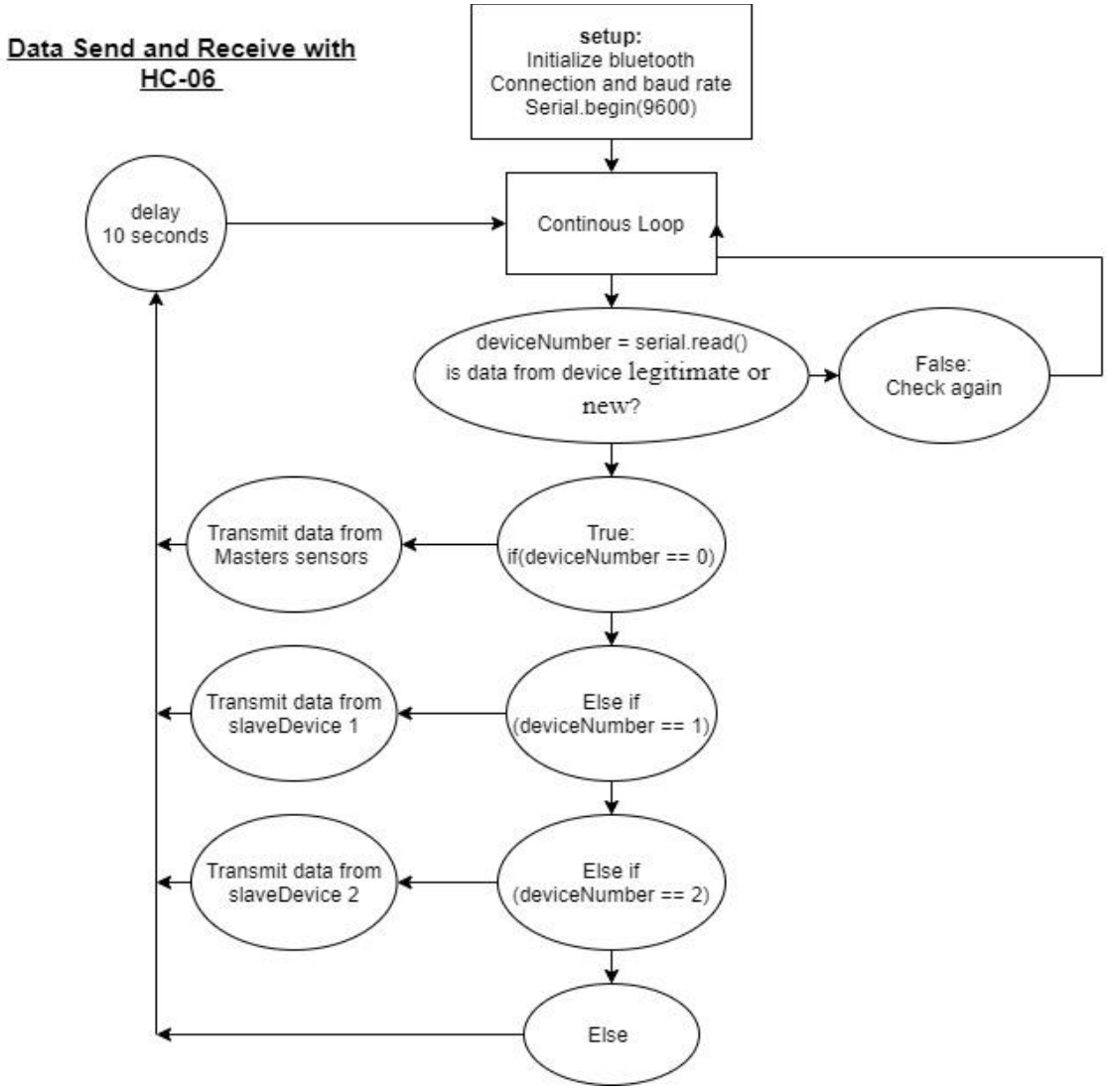

Figure 37. Data Send and Receive via Bluetooth

5.3.5.2 LoRa with RFM95W

The RFM95W LoRa module uses SPI to implement its functionality. The Arduino software then implements an SPI communication interface in order for the LoRa module to be implemented. There are official SPI libraries available for Arduino which made this part of the code very simple to implement. The complex part of the LoRa software implementation is implementing the mesh network functionality. Originally, the team planned to implement a mesh network which is described in this section. The software must create a network of all the LoRa nodes that it can communicate with and construct a mesh network based on these nodes. Concepts in network topology were used in order to attempt to implement the mesh network. There are basic mesh network resources available for Arduino, but these would have all required some modification in order to be implemented. The basic logical structure of a mesh network is shown in the **Figure 39** below:

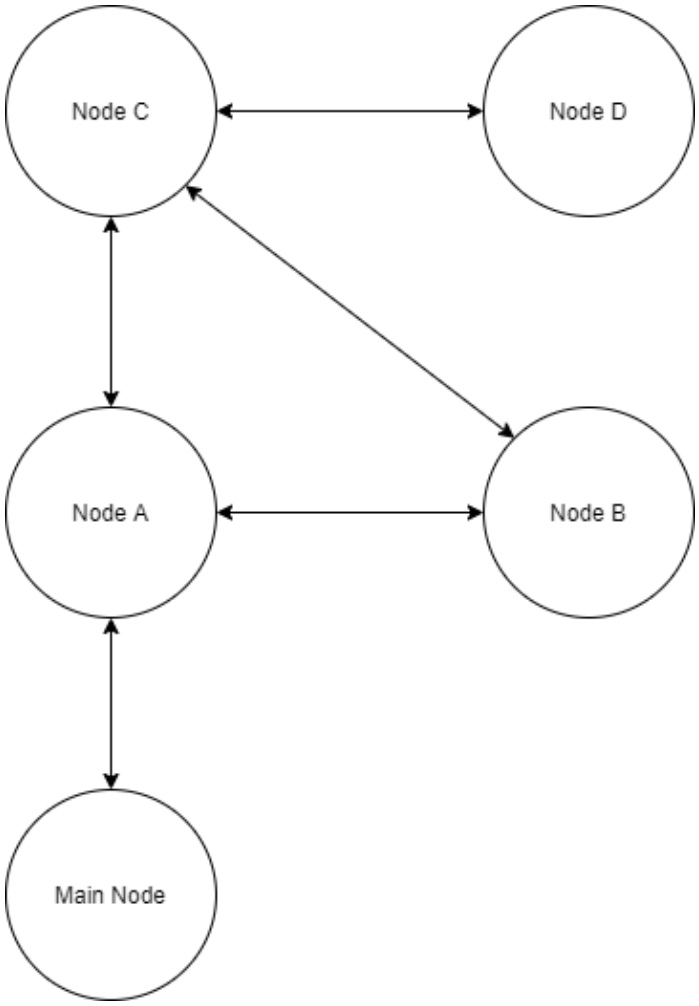

Figure 38. Mesh Network Example

The mesh network shown is partially connected network which is very similar to what the final F.A.S.T. network would look like if the original mesh network were implemented. The main node represents the main F.A.S.T. device which incorporated Bluetooth functionality to communicate with the user and the other nodes will be auxiliary F.A.S.T. units which will contain less functionality as the main unit in order to keep costs down and remove redundant measurements. The main F.A.S.T. unit was then able to receive data from units that are not connected directly to itself which is where the mesh network comes into play. The mesh network would have allowed data from nodes on one side of the network to reach nodes on the other side of the network even if these nodes are not directly connected. In the previous example, Node D will be able to transmit its data all the way to the main node by going through the intermediate nodes in the network.

The software implementation would have made use of available mesh network Arduino libraries and modify them to best suit our needs. Several software iterations were produced. The main F.A.S.T. unit acts as a hub of sorts and requires its own code that is functionality different than that of each additional F.A.S.T. unit in the network which acts as nodes. The main F.A.S.T. unit's code would have originally required mapping out the entire mesh network using data received from all nodes. Nodes would have been required to send data from other nodes to the main unit if the main unit cannot establish a direct connect with a specific node. Once the main F.A.S.T. unit had mapped the mesh network, then this map would have been provided to all nodes in the mesh network and would have been used to facilitate communication through the mesh network. Having a map of all the nodes would have allowed network traversal algorithms to be used in this implementation of which there are many each with their own sets of benefits and disadvantages.

During development of the F.A.S.T. network, the team quickly realized that a mesh network would have not been necessary due to our use case and would have been impractical. The code size required to implement a mesh network would have taken up most of the available program space on the ATMega328P which would have either required an additional MCU that was solely responsible for the mesh network communication or a more expensive and complex MCU would have been required. The range observed between units also made a mesh network unnecessary. Fields offer ideal LoRa performance and distances of up to 10km can be observed in similar use cases. The team decided on implementing a star network in which the master unit was the center of the network. Functionality the transmissions were still made in the same way, but no mapping was needed, and all auxiliary units sent their transmissions straight to the master unit.

5.3.5.3 JSON for Transmission

Data sent via the F.A.S.T. network is be stored to a JSON file for easy separation and access to each individual device along with its sensor data when it reaches the master device.

Shown below in **figure 40** is an example of how we format the files containing all of the devices and their relative data.

Each device contains a name and sensor details, delimited by brackets. This file is built up from each device and then sent to the main device for proper delimiting and display via the android application. If a device is missing a sensor or an error has occurred when writing to the JSON file, we store a -1 to the double value or a null character to a string missing a value that way we can tell if new data has had an error updating, or that the devices doesn't include that type of module. The names for each device and sensors never change but the output values can differ between devices. Once delimited they are to be stored in a structure which is discussed in the next section. Each JSON file has the same structure every time so we are able to consistently parse the files. In section 5.2.5.5 we describe the reading and writing of the JSON file, and that it will always be the same size because we always know exactly how many of the devices we have. The data from this JSON file will need to be read or written to by each of the different devices in a precise way for easy access later. In the next few sections, discuss the storage, sending, and separating of this data for user access once it reaches its final destination of the main master controller.


```
[
   {
      "device name":"device A",
      "sensor details":{
         "temp":91.1,
         "uv":8.0,
         "windSpeed":0,
         "luminosity":524,
         "pressure":101.325,
         "humidity":73.3,
         "soil moisture":73.3,
         "measurement time":13201
      }
   },
   {
      "device name":"device B",
      "sensor details":{
         "temp":91.1,
         "uv":8.0,
         "windSpeed":0,
         "luminosity":524,
         "pressure":101.325,
         "humidity":73.3,
         "soil moisture":73.3,
         "measurement time":13211
      }
   }
]
```

Figure 39. JSON storage example

During the development of the F.A.S.T. we found that using the JSON file was more complicated than sending the data via serial output and having the Bluetooth thread and application delimit the data. As opposed to having the Arduino software delimit the data which would take up too much space.

5.3.5.4 Data Structure for Storage

The F.A.S.T. network sends data from one node to another across several nodes. This data is sent in a compact and unified way in order to ensure that data is received intact and in a timely manner. Therefore, we will send the data via a JSON file as described in the previous section. A data structure was designed for this purpose to store the data taken from the JSON. The data structure designed is used for local storage on each node to keep track of the latest readings of the device. The data structure is converted to a JSON for transmission throughout the F.A.S.T. network.

The code that will be run on the ATmega328p is written in the C programming language. The C programming language allows one to implement custom data structures through the use of structs. A struct named "NodeReading" was designed which held all of the relevant information that the node in question would have. Each node will periodically take measurements and store the value of these measurements in an instance of a NodeReading struct. The NodeReading struct contains values such as node temperature, the intensity of the UV light that the node is experiencing, the windspeed that the node is withstanding, the luminosity being read by the node, the pressure in the surrounding area, and the humidity in the nodes immediate vicinity. A class diagram of this data structure is shown below in **Figure 41**.

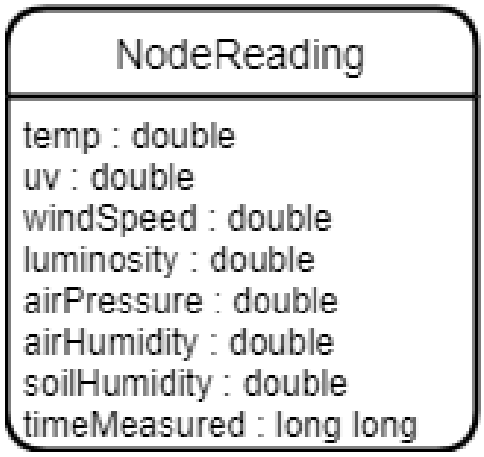

Figure 40. NodeReading Class Diagram

With the use of this NodeReading structure, the data can be sent from one node to another in a very efficient manner by simply converting the data to a JSON before transmission. The information can be kept in a compact structure on the local node before being sent just through a single instance of the NodeReading JSON. This implementation saved on local storage and minimize transmission time.

