UNIVERSITY OF CENTRAL FLORIDA

F.I.R.E: Fire Intelligent Response Equipment

Department of Electrical Engineering and Computer Engineering
Senior Design II Project Report, Group 4, Summer 2020

Noora Dawood – Electrical Engineering

Nicholas Hainline – Electrical Engineering

Jonathan Kessluk – Computer Engineering

Arisa Kitagishi – Computer Engineering

SIEMENS
Ingenuity for life

The Siemens and STEM@SIEMENS logo are reprinted with permission from Siemens / STEM @ Siemens.
This is a project sponsored by Siemens and is not a Siemens Publication.

Table of Contents

Table of Figures

Index of Tables

1. Executive Summary

Fires cause massive environmental damage. This damage can be in the form of physical damages but also the monetary value of all the structures and items it destroys. Timely response to a fire is key as the sooner a team can be assembled to fight the fire the less damage that occurs. The F.I.R.E. system's goal is to detect fires and send alerts about the fire across large distances so that response teams can be brought together swiftly. This allows cities, states, and governments to effectively and efficiently monitor forests and large expanses of land for fires and alert a location that could be many miles away from the starting point of the fire. The system uses new wireless technology combined with machine learning and image processing techniques to determine if there is a fire in its vicinity and send an alert across the network. Creating a mesh network, the system will send