Data is stored in a structure of floats when it is read from the nodes and sent via LoRa.

5.3.5.5 JSON Write and Read

Described in the sections above we need to store the data for proper transmission between devices, as well as read the data to be displayed across all devices. First, each fast device needs to write to the JSON file, the initial device will write to the first device delimited by its device name, and its opening and closing brackets. The data is initially stored as a -1, that ways we know if the data has not been stored yet, as well as if there is an issue with the sensor that is going to overwrite the -1 in the data.

Each JSON file has the same format throughout the process of sending and receiving. That way when it finally reaches its last destination, that being the main master device, the function always reads the same file format. This is possible because we always have the same number of devices. Once it is properly stored in the formatting shown above in **figure 41** it then it is sent to the android application via UART for viewing on the user interface. Below in the flow chart of **figure 42** describes the reading of the JSON file via software. Each device is delimited by an opening "{" and closing bracket "}", as well, the data from each sensor is delimited by a comma ",". For easy reading and storage onto the NodeReading structure.

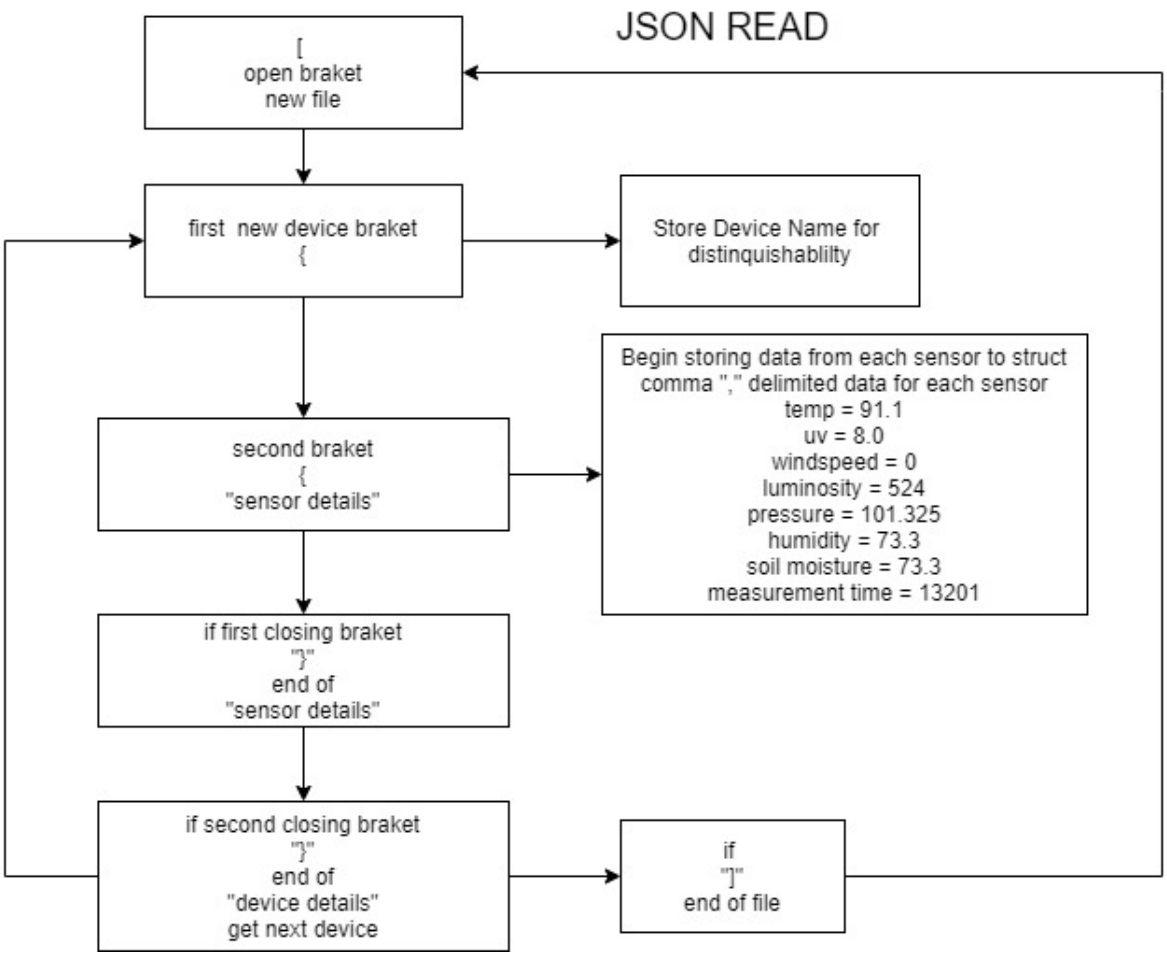

Figure 41. JSON Read

JSON was not used as sending data in the form of a structure of floats allowed for a small amount of space to be taken up for sending and receiving between units.

5.3.6 Sensors

Each device has multiple sensor modules that require connection to the main F.A.S.T. device. Some of the sensors use different styles of communication to its processor such as UART, I2C, or analog to digital conversion this is all controlled by the Arduino software. Using LORA we have created a web network where each individual device can send data back to the master device. Though the data is sent via a physical medium, we are required to time the transmission of this data to ensure the data that has been received is the information we intended to receive.

Each device can store its own data in a specific order to a file. The file will be checked for legitimacy for each value as well as size to ensure each file then sent back to the main device once the device is sure everything is formatted correctly for reading later.

5.3.6.1 TMP006 (Temperature of an object)

To gain access to the data provided by the sensor we first need to configure its I2C address. This is done by the setup function which will write to the device the memory location we intend to read on the sensor. For the TMP006 we write to one of the address from 0x40 to 0x47, since our master device has multiple devices connected, we may need to select a different address in this range. The device now reads the same address to grab the updated data to be calculated accordingly. This module is used to check the temperature of each device to ensure there is no overheating throughout board. [44]

We ended up not using the TMP006 because the BME280 provided us with a temperature reading for each unit as they were soldered to each PCB.

5.3.6.2 Temperature Sensing with TMP36

The TMP36 is used to read ambient temperature. It sends data back to the main microcontroller using the analog to digital converter. We use the following conversions below on the read input voltage to the microcontroller from the output pin of the TMP36. The code ranges our analog input voltage from 0 to 1023. The software uses the equations below to convert this input value directly to the current voltage output at the specified pin.

The decision of which equation to use depends on the input voltage to the temperature sensor. **Equation 4** uses input voltage of 5V, while **Equation 5** uses input voltage of 3.3V. Once the device has our voltage reading in millivolts it converts the value into temperature using the formula in **Equation 6**. Finally, we can convert the temperature to Fahrenheit using the **Equation 7** below. Once the values have been stored into a variable, this data can be serial printed and sent back to the main device for viewing on the android application user interface. [35] We are using the input voltage value of 5 volts to the TMP36 and therefore we take advantage of **equations 3** and **7**.

Equations 3. Voltage at pin with 5V input

$$V_p(mV) = V_{in} * \frac{5000}{1024}$$

Equations 4. Voltage at pin 3.3V input

$$V_p(mV) = V_{in} * \frac{3300}{1024}$$

Equations 5. Centigrade Temperature (C°)

$$C° = \frac{V_p - 5000}{10}$$

Equations 6. Fahrenheit Temperature (F°)

$$F° = \left(C° \times \frac{9.0}{5.0}\right) + 32$$

Instead of having one single device to take the ambient temperature, we use the BME280 with multiple data readings to limit cost and allow for simplicity.

5.3.6.3 UV sensing with GUVA-S12SD

The GUVA-S12SD chip detects UV radiation in sunlight then using the analog to digital converter gives us a readable value to convert to UV index. The module outputs an analog voltage (0 and 1.0V) which varies with a change in UV light intensity (0 to 11+ UV index). First the analog read value from the module is read into the processor to be converted by the ADC. Once we have our digital input voltage it is converted to sensor voltage using **Equation 8** below.

Table 9 shows the conversion values for the analog voltage to the digital sensor voltage, to the UV index value which is displayed on the android application user interface. The input voltage to the UV module is 5.0v, therefore we calculate the analog to digital conversion with using Equation 8 with an input voltage of 5 V. We can as well use 3.3V but the output voltage (V_{out}) which has a smaller voltage per bit value and is more difficult to read. A smaller input voltage causes a smaller range of voltages to be assigned to each digital value which can require a more precise measurement system in order to convert from analog to digital. However, we implement an algorithm which works with both the ideal 5V input voltage along with the non-ideal 3.3V input voltage to ensure that the UV sensing system works in all possible cases. [36]

Equations 7. Sensor voltage (V_{out})

$$V_s = \frac{(V_A)}{1024} \times Digital\ Input\ Voltage$$

UV index	0	1	2	3	4	5	6	7	8	9	10	11+
Vout (mV)	<50	227	318	408	503	606	696	795	881	976	1079	1170+
Analog read value	<10	46	65	83	103	124	142	162	180	200	221	240

Table 8. UV sensor GUVA-S12SD

5.3.6.4 VEML7700 Lux sensor

The VEML770 lux sensor is an I2C device. The device register addresses are located at addresses 00h to 06h (03h is reserved) while the write address is located at 20h and the read address is located at 21h. Using the reference code developed by Adafruit from the

Adafruit library we are able to configure our lux sensors. The software provided uses the sensor module to find 16-bit dynamic range for ambient light or Lux the SI unit ranging from 0 to 120k lux with resolution down to 0.0036 lux/ct. Within the software, we can adjust gain and integration times to the settings we find mind useful for the solar panel rotation. VEML7700 can memorize the last ambient data before shutdown and keep this data before waking up again, therefore we can take advantage of this for power saving.

5.3.6.5 Anemometer Wind Speed Sensor

Using the Adafruit anemometer wind speed sensor the F.A.S.T. main device monitors the wind speed in the area ranging from 0 to 32.4 meters per second. The output voltage of the wind sensor ranges from 0.4 V to 2.0 V. With a simple conversion equation, the ratio found is 0.049 volts per meter per second. We simply use an analog read function and configure our refence voltages to the values output by the sensor. The code stores multiple readings of the wind and find an average value to be output to the file storing the rest of the data from the slave devices. [45] This sensor has a very large power requirement and as such should only be turned on during readings in order to save power and maximize battery life. The last measurement should be stored on-board along with the time that the last measurement was taken in order to determine if a new measurement is needed. If enough time has passed between the current time and the last measurement, then the anemometer should be turned on and a new measurement should be taken.

5.3.6.6 Adafruit STEMMA Soil Sensor

Each of the child devices include the STEMMA soil sensor module which uses an I2C interface and includes the adafruit.h library functions to find the humidity of the soil the device is placed into. Using I2C, we read the data onto the individual device. The data is read at a 9600 baud to be stored with the data from the other sensor modules of that device. The sensor uses very little power even during measurements, so power saving measures do not need to be taken unless necessary. The measurement should be stored on the device until the next transmission is needed. The data is processed to ensure legitimacy before being sent via LORA to the main device in order to not confuse the user and provide invalid data. [46]

5.3.6.7 Ambient Humidity and Pressure Sensor

The BME280 module is an I2C device which takes a selected baud rate as well as a serial clock and serial data line. The BME280 sensor module can read the surrounding barometric pressure, temperature, and ambient humidity using I2C or SPI communication protocol. In this case we are implementing the sensor module using I2C though there are version with SPI capabilities. [37]

The humidity sensor provides a fast response time for fast context awareness application with high accuracy. The pressure sensor measures absolute barometric pressure with high accuracy and drastically lower noise than BMP180 or other similar sensor modules. Temperature sensing is optimized for low noise and high resolution and its output is used

for temperature compensation of the pressure and humidity sensors which can be used for estimation of the ambient temperature though we are using a separate device for temperature sensing. [47]

Because the temperature sensor on the BME280 is more accurate than the other temperature sensors, we use the BME280 for ambient temperature reading on each of the devices.

5.3.7 Solar Tracking Algorithm

Each of the VEML7700 devices all have the same I2C address, therefore a creative solution was needed to solve the reading of multiple lux sensors. Using the TCA9548A I2C multiplexer the device can read from four individual VEML7700 lux sensors that are required for the master unit's solar tracking ability.

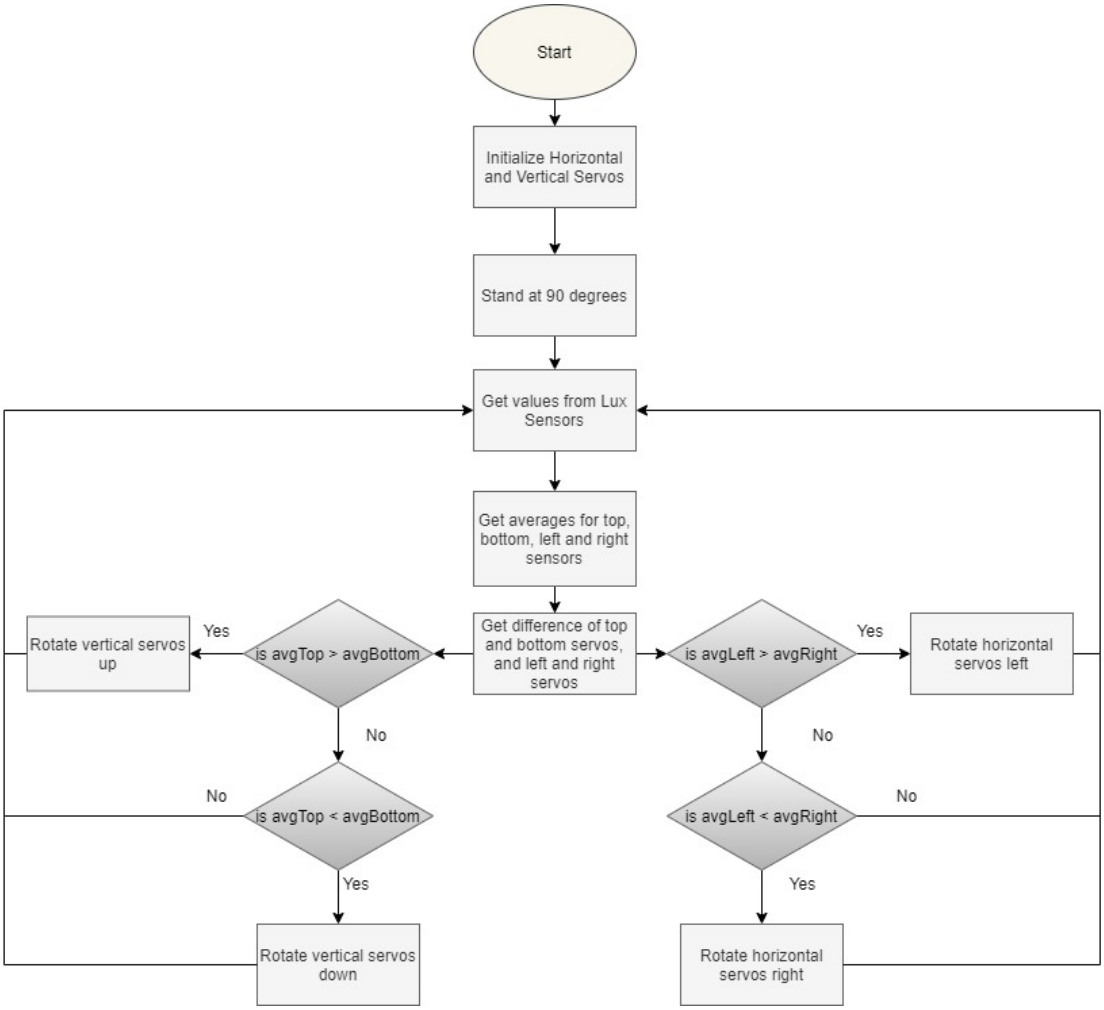

Figure 42. Solar Tracking Algorithm

The four lux sensors are named top left, top right, bottom left, and bottom right and are placed on the mount accordingly. After getting values from the lux sensors, the averages of the top, bottom, left and right sensors are calculated. Next, the differences between the top and bottom as well as left and right are calculated. The average of the top sensors will then be checked in order to determine if it is greater than the average of the bottom sensors. If the average of the top sensors is greater, rotate the vertical servos up. If not, check to see if the average of the bottom sensors is greater than the average of the top sensors. Rotate the vertical servos down if the average of the bottom sensors is greater. If not, the vertical servo remains in the same position. The same logic is used for the horizontal servo. **Figure 43** shows the flochart of the tracking algorithm,. [48]

5.4 Android software

Using Java, an android application is created with Android studio to show the data sent back to the main device from the LORA web system. The main F.A.S.T. device first confirms that the data incoming from the slave device is legitimate. Once confirmed, the is delivered to the Android application via the Bluetooth HC-05 sensor. This data is stored to a JSON file for easy extraction and separation by device and sensor type. Data collected from each device is show on the in-application user interface every minute when the new data is available. The user interface displays the specific data that each individual device collects (Only the main device will display wind speed and the direction the solar panel is currently facing).

5.4.1 User Interface

A user interface is created to display the data sent from each F.A.S.T. device **Figure 44** below shows the complete visual layout of the application. The main screen (also known as activity) displays each F.A.S.T. device as a clickable button. When a corresponding button is selected the activity is changed to a display screen showing the properties of the selected device such as, wind speed, ambient temperature, or humidity. The individual device activity display is scroll-able (with a finger swipe) to fit each piece of data sent to the main device. Next to the type of measurement to the output value currently being read from the sensor. The bottom of the application displays a refresh time, every minute when each device data is sent back to the main device the refresh timer is reset. This also means that the most recent data has been updated to the main device and is ready to be redisplayed on the user interface activity. Finally, the interface has a back button that can take you back to the main user interface or activity.

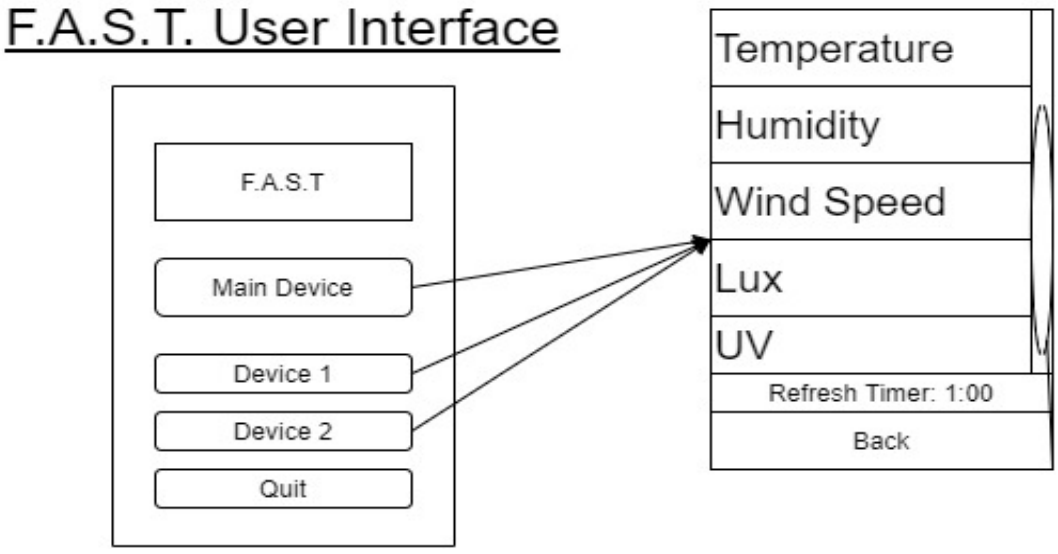

Figure 43. F.A.S.T. User Interface Design

The buttons on the main activity interface and the on the individual activity pages are controlled by on click listeners. When a button is pressed, the intended code runs to open the new device page. In each case the device can use the on click listener to open new activities to view the data incoming from the intended device, head back to the main activity screen, or control the scrollable list. The data view is updated dynamically with the clock of the refresh timer. The refresh timer runs on its own thread that follows when the data from the Bluetooth input has been refreshed. The application resets the timer each time the device receives new data via RX buffer from the module. Once the main F.A.S.T. device receives the new data, the user interface can be updated to show the new data.

5.4.2 Bluetooth via Android application

The F.A.S.T. device sends data to the android application user interface using the HC-05 Bluetooth module. Using Android Bluetooth library, the device listener coded into to the android application can send or receive incoming data to or from the application in the presence of the main device. We first separate the data accordingly, which is stored onto the main F.A.S.T. device. Since each device has unique data, we ensure the data is correctly separated for display. A thread runs that reads the incoming data as a byte packet while there still is incoming data, and this continues until all the data has been sent, or we see the signal end. The android device must first be paired with the main F.A.S.T. device to view the data, then it is displayed on the user interface. **Figure 38** below shows the flow of reading each byte from the output device after it has found and connected to the HC-05.

First the android device needs to be connected to the Bluetooth connection of the main F.A.S.T. device. The android application accesses the master device via a device name string that has been hardcoded to the HC-05 module which is just called “HC-05”. If the connection is successful, we then set the applications intent to connect and begin the data

listener. The data listener pulls data from the output stream in byte packets and stores each one on a read buffer until the next delimiter. The data is stored from each device to a JSON file for easy delimiting and then stored to the structure described in the data structure design section for easy access via the microcontroller. A bracket character indicates when the current piece of input is finished within our data output JSON file. This ensures that the correct device data is viewable on the user interface. When the current piece of data is finished the data is stored in such a way that the rest of the software can access this data. This data ise stored on an external read file in specific order that can easily be reread for output back to the application. Before the data can be displayed on the user interface it will be checked within the device's individual read limits to ensure the data is legitimate. Once, this has finished and the data is confirmed valid, a new thread runs to display the data via the android application. Each minute new data is created and stored to this output file from via the LORA signals. Then, a Bluetooth read function is implemented to wait until the refresh timer has finished. Once the refresh timer reaches 0:00 each new string of output device data is read and refreshed on the device's main display user interface shown above in **Figure 45**.

Data from each device is sent via the LORA connection and stored to a JSON file that can easily be read by the main device microcontroller. From there, the data is stored to structures that can be indexed for correct separation and displayed as intended and discussed above. Because we are implementing multiple LUX light sensors, we are only display the one with the highest value, or the one in the direction of the highest LUX value read.

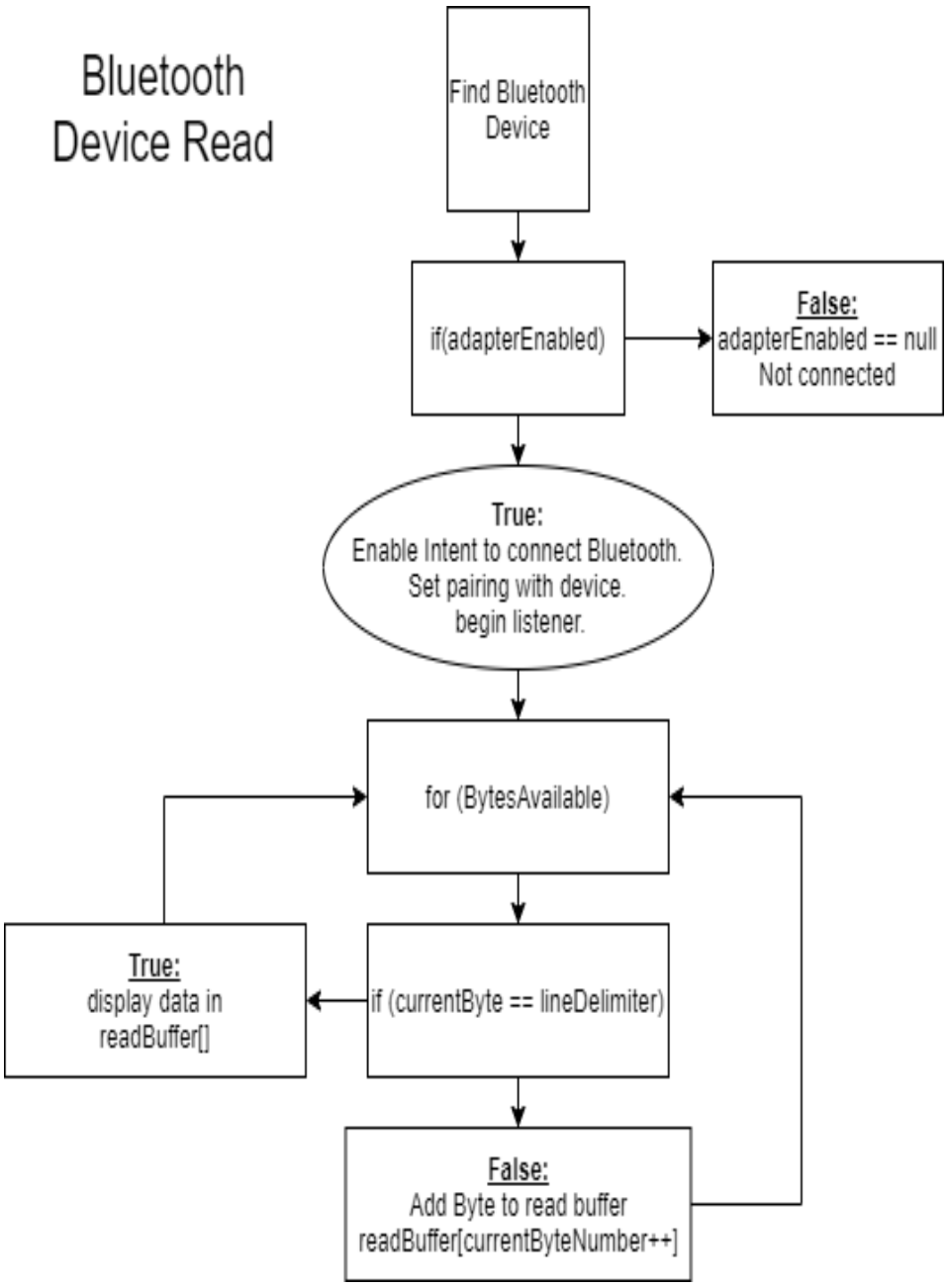

Figure 44. Reading Bluetooth bytes from main output F.A.S.T. device

LORA pushes data which is sent from each individual device back to the main F.A.S.T. master. The master device then pushes the data via Bluetooth to the android application. The application contains an interface that displays data from each individual device as well as the master itself.

A timer is not used to delimit each device reading but instead we are continuously grabbing data from each individual device as it is sent to the master device via LoRa, then outputted using Bluetooth UART and the HC-06 module.

5.5 LCD Module Software

To work with the LCD, the team takes advantage of the LiquidCrystal library which allows control of compatible LCDs such as the LMB162ABC. The goal is to output data from the different sensors and include as much as possible. Since we can only output 16x2 characters at a time we need to have a rotation. We choose to output one data at a time on the screen and this results in a clean looking format, however it is not very efficient. As an example, if the data we want to read is four iterations away then the user must wait twice as long compared to two different data available at a time. The goal is to be able to utilize the space in the screen and cut the user's wait time so that they can have access to desired information at a reasonable time.

To accomplish the issue of long wait time, we have two different sensor data available on the screen at each iteration. By having two different displayed on the screen at each iteration we carefully plan each of their positions. If we are not careful, we risk having displayed under the wrong name. For the formatting we have the top row of the LCD include the title of the data such as "Temperature" or "Humidity." This task is accomplished by using the lcd.print() function. This function is responsible for printing strings and is used for all the sensor titles printed on the first row. On the bottom of the titles is the matching data read from the sensors. For example, under "Temperature" is "85 F" and under "Humidity" shows "45%." To write data coming from the sensors to the LCD we use the lcd.write() command. This function is responsible for printing data that is read from the sensors.

We ensure that the formatting and spacing are all even. To accomplish this, we control the placement of the letters. We must always have a space between the titles as well as between the data values as to not confuse the users. For this task we use the lcd.setCursor() command. The lcd.setCursor() command is responsible for determining where everything is going to be placed. This command takes in both column and row coordinates. The column coordinates start from 0 to 15 and the row coordinates start from 0 to 2.

There sre instances where both the full titles of the sensors will not fit the screen since we are limited to 16 characters in addition to the necessary spacing in the middle. As a solution we use abbreviation for the titles that makes sense. If we want "Temperature" and "Humidity" both in the same iterations, then we output "Temp" and "Hum." There are instances where sensors will fail due to reasons such as the sensors breaking or poor connection. If the sensors fail, we output an error message below the failed sensor's title. **Figure 46** is the software flow diagram for the LCD.

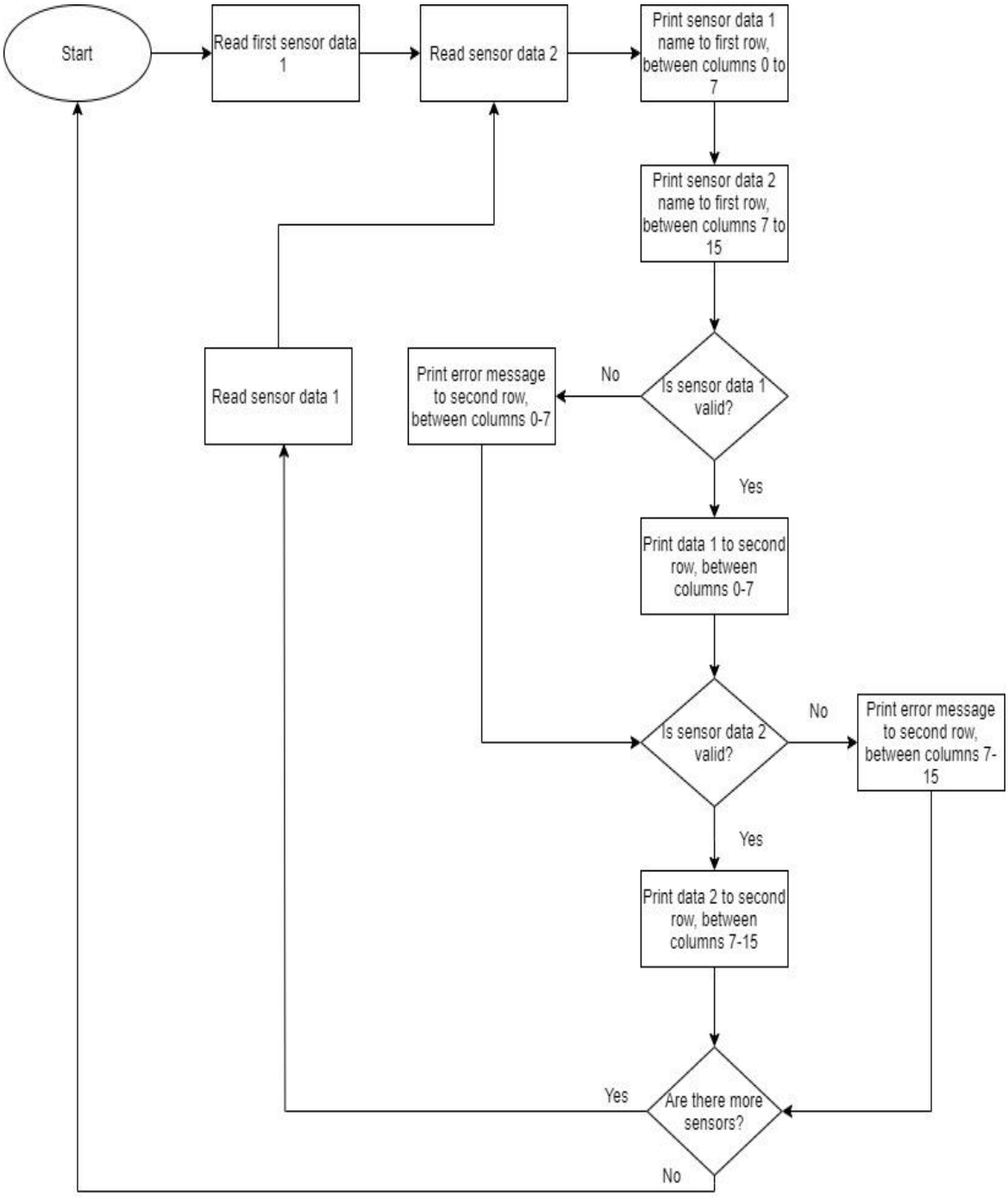

Figure 45. LCD software flow diagram

After testing we realized that using an LCD was more costly than just output everything with the serial monitors and our laptops as well as developing a simple android application and Bluetooth to display the intended data needed. Therefore, software testing for the LCD was not needed or used.

6 Testing

In this section we introduce the testing of all the modules used to create the F.A.S.T. device. Each sensor, module, and charging device was tested to ensure the data we are receiving is correct and it is the data or output we intended to receive from the device or that the correct power is being supplied to the device. Once we have tested and ensured the devices work as intended, we then add them to a confirmed parts list which is used to create the actual final design of the F.A.S.T. device.

6.1 Communication Testing

This project requires extensive and repeated use of several different wireless communication methods. Bluetooth communication is used to communicate between the primary F.A.S.T. unit in the F.A.S.T. mesh network and the android companion app. LoRa communication is used to implement the F.A.S.T. mesh network which facilitates communication between all F.A.S.T. units in the network. The F.A.S.T. mesh network facilitates the wireless communication by not requiring that each node be directly connecting to each other node in the network in order for results to be transmitted across the network to the primary unit.

6.1.1 HC-05 for Bluetooth

Testing of the Bluetooth module requires the use of a cellphone with a Serial Bluetooth Terminal application. This application displays the output of the Bluetooth HC-05 to a terminal much like using a terminal on a personal computer. To test that the module is working correctly, with the proper connections we send data from an ultra-sonic range finder that's data is displayed directly on to the phone's terminal. To test and run also, we have designed a simple test software that can read output data from the phone application terminal. This test software checks a serial read value or, a value that is sent via the Bluetooth terminal, if the input value is 1, an LED attached an output pin on the Arduino is turned on, if the input value sent is 0, then the LED turns off. These two tests ensure that the Bluetooth module is working properly. First by checking if data can transmit through the transmission line, then by checking if the HC-05 device is receiving the correct data through the receiving line to the Arduino that viewable with the change in LED status.

Once the HC-05 is connected to the F.A.S.T. device we are able to test the Bluetooth module using our android application. Displayed to the user interface should be similar data received when testing the sensor modules. To ensure the data is correct we move the device to different locations with changes in the environment that cause changes in the display data. As well, to ensure data we can inspect the national weather service website. Once Bluetooth is implemented and tested for correctness, we have as tested other sensor devices sent directly to the user interface on the android application. Once the correct data is displayed, we have completed testing of the Bluetooth module.

6.1.2 LoRa Range Test

Each F.A.S.T. unit was expected to connect to each other over relatively far distances. Each unit was tested to ensure that it met the range required to be considered long range. In order to test that the LoRa communication was working as expected with regards to the range requirements, a test program was loaded onto a child F.A.S.T. unit which simply broadcasted a message and waits for a reply. This a similar program was loaded onto a master F.A.S.T. which simply received the message and printed it to the console. First, the connectivity was tested. The team ensured that the LoRa modules were able to communicate in the first place before they tested their range. Once their connectivity was established, one unit was placed at a known location and the other unit was initially located next to the first unit. The second unit was then moved slowly away from the first unit while monitoring the serial output of the second unit. Once the serial output, ceased to show that the second unit's transmissions, then the test ceased and via GPS on our smart phones we determined how far the second unit had to be moved before the first unit lost contact. The test was considered a success because the connection was maintained for at least 200 meters. The test should be conducted under several different conditions. The first range test was first conducted on a beach and a range of 304.8m was observed between the units. The final range test was conducted at UCF near the Intramural Sports field. The range observed in the final test was approximately 500m and the test was ceased due to the team running out of available space to conduct the test. When the test was concluded the transmissions were still being successfully received.

6.1.3 LoRa Mesh Network Test

Originally, the F.A.S.T. network was going to be a mesh network and as such required testing. The original mesh network functionality of the F.A.S.T. network needed to be thoroughly tested because it was originally going to be one of the main features that this project implements. In order to test the LoRa mesh network, a miniature network was intended to be constructed. The mesh network was going to be implemented with several barebone LoRa nodes. These units would not have needed to contain all the final sensors or even the self-sufficient features. These units were going to be connected in a mesh network in such a way that the network was partially connected. The mesh network was originally going to be tested by sending data through the mesh network from one node in the network to another node in the network. The previous two nodes were required to not be directly connected and had to be instead indirectly connected through an intermediate network. This test was originally going to only test that the network can transmit data through itself.

Ultimately, the decision to use a mesh network for our F.A.S.T. network was scrapped, and a star network was used in its place. The reason for this change was one of practicality and reducing code complexity. A similar test was conducted to test the star network in which several child units were engaged, and a master unit was activated. The test was to be considered a success if the master unit was able to receive transmissions from all the child

units in the network. This test was conducted many times and each time the master was able to successfully receive transmissions from all child units consistently and without errors which gave credible evidence that the F.A.S.T. network was working successfully..

6.1.4 LCD Module Testing

We have tested the LCD to ensure that it is working as intended. To test that the LCD we output text, and we wrote a program that uses the command lcd.print() followed by a string we want to output to the LCD. After loading this program to the Arduino, we ensured that we are seeing the same string on the LCD. Once this test passes we know that our LCD is functioning and it is able to output text. We then tested the LCD outdoors to ensure that the brightness settings are acceptable outside. We then tested the LCD throughout the day and especially during the brightest part of the day.

We decided that using an LCD was more costly than testing the sensors and units with the serial monitor and the Bluetooth application therefore we left LCD testing out of the project development.

6.2 Mechanical Testing

This project has a few mechanical parts which were all thoroughly tested to ensure the functionality of this project. The mechanical aspects of this project are of equal importance to the electronic and software parts and was tested with the same vigor and thoroughness. Failure of any of the mechanical systems in this project will lead to failure of the overall project as well. Because of this the tests discussed in this section were designed and was used in the testing of F.A.S.T. project.

6.2.1 Dual-Axis Testing

To test the dual-axis system, test programs were loaded onto the F.A.S.T. units which go through a range of motions moving the panel across all the possible angles that will be expressed in this system. The test directed the servos to spin in such a way to demonstrate that the panel on the F.A.S.T. unit was able to follow the sun effectively across a normal day. This test ensured that the panel can be rotated a full 180 degrees and that it can tilt safely and with relatively high speed. The Dual-Axis system also provided consistent performance in outdoor conditions over long periods of time. The secondary test consisted of running a program on the F.A.S.T. unit that operates the dual-axis system over a long period of time to ensure that calibration was retained, and performance did not suffer.

6.2.2 Outdoor Material Testing

The materials used to build the structure of each F.A.S.T. unit have been thoroughly tested to ensure that it can withstand typical outdoor conditions for extended periods of time. Each F.A.S.T. unit is expected to be able to withstand outdoor conditions indefinitely due to the nature of the project. The material used to build the unit is adequate for such an endeavor. The material must be waterproof and resistant to extended periods of direct

sunlight. The main housing used in this project will contain sensors, the MCU, and all PCB must be waterproof to ensure that the unit is not damage during any event in which water falls upon the unit such as rain or accidental splashes. To test the waterproofness of the housing, the housing will be sealed as it will be during the final project and then sprayed with a garden hose for several minutes. The test will be considered a success if upon opening the housing no water is found inside. The material's resistance to direct sunlight along with extended outdoor conditions will be tested by placing a sample of the material outdoors for several days and checking its conditions after the days have passed. The material test will be considered a success if the material shows no measurable lose in durability or significant discoloration.

6.3 Sensor Testing

Each sensor module is individually tested using an Arduino uno using some simple test software for the specific device. We can first view the data output from the module via the serial terminal in the Arduino IDE. Then we use and LCD to display the sensor outputs, Finally, each sensor is tested using UART to the android application to see if all the correct data is transferred the way we have intended.

Since we need to test the different sensors in different environments we use and LCD to view the output data for each slave device. We individually test each sensor this way, displaying the data on the LCD with the Arduino uno. This makes sure that our code for each senor, as well as each LCD is effective in finding the information that we need. The sensors are individually tested to ensure the data they output is correct and converted the correct way by the software. In some cases, we are using the national weather service website to check if our incoming data from each of the slave devices and their sensors is relatively accurate. Some sensors such as the Adafruit STEMMA soil sensor require us to create a miniature experiment using a control to test and compare to the output of each of the sensors. These experiments ensure the data from each of the sensors is being read the correct way. Others require us to purchase a similar product to use and compare the products output such as the anemometer wind speed sensor.

6.3.1 VEML7700 Luminosity Sensors

We are using multiple VEML7700 devices to sense the luminosity of the area around the main device. Therefore, each sensor needed to be tested using i2C connection with the microcontroller. Each lux sensor is required to have a different address for reading and writing and we use these addresses to distinguish between them. We first individually test each of the sensors alone on the Arduino uno each output value from the sensors should be in the ranges between 0 lx to 120k lx. They can simply be tested to show the range by covering the sensor with a finger, and then placed close to a light. Each sensor shows similar output.

Once we have tested each of the sensors, we can combine the code for each to read all of them at the same time. Using their unique address, we are able to distinguish between them,

each is given a direction value associated with the address and this value is used to determine which way the solar panel should rotate.

6.3.2 GUVA-S12SD UV Sensor

Each of the F.A.S.T. devices hold a GUVA-S12SD which is used to detect intensity in ultraviolet radiation or UV intensity. The modules typical UV detection is between the wavelengths of 200 nm to 370 nm. These sensors are thoroughly tested to ensure the output data of each is correct. The first test we apply to ensure correctness is placing the sensor inside, which is read as a UV index of zero. In an outside shady area, the devices output values of the sensors will range from 3 to 5 which increases as we increase the amount of sunlight hitting the device directly. An extreme reading will be 10+ which will tend to happen in Florida at 12:00pm.

Below in **Figure 47** shows expected values of UV index in Orlando Florida on March 29th, 2020. We check the sensor output each hour and graph the data. We receive data like the **Figure 47** below if the sensor is outputting correctly. The last thing we have done to ensure our data is correct is to check a weather websites daily UV index graph and compare the output values of our sensor to the data provided by the site. The values start low, as the sun comes up, the UV index increases around midday,

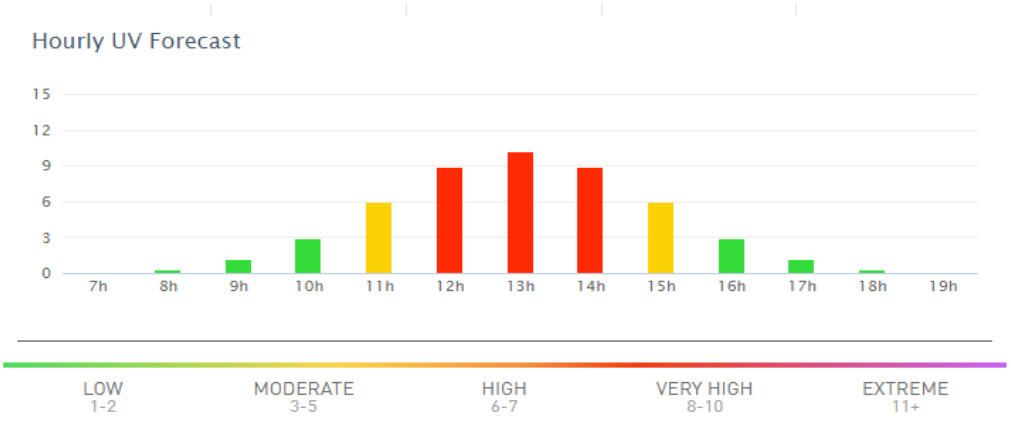

Figure 46. Typical Florida UV Index

6.3.3 BME280 Humidity and Ambient Pressure

Each of the F.A.S.T. devices contain a BME280 humidity and ambient pressure sensor. The unit the Arduino software outputs for pressure is of the unit pascal. This measures atmospheric pressure which is the measure of the weight of air molecules at ground level. Humidity from the BME280 is read as a percentage 0 to 100 of water vapor in the air (± 3% relative humidity). As humidity increases, pressure decreases.

To test the sensor, we compare the sensor output values which is viewable on the LCD display to values for the day provided by the national weather service website in Orlando Florida. Humidity from the sensor should have similar output for example, on March 29th,

2020 the humidity is 62% with the sensor reading a similar value [49]. As for pressure, most weather websites display the data in millibars and not pascal, we need only to divide the output pascal value by 100 to get the value in millibars. Once we have the value in millibars we can compare it to the national weather service data.

6.3.4 TMP36 Temperature Sensor Testing

We begin testing of the TMP36 temperature sensor which is first be connected to the Arduino individually and tested inside with the PC terminal. The output of the sensor is compared to the thermometer of an indoor A.C. unit which can be fluctuated and observed on the serial terminal. If these simple tests pass, we can proceed with outdoor testing. Outdoor testing involved using an LCD to display the output data. Using the data from the nation weather service website we test each of the temperature sensors individually. Each of the sensors output is compared to each other as well to ensure each sensor is getting to the correct output. In addition, the sensor is then pushed to different extremes to ensure the devices values fluctuate correctly within its limits. The last thing we do to test the temperature sensor is purchase a cheap thermometer that is designed for ambient temperature and compare our output values to the thermometers displayed temperature.

6.3.5 Soil Humidity and Temperature Sensor

Using the Adafruit STEMMA soil sensor we find the temperature and moisture of each F.A.S.T. device's surrounding soil. The output values for moisture range from 200 (very dry) to 2000 (very wet). Temperature output is in degrees Celsius (±2°C), then converted to degrees Fahrenheit. To test to ensure the output data from the sensor is correct, we are filling multiple (at least 4 or, the number of sensors we are using) pots of soil with moderately dry soil. One of these pots of soil is left alone as a control reference for the rest of the pots of soil. Therefore, all of the readings for each pot read the same or close to the same moisture value at first. We then add different levels of water to the soil to create different levels of moistures throughout each of the pots. Then, each of the moisture sensors is placed into the soil and the output values are observed and written down for each sensor. We then rotate the sensors down the pots of soil testing each sensor with each pot of soil and comparing the output values. The data for each of the pots of soils for each sensor should be similar, if not, a sensor may need to be replaced.

6.3.6 Anemometer Wind Speed Sensor

The Anemometer Wind speed sensor counts the cup rotations, or portion of the rotations over a period of time. A certain number of rotations over time equates to a read wind speed from the device. To ensure we are getting correct outputs we first connected the device indoors and spin the device to make sure the device is outputting. Then, we purchase an already assembled wind sensor. With the Anemometer sensor, and the already assembled device we compare the output values to ensure the data we are receiving from the sensor is correct. As well, we can compare the output values to the values of wind speed given by

the national weather service, though it will have error it has given us a good idea of the current wind speed we should be seeing output by the device.

6.4 Overall Function Testing

Once the individual parts of each F.A.S.T. unit had been tested, they were then tested in unison. This section will cover the tests were performed in order to ensure that each F.A.S.T. unit functions individually as expected and that the correct data is displayed to the android application user interface.

6.4.1 Average Conditions Testing

A single F.A.S.T. unit was placed outdoors under average conditions. This was performed on a relatively sunny day, with low to moderate wind speed along with no rain. This test will simply checked to ensure that everything is wired up properly and all sensors are communicating properly. First, the solar tracking was tested. This test passed, when the solar panel follows the sun in such a way that power efficiency is maximized. Second, the F.A.S.T. unit's sensors were tested in unison to ensure full functionality. External sensors were used to verify the values that the F.A.S.T. unit was recording. A laptop was used to monitor the values read by the sensors through UART and the values were verified through the F.A.S.T. companion app.

6.4.2 Non-Ideal Conditions Testing

Originally single F.A.S.T. unit should be placed outdoors under non-ideal conditions. The test should be mechanically similar to the test described in section 7.4.1, but it should be under different conditions. Conditions such as low visibility should be tested along with conditions such as rain. The F.A.S.T. units must be tested in these Non-ideal conditions to ensure that the unit will function properly and give useful data even when conditions are not favorable. A rain test should be conducted to ensure that the onboard sensors still provide meaningful data even during rainfall. The rain test will also ensure that sensitive electrical equipment is properly shielded and protected from rain. A F.A.S.T. unit should also be placed outside during high winds to ensure that all parts of the unit are properly secured and can withstand a reasonable amount of high winds. The output of the sensors that are included in each F.A.S.T. unit should also be verified to ensure that all sensors are provided correct and useful data even during high wind conditions. If weather conditions never occur that would facilitate the above conditions, then the previous conditions may be simulated. High windspeed may be simulated through the use of a leaf blower or an industrial fan. Rain may be simulated through the use of a garden house with a shower nozzle. While the test above would have been valid, due to the lack of weather proofing that took place a non-ideal condition test was not taking other than to test the output of the solar panels in non-sunny conditions.

6.4.3 Mesh Network Testing

Originally, this project was going to implement a mesh network, so the following test was derived. Multiple, already tested, F.A.S.T. units must be deployed outdoors and allowed to connect to one another through their LoRa modules and antennas. The units must be connected in such a way that not all nodes are connected to all other nodes in the network. Ideally, a primary node will connect to a middle node and this middle node will connect to a distant third node. The mesh network will be tested by ensuring that the farther node's readings are reaching and being transmitted to the middle node and to the primary node. The readings should be verified with external third-party sensors to ensure that sensor data is not corrupted or altered while it is traversing the mesh network. Ideally, the test should also include monitoring of the data being sent and transmitted through each node in the mesh network by connecting laptops through UART and printing out each transmission message to the laptop for verification.

However, a star network was implemented instead. This star network was tested in a very similar manner than the one proposed above. Multiple, already tested, F.A.S.T. units were deployed outdoors and powered on. A star network with four nodes was assembled using one master units and three child units. The units were separated by a significant distance and main unit was connected to the app. The point of this test is to ensure that the star network is working as intended. When the app was activated the app was receiving measurements from all the units in the star network which confirmed that the star network was fully functioning.

6.5 Power System Testing

Testing the entire power system was the only real way to ensure that the F.A.S.T would be able to independently power itself in most conditions. To test the overall system including the photovoltaic module, battery, charge controllers, and load it was important to run individual tests on each component before testing the system. This ensured that any unusually high losses or malfunctioning components could be sorted with before trying to judge the whole power system. It was also important that certain components are not only tested together as especially in the case of connecting the photovoltaic module directly to any load can cause issues do to the difference in startup current. Ultimately, testing individual components also gave a better profile of overall losses and reduction in efficiency for the system as whole.

The most basic test for the overall system was connecting the photovoltaic module to a charge controller which was then connected to a battery. We then placed the photovoltaic module in full sun and record temperature and relative humidity ensuring that the battery is able to receive significant charge in most "normal conditions". Then using this battery that received a standard charge from the photovoltaic module, we then tested how long the load can run off the battery power. More specifically we required voltage regulation from the battery to the load for similar reasons as why a charge controller is needed to connect the photovoltaic module and the battery. Even though the actual F.A.S.T. schematic required to different loads just to test that the overall system can power a load

at all is to have a 5V .1A load. Once this power system was able to safely supply the load, we then added the voltage regulators and their respective loads that they are stepping to.

6.5.1 Photovoltaic Module Testing

For this project it was important to test the photovoltaic module in different testing conditions to ensure it had the capacity to supply adequate power to the F.A.S.T. The main test was measuring total power output from the power module over the course of the day. As discovered in our research the efficiency of photovoltaic modules varies greatly depending on ambient temperature, moisture, and most obviously total daylight hours. For one test the photovoltaic module was tested in direct sunlight and the temperature and relative humidity will be recorded every hour. We continued to do this test on a day with different weather to receive sufficient data.

Another test consisted of placing the photovoltaic module in areas that are not in complete sunlight. We looked at the profile of the location to determine the total area of sunlight that is on the photovoltaic module. Similar to the previous test the temperature and humidity were recorded on the hour so that the data can be graphed to observe total power output in different conditions. It is likely that the F.A.S.T will not always be in an area of direct sunlight as crops may cast shadows on the weather station.

Another test involved charting how much voltage and current the panel produce in different corresponding levels of sunlight.

6.5.2 Tracking Testing

Originally it was important that the tracking feature makes sense to implement in terms of efficiency, so it was important to test it extensively with the photovoltaic module. The best way to accurately test whether tracking made sense to implement was to test the photovoltaic and tracker combo in a similar situation as the photovoltaic module was tested alone.

The first test would have consisted of the tracking photovoltaic module in direct sunlight and then taking temperature and relative humidity. To make the results of this test meaningful they would have needed to be compared to the solo photovoltaic module test with the similar day to day conditions.

Solar tracking in shady areas is where we anticipated to see the greatest efficiency gain as the solar tracking can optimize the amount of sunlight over the photovoltaic module. However just because the efficiency would likely increase in these situations does not mean it would be enough to justify using power to move the module. Tracking would also create an extra current demand from the battery as the load would increase as the servos needed power. The thermal repercussions of this could make the feature not worth it after testing.

We could then continue to utilize data from the F.A.S.T sensors to establish a set of conditions where it is advantageous to use solar tracking and then other situations where it is not efficient in terms of power used to move the module versus gain achieved. For example, on geographic situations, like the shadow of a build makes no means for the module to get full sun even on a clear day, then it could be impactful. The modules who would only need few correcting movements over the course of the day would have the most to gain. However, from a software standpoint it would be somewhat difficult to anticipated correctly what days would be impactful unless it took weather data from an outside source or was connected in an extremely vast mesh network making it a feature we did not include or test in our project do to design and budget constraints.

6.5.3 Battery Testing

Similar to the photovoltaic module testing the battery was essential to determine whether or not the F.A.S.T. would be able to sufficiently power itself without support of the grid. It was important to get an accurate capacity of the battery along with also getting a sense of it degradation with charging and discharging. To test the overall capacity of the battery we simply used a DC power source and record the output voltage and current from the power source. We then charged the battery for the length of time that would fill its 3000mAh capacity. We then drained the battery measuring the output. From this we were able to understand not only the total battery capacity, but we also gained a sense of any efficiency losses if there are any.

The other test consisted of charge and discharging the battery and recording the total power in and the total power out. After this test was run enough times it we saw the battery have less overall capacity and maybe less efficient discharge. We compared these values to the baseline discovered in the first test. It was possible to add more batteries to supply the F.A.S.T. using an expandable battery holder. We decided we wanted around 6000mAh after this test.

Another test that the battery was put through was to test how susceptible it was to charge degradation as the battery will charge not to full capacity and then discharge completely. This test was important as there could be scenarios where the battery was completely drained after the solar panel can no longer provide power. We ran this specific test on a new battery as the second test could have influenced the outcome.

The final test was charging and discharging the battery irregularly. This was the best to simulate the charge and discharge profile of the battery attached to the F.A.S.T. These results were then be compared to the baseline efficiency and total capacity in the first test to see what toll normal use will take on the battery. The data was then used to extrapolate how long the battery could last before needing to be replace. Since it was one of the specifications that the F.A.S.T. will be able to independently hold power but also be low maintenance these test results allowed us to pick two 3000mAh batteries for a totally capacity of 6000mAh.

6.5.4 Charge Controller

The charge controller allowed the photovoltaic module directly to charge the battery. This device acts as a voltage regulator but also has a built-in capacity to sense when the battery is reaching full charge and to no longer allow the photovoltaic module to charge the battery. This device is crucial for battery longevity. Test for the charge controller started by measuring its efficiency. The lab generated power source was inputted into the charge controller and the charge controllers' output was used to measure any losses that came from the charge controller itself. The charge controller was also tested using the photovoltaic as the source in a similar set up. The efficiency varied between the photovoltaic source and lab source so extra losses needed to be accountable for having a less robust power source. One of the main features of the charge controller was the ability to stop the battery from overcharging. This feature was tested with batteries who have their capacities specifically mapped for under their own test. We then measured the output from the charge controller connected to the source and made sure it sensed when the battery was at capacity accurately.

6.5.5 Voltage Regulator Testing

Testing the voltage regulator was very similar to the charge controller as they are essentially the same device, but the charge controller has added features. It was important to get an understanding of the overall efficiency and potential losses. However, the voltage regulator should have been printed directly onto the PCB so we needed to re-create the regulator and test it via a bread board. Using a breadboard and a in lab power source to simulate the photovoltaic we measured the output from the switch and linear regulator in series to establish the efficiency of our hybrid regulator.

Since this device needed multiple different input voltages it was important to also test the voltage regulators working in unison. We also discovered vulnerabilities related to the increase in discharge current to the battery. It was important that the battery does not overheat, and rapid discharge contributes to a less stable battery. Another possible result of the increased current drawn from the battery was the downgrade of overall efficiency. All these factors were test and the original PCB design was found to be unsafe so in the interest of time a through whole PCB was used to create a more stable connection between modules for the mower system.

6.5.6 Load Testing

The load we tested was the microcontroller completely configured with all the sensors. It was difficult to anticipate how much power the load without physically measuring it. There was also likely be changes in the software that would allow for the load to have more of a dynamic draw. Since one of the requirements was that the data would be able to be real time the microcontroller would likely use less powered then what we measured in this test.

The first test for the load was measuring the amount of power required for the device in low power mode. The second test was measuring the amount of power required for the

device in its active power mode. With these to metric as dynamic load profile could be establish as from a software standpoint, we were able to estimate over a span of time how much time the device will spend in each of its power modes.

While the first two test helped us establish a general idea of the load requirement there was nothing that can beat measuring the power required for load toggling between low power and active power modes on its own accord. After we implement the software, we measured the total load as well as the time it spends in low power and active mode so that it can be compared to the estimated amount in the first two tests. This test proves that the first two test are accurate, and we were then able to uses the finding from the first two test as the software changes without having to test the specific case over again.

6.6 Software Testing

In this section we discuss the testing of software. Each F.A.S.T. device has a unique processor and other hardware components. Therefore, each device has its own software requirements whether that be, controlling of the sensors attached to the device, control of the power systems and solar panel, or development of the android application user interface. Each devices software is needed to first be tested to ensure correct data output, as well as proper management of this data for correct communication between the device which includes, timing of data distribution. Finally, the developed android application requires testing to guarantee proper device connection, easy user use, and legitimate incoming data references.

6.6.1 Arduino C Software Testing

Software for each devices ATMEGA328P's is tested with an Arduino Uno. We individually test each sensor using the Arduino uno and the code to correctly make use of the sensors or other components. Then, the software for each sensor is combined to test the system of sensors. The ATMEGA chip can be replaced inside the Arduino and programmed to ensure everything works properly.

The chip can then be directly placed into our designed PCB. Once the PCB is designed, we have tested each processor with its stored software in its specific device PCB. From there, we started testing the individual software for each device. Using an LCD display, we output the data to be viewable to ensure the device is outputting everything correctly. The software for the LCD is similar for each device, but the sensors may be different. The LCD displays the data for one sensor at a time, that will swap display information after a few seconds.

Analog input is tested using the TMP36 temperature sensor and the GUVA-S12SD UV sensor. Using the analog read function we just need to set the output of the sensor to the pin we direct the function to read from. This value is stored into a float value, sensorReadValue. We then apply the conversion equations defined in the software design section 5. Which converts the analog input to a real temperature value ranging from (-40°C to +125°C). If the outputs are outside this range, there is some sort of issue.

I2C is tested using the BME280 pressure and humidity sensor. Baud rate for the Arduino uno is set to 9600, and if the connection is properly set up, we should receive serial output values to the Arduino monitor within the devices output ranges 300 to 1100 hPa and 0 to 100% humidity and not random ascii characters.

UART is tested using the Bluetooth HC-05 sensor, as shown in section 6.1.1 HC-05 Bluetooth testing we need to test the devices UART capabilities using an external terminal application on an android device. We use a baud rate of 9600 on the terminal as well as on output from the microcontroller to ensure the data timing is correct. We test the UART output from the controller to the android device with the TMP36 temperature sensor. The data from the TMP36 temperature sensor tests the output and should display directly to the terminal with the serial.print function if baud rates are the same and everything is correctly connected.

SPI is tested using the RFM69HCW LoRa module itself. The module functions through SPI and this project already have multiple of these modules on hand. The SPI protocol is integral because it is what the ATMega328p interfaces within order to communicate throughout the LoRa mesh network. Two F.A.S.T. units are loaded with a test program and each of the units will have a RFM69HCW LoRa module wired up to the SPI pins on the MCU. The test program simply sent a “Hello World!” message from one unit to another. This test ensures that the SPI protocols on both of the ATMega328p ICs are working properly because to send a message through the RFM69HCW module one must use SPI and to receive a message one must also use SPI as well. This test ensures that the software that operates the SPI is functioning properly as well.

6.6.2 Android Application Software Testing

First, for android application testing we have ensured that the Bluetooth connection is confirmed between the cellular device and the HC-05. Therefore, the application will test that a connection has been established, we simply output via toast that a connection was established correctly. Once established, we output the data of one BME280 humidity and pressure sensor to the application user interface. A thread needs to run within the application to update the new data to a label as it comes into the application. This tests that the data is being output through the device correctly. The rest of the data is sent to the application in the same way so testing one sensor should be efficient in seeing that the data is being sent correctly. Lastly, we test that the JSON file is sent via UART to the android application and the formatting is correct for easy reading.

The next part of the application to test is the different elements of the user interface. We need the buttons to take us to new activities that display the data that we previously tested. We test all of the on click listeners that call functions to ensure we are taken to the correct unique device activity page. Once on the new activity page, we have tested that if our list is greater than the displayable view, the scroll bar correctly scrolls to display the data that is currently not if view. We simply add rubbish data to the scrollable list, at least enough to create a list large enough that we need to scroll through them. The last thing we needed

to test is the reset timer, which resets when new data is ready. A timer thread function runs to show and count down the timer. Testing this requires us to watch the timer countdown to 0 seconds and then reset again back to 60 seconds. If the timer its 0, the thread that updates the user interface every second is re-ran.

Next, the back buttons on the individual device screen are tested. They take us back to the main screen where we can then, click another device. Another device has been tested to make sure that the data is correctly displayed after we have already previously viewed a different device.

Finally, the android application needs to correctly delimit and read the data from the JSON file described in section 5.2.5.3 JSON storage example that is built between each F.A.S.T. device. Once the file has been received by the main device it is sent via UART to the android application. We then ensure that the data has been received by the android application, each JSON file has a unique formatting for consistent reading and writing, therefore it is the same format each time. If the data is being sent and delimited correctly, it displays via the user interface in its proper position.

The application and was created a little differently in that instead of using separate buttons to view different devices screens, we just used a scrollable fragmented application. Though testing was performed in a similar manner with displaying nonsense data to the application to ensure it is all being sent via Bluetooth correctly.

7 Administrative Content

In this section we discuss the administrative aspect of our design project. Administrative content is essential to any engineering project as it allows for the team to be on the same page. It also holds the team accountable as milestones had listed dates. In the first section, we will discuss the different milestones we completed and include dates for when the milestones were expected to be completed. In the next section, we will discuss our budget plans as well as discuss how we plan to spend that budget. Administrative content is often overlooked as a necessary tool to complete an engineer project that meet all standard and specification requirements. The bill of materials as well as a general timeline for project completion are discussed in this section. Creating a bill of materials and the general project timeline are a necessity for any well-organized project and have a further benefit of providing a template of sorts for future revisions of this project to steam from.

7.1 Milestones

To ensure the success of our design project, the team completed certain milestones throughout the 2020 Spring and Fall semesters. We were given two semesters to complete our project which means that by the end of our second semester, our project should have been fully completed and working properly but due to constraint some features were cut. The project had to be already been designed, implemented, and tested. Milestones allowed for us to be accountable as well as itemize project, but unforeseen circumstances prevented some finished systems. The divide and conquer mindset towards milestone lead us to more productive work.

Senior Design I consisted of mostly researching, working on the design, and writing the document. In the beginning of the semester we had to choose our team members as well as decide on a project idea. The biggest milestones for this semester are the three different project report submissions. Each member contribution of written pages must at least be thirty and must be met in individual increments before the milestone due dates. The increments of pages must be finished the day before the milestone is do so that the paper can always be formatted before submission. The rest of the semester consists of weekly meetings and completing tasks to make the due dates. Towards the end of Senior Design II, breadboard testing and prototyping is to be performed and design finalization should be made. **Table 10** includes the milestones for Senior Design 1 along with their start dates and end dates.

Senior Design II consisted mostly of implementing the design, testing the design, and editing the document as needed. Major design decisions were not permitted by this point and only slight designs revisions and design fixes were performed by this point. By the end of this Senior Design II we had our project completed and ready to be showcased along with presented to the best of our ability. **Table 11** includes the milestones for Senior Design 2 along with their start dates and end dates.

SD1 Task #	Milestone (Task)	Start Date	End Date
1	Get project approval	1/17/2020	1/17/2020
2	Finalize parts list	1/20/2020	1/31/2020
3	Divide and Conquer Version 1	1/20/2020	1/31/2020
4	Divide and Conquer Version 2	1/31/2020	2/14/2020
5	Project Report (60 pages)	1/31/2020	3/20/2020
6	Begin basic PCB design	3/20/2020	5/31/2020
7	Begin software/embedded development	3/20/2020	5/31/2020
8	Order essential parts	4/3/2020	5/31/2020
9	Build breadboard prototype	TBD	5/31/2020
10	Ensure functionality of breadboard prototype and perform design revisions as necessary	TBD	5/31/2020
11	Project Report (100 pages)	3/20/2020	4/3/2020
12	Manufacture PCB	TBD	5/31/2020
13	Assemble the PCB prototype and test	TBD	5/31/2020
14	Ensure functionality of PCB prototype and perform design revisions as necessary	TBD	5/31/2020
15	Project Report (120 pages)	4/3/2020	4/21/2020

Table 9. Senior Design I Milestones

SD2 Task #	Milestone (Task)	Start Date	End Date
1	PCB corrections or revisions	TBD	TBD
2	Fix any remaining issues available	TBD	TBD
3	Lock in scope of project as well as design	TBD	TBD
4	Manufacture final PCB	TBD	TBD
5	Finalize code and code documentation	TBD	TBD
6	Assemble final project unit	TBD	TBD
7	Test project unit and make any repairs or corrections	TBD	TBD

Table 10. Senior Design II Milestones

7.2 Budget Analysis

Since our project was not sponsored, the budget for this project came directly from the team members. The team has decided to keep the budget under five hundred dollars per person and this goal was achieved. To ensure that we didn't go over budget but still maintain overall project quality we needed to pay special attention to areas of unnecessary cost such as product housing, power supply, and sensor accuracy trade off. Ultimately, we decided if the budget will be exceeded to maintain the overall quality of the product if needed. It was also important to limit the cost of the overall materials so that if the F.A.S.T. ever did go to market it would have a relatively low cost of production so that it can be sold to all socio economics level who would benefit. However, since we are developing a working prototype, we didn't factor the initial research and development cost into the cost to make each individual unit, just the actual cost of the parts. **Table 12** below shows a breakdown of the different expected components needed to complete the project. Also included in the table is the price and quantity of each component. The final row displays the tentative total amount.

After comparing and researching MCUs which has I2C capabilities as well as SPI and UART, the ATmega328P has been chosen. The team allotted $10 for the MCU and the Atmega328 chips sell for about $3 a piece online. With $10 we could buy 3 Atmega328P chips. Since our project is a solar tracker, a solar panel is a must. The team has chosen to use a 5 Volt 2.5Watt solar panel. These solar panels come in pack of two and cost around6 dollars per unit. This low-cost solar panel in conjunction with some lower cost batteries made the F.A.S.T. well within the range of affordability.

We also needed a battery for our device. For a battery the team has decided to use a rechargeable 18650 lithium ion battery. Lithium batteries have a small form factor and will be able to charge and recharge without degrading too much. The amount capacity required for this project will be specified further after testing, but batteries will be less than 3 dollars for a battery. A 6000mAh will satisfy our rough calculation to power the weather station and it cost around $3 dollars so a pack of 4 will be purchased for $9 dollars and an

expandable battery holder can be used in the even one single battery does not meet the power requirement for the F.A.S.T . A solar charger controller was also needed. This automatically managed the working of the solar panel and the battery. For this component we have chosen a TP4056 charge controller pack from Amazon. This charge allows the battery to be charged from the solar panel directly and monitors the temperature and battery voltage, so no harm comes to the battery. The voltage regulators used to create the two different VCC lines will be printed using the PCB. The regulators should be relatively high efficiency but also low area profile. It would be better to pay slightly more for more space as a large PCB will cost more than most smaller components. The team also needed to budget for a photovoltaic panel.

F.A.S.T also included several sensors. To track the sun, we needed about five lux sensors. The team has decided to use the VEML7700 from Adafruit which costs $4.95 a piece so the lux sensors in total will be about $25. Another sensor we included is a UV sensor module and we have decided on Adafruit's GUVA-S12SD. This sensor can detect the 240-370mm of light and it costs $6.50. A soil sensor was also needed, and we have decided to get Adafruit's STEMMA Soil Sensor which cost $7.50. We also wanted our device to be able to measure wind speed. To achieve this, we have decided to include an Anemometer Wind Speed Sensor with Analog Voltage Output which cost $44.95 on Adafruit. Since our device will act as a weather station, we need a temperature and humidity sensor. For this we decided to get an AM2320 Digital Temperature and Humidity sensor which costs $3.95. For building our housing, we decided to spend around $50 on building materials. For servos that were used to angle the panel, we allotted $40 in total. We also allotted $100 for any miscellaneous cost which includes replacing parts, shipping, and any other surprises.

The housing for the F.A.S.T. remained undecided. However, we have allotted a decent amount of money such that we can do some research and development. Currently the research conducted has shown that 3D printing will likely be the best way to implement the housing for the device. There will be cost associated with filament material required, along with using 3D printer, and we may possibly need to outsource some of the design if we are unable to properly design the housing. With all of these things considered we have budgeted $50 for researching different materials and actually housing the idea. Due to time constraints the housing did not end up getting fully developed.

The F.A.S.T. network depended on implementing LoRa functionality through the use of LoRa transceivers. For this purpose, the RFM69HCW Wireless Transceiver was selected and has an associated cost of $5.95 per unit. Each F.A.S.T. unit will require one LoRa transceiver. The transceivers all required an adjustable 434MHz external antenna. These tend to cost around $5.00 each. Three units wer made, the LoRa functionality ended up costing a total of $32.85 for the entire project which considering the extreme range granted by the LoRa functionality grants a very good price-to-performance metric.

The F.A.S.T. bill of materials has a section for miscellaneous cost. This cost will include any minor additional cost that will occur in the product. This area also includes our budget for printing the PCB. This means that when choosing components, we have to factor in the number of components used and area they take up so that we can stay within our miscellaneous cost budget. The cost of the PCB was very high so only one module was

made with a PCB and the other modules were made with a very small PCB with less features just to demonstrate the mesh capabilities.

A expected bill of materials is shown below in **Table 12**. This bill of materials is for a single F.A.S.T. unit. If multiple F.A.S.T. units are required, like in this project, then the bill of material should be multiplied by the number of F.A.S.T. units required. This unit-specific bill of materials will allow anyone who intends to reproduce this project to easily determine what their overall network cost will be by providing the individual unit cost. Sensors can be easily added to this project which will increase the cost, but they can also be removed which would reduce the overall cost of the project. The bill of materials provided is intended to be flexible and adaptable to the user's needs and project's requirements. However, there are certain components which are necessary and cannot be omitted such as the RFM69HCW LoRa module, the ATMega328p, the Bluetooth module, the battery, and the solar panel. Besides these components, new sensors may be added with relative ease. Overall we ended up exceeding the expected cost but stayed within our budget.

Component	Price	Status
ATmega328p DIP	$4.30	Acquired
Solar Panel (2 count)	$12.99 *2 = $25.98	Selected
Lithium Polymer Ion Battery (4 Count)	$9.90	Selected
Battery Holder	$6.99	Selected
TP4056 Charging Module (10 count)	$8.78	Selected
On/Off Switch (30 count)	$7.99	Selected
VEML7700	$1.95 * 5 = $9.75	Acquired
TMP36	$1.48	Acquired
GUVA-S12SD	$7.48	Acquired
SPT5525LV-360 (Servo)	$13.50 * 2 = $27.00	Selected
LMB162ABC (LCD)	$8.00	Acquired
Building materials	$50.00	Research
STEMMA Soil Sensor	$7.50	Selected
Anemometer Wind Speed Sensor	$45.00	Selected
BME280	$6.55	Acquired
RFM69HCW Wireless Transceiver - 434MHz	$5.95	Acquired

434MHz Antenna	$5.00	Acquired
HC-05 Bluetooth Module	$7.99	Acquired
Assorted Resistors and Capacitors	Minimal	Selected
Misc costs	$100	Research
Tentative total:	$345.64	

Table 11. Quantity and Cost of Hardware Components for a Single F.A.S.T. Unit

8 References

[1] D. Charles, "Farmers Got Billions From Taxpayers In 2019, And Hardly Anyone Objected," 31 December 2019. [Online]. Available: https://www.npr.org/sections/thesalt/2019/12/31/790261705/farmers-got-billions-from-taxpayers-in-2019-and-hardly-anyone-objected. [Accessed 15 February 2020].

[2] "The impact of climate change on smallholder and subsistence agriculture," 11 December 2007. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2148357/. [Accessed 2020 10 February].

[3] "Effects of environmental change on population nutrition and health: A comprehensive framework with a focus on fruits and vegetables," 28 March 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5814744/#ref-72. [Accessed 10 February 2020].

[4] C. Parenti, "Flower of War: An Environmental History of Opium Poppy in Afghanistan," Johns Hopkins University Press, 2015.

[5] G. Fifer, "The truth about feeding 9 billion people is millions will have to feed themselves," 25 September 2014. [Online]. Available: https://www.theguardian.com/global-development-professionals-network/2014/sep/25/subsistence-farming-chad-food-security. [Accessed 20 February 2020].

[6] G. Doran, "SUSTAINABLE AGRICULTURE IN CHAD," 31 January 2018. [Online]. Available: https://borgenproject.org/sustainable-agriculture-in-chad/. [Accessed 17 February 2020].

[7] A. Sentosa, "Arduino Weather Station," 2016. [Online]. Available: https://www.instructables.com/id/Arduino-Weather-Station-AWS/.

[8] "Ambient Weather WS-2902A Smart Weather Station with Wifi Remote," 2020. [Online].

[9] "Fixed Tilt Solar Mounting Racks," Omco Solar, 20 September 2019. [Online]. Available: https://omcosolar.com/fixed-tilt-solar-mounting-racks/. [Accessed 5 March 2020].

[10] D. McGee, "FIXED-TILT VS. AXIS TRACKER SOLAR TRACKER," [Online]. Available: https://www.kiewit.com/plant-insider/current-issue/fixed-tilt-vs-axis-tracker-solar-panels/. [Accessed 15 March 2020].

[11] C. Juda, "5 Ways to Track Your Solar Tracker," 18 January 2013. [Online]. Available: https://blog.pepperl-fuchs.us/blog/bid/253098/5-Ways-to-Track-Your-Solar-Tracker.

[12] "Argometeorology: Wind and Plant Growth," 2016. [Online]. Available: http://agritech.tnau.ac.in/agriculture/agri_agrometeorology_wind.html. [Accessed 10 March 2020].

[13] B. G. Bareja, "Light Intensity in Plant Growth and Development," cropsreview, April 2011. [Online]. Available: https://www.cropsreview.com/light-intensity.html. [Accessed 17 March 2020].

[14] B. Jovanić, B. Radenković and M. Despotović-Zrakić, "High pressure effect on photosynthetic properties of green plant leaves," *Russian Journal of Plant Physiology*, vol. 63, p. 85–91, 2016.

[15] S. K. F. ,. S. C. L. J. Aakansha Chadha, "Influence of soil moisture regimes on growth, photosynthetic capacity, leaf biochemistry and reproductive capabilities of the invasive agronomic weed; Lactuca serriola," 2019 June 2019. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0218191. [Accessed 4 March 2020].

[16] C. Karpfinger, "Colin Karpfinger," Colin Karpfinger, 27 July 2010. [Online]. Available: http://colinkarpfinger.com/blog/2010/the-dropouts-guide-to-antenna-design/. [Accessed 30 March 2020].

[17] "Types of PV Systems," Florida Solar Energy Center , [Online]. Available: http://www.fsec.ucf.edu/en/consumer/solar_electricity/basics/types_of_pv.htm. [Accessed 20 March 2020].

[18] "Monocrystalline vs Polycrystalline vs Thin film Solar Panels," exsolar, [Online]. Available: https://www.exsolar.co.za/mono-vs-poly-vs-thin-film-panels.html. [Accessed 20 March 2020].

[19] K. E. Y. H. ,. W. W. E. D. D. Martin A. Green, "Solar cell efficiency tables," 20 December 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/pip.2573. [Accessed 20 March 2020].

[20] O. T. a. A. V. AyGegül TaGçJoLlu, "A Power Case Study for Monocrystalline and Polycrystalline," Hindawi Publishing Corporation, 23 January 2016. [Online]. Available: http://downloads.hindawi.com/journals/ijp/2016/7324138.pdf. [Accessed 20 March 2020].

[21] [Online]. Available: https://www.amazon.com/Newpowa-Watts-Solar-Module-Marine/dp/B00W813E4I/ref=sr_1_3?dchild=1&keywords=solar+panel+20w&qid=1585586995&sr=8-3. [Accessed 20 March 2020].

[22] [Online]. Available: https://www.amazon.com/Newpowa-Polycrystalline-Efficiency-Module-Marine/dp/B00W80N8TA/ref=sr_1_13?dchild=1&keywords=solar+panel+20w+poly&qid=1585587606&sr=8-13. [Accessed 20 March 2020].

[23] "Types of Batteries," PRBA, [Online]. Available: https://www.prba.org/battery-safety-market-info/types-of-batteries/. [Accessed 22 March 2020].

[24] "Lithium-ion vs. lead acid batteries," energysage, [Online]. Available: https://news.energysage.com/lithium-ion-vs-lead-acid-batteries/. [Accessed 23 March 2020].

[25] "Amazon," [Online]. Available: https://www.amazon.com/dp/B07TXJNC5R/ref=sspa_dk_detail_3?psc=1&pd_rd_i=B07TXJNC5R&pd_rd_w=t6HTW&pf_rd_p=45a72588-80f7-4414-9851-786f6c16d42b&pd_rd_wg=QJyI2&pf_rd_r=EK2WYCG6BE3VM1RE57CE&pd_rd_r=8bb538cc-ad44-48e8-912b-e4313d5ec7cd&spLa=ZW5jcnlwdGVkUXVhbGl. [Accessed 24 March 2020].

[26] "How to Pick the Right Voltage Regulator(s) for Your Design," Predictable Designs, [Online]. Available: https://predictabledesigns.com/how-to-pick-the-right-voltage-regulators-for-your-design/. [Accessed 24 March 2020].

[27] S. M. Hairul Nissah Zainudin, "Comparison Study of Maximum Power Point Tracker," 19 Decemember 2010. [Online]. Available: http://www.sdaengineering.com/mepcon10/papers/278.pdf. [Accessed 24 March 2020].

[28] Rui, "7 Arduino Compatible Displays for Your Electronic Projects," 2 April 2019. [Online]. Available: https://randomnerdtutorials.com/7-arduino-compatible-displays-for-your-electronic-projects/.

[29] "Properties Table," 30 May 2019. [Online]. Available: https://www.simplify3d.com/support/materials-guide/properties-table/?filas=abs%2Cpla%2Cpetg.

[30] "HiTec Multiplex," [Online]. Available: https://hitecrcd.com/products/servos/digital/sport-giant/d840wp-32-bit-multi-purpose-waterproof-steel-gear-servo/product. [Accessed 31 March 2020].

[31] "HS-645MG High Torque, Metal Gear Premium Sport Servo," HiTec Multiplex, 31 March 2020. [Online]. Available: https://hitecrcd.com/products/servos/sport-servos/analog-sport-servos/hs-645mg/product.

[32] "Guide to IPC Standards for PCBs," 1 April 2019. [Online]. Available: https://www.mclpcb.com/ipc-standards-for-pcbs/. [Accessed 13 March 2020].

[33] W. Stallings, "Introduction To Bluetooth," 26 10 2001. [Online]. Available: https://www.informit.com/articles/article.aspx?p=23760&seqNum=3. [Accessed 30 3 2020].

[34] "Top 3 Standards for Lithium Battery Safety Testing," eurofin, [Online]. Available: https://www.metlabs.com/battery/top-3-standards-for-lithium-battery-safety-testing/. [Accessed 23 March 2020].

[35] L. Ada, "Using a Temp Sensor," 7 29 2012. [Online]. Available: https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor. [Accessed 15 2 2020].

[36] W. Hauptstrasse, "GUVA-S12SD," 2 February 2011. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/1918guva.pdf. [Accessed 2 February 2020].

[37] Gerhard-Kindler-StraBe, "BME280 Combined humidity and pressure," 1 September 2018. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf. [Accessed 2 March 2020].

[38] VISHAY INTERTECHNOLOGY, INC, "High Accuracy Ambient Light Sensor With I2C Interface," 20 September 2019. [Online]. Available: https://www.vishay.com/docs/84286/veml7700.pdf. [Accessed 3 March 2020].

[39] Guangzhou HC Information Technology Co., Ltd. , "HC-06 Product Data Sheet," 6 April 2011. [Online]. Available: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf. [Accessed 6 March 2020].

[40] Atmel Corporation, 1 February 2009. [Online]. Available: https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf. [Accessed 1 March 2020].

[41] "Digital Pins," Arduino, [Online]. Available: https://www.arduino.cc/en/Tutorial/DigitalPins.

[42] "map()," Arduino Reference, [Online]. Available: https://www.arduino.cc/reference/en/language/functions/math/map/. [Accessed 8 March 2020].

[43] Arduino Project Hub, "Getting Started with HC-05 Bluetooth Module & Arduino," April 12 2019. [Online]. Available: https://create.arduino.cc/projecthub/electropeak/getting-started-with-hc-05-bluetooth-module-arduino-e0ca81. [Accessed 27 February 2020].

[44] "TMP006 Hookup Guide," SparkFun, [Online]. Available: https://learn.sparkfun.com/tutorials/tmp006-hookup-guide/all. [Accessed 10 March 2020].

[45] J. Horsey, "DIY Arduino Wind Speed Meter Anemometer Project," Geeky Gadgets, 17 July 2019. [Online]. Available: https://www.geeky-gadgets.com/arduino-wind-speed-meter-anemometer-project-30032016/. [Accessed 5 March 2020].

[46] L. Ada, "Adafruit STEMMA Soil Sensor - I2C Capacitive Moisture Sensor," Adafruit Learning System, 1 July 2018. [Online]. Available: https://learn.adafruit.com/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor/downloads. [Accessed 8 March 2020].

[47] L. e. a. Pieterse, "Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity)," Random Nerd Tutorials, 9 July 2019. [Online]. Available: https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-humidity/. [Accessed 3 March 2020].

[48] P. Doherty, 12 July 2013. [Online]. Available: http://www.exo.net/~pauld/activities/astronomy/rotation.html. [Accessed 30 March 2020].

[49] NOAA, "National Weather Service," US Dept of Commerce, [Online]. Available: https://forecast.weather.gov/MapClick.php?CityName=Orlando&state=FL&site=MLB&lat=28.5047&lon=-81.3742#.XoDJ9YhKiUl. [Accessed 24 March 2020].

[50] "Arduino and GUVA-S12SD UV Sensor," 19 October 2016. [Online]. Available: http://arduinolearning.com/code/arduino-guva-s12sd-uv-sensor.php. [Accessed 1 March 2020].

[51] "National Weather Service," [Online]. Available: https://forecast.weather.gov/MapClick.php?CityName=Orlando&state=FL&site=MLB&lat=28.5047&lon=-81.3742#.XoZi64hKiUm. [Accessed 21 February 2020].

[52] "Guide for BME280 Sensor with Arduino (Pressure, Temperature, Humidity)," 9 July 2019. [Online]. Available: https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-humidity/. [Accessed 21 March 2020].

[53] "Adafruit STEMMA Soil Sensor - I2C Capacitive Moisture Sensor," [Online]. Available: https://learn.adafruit.com/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor/downloads. [Accessed 20 February 2020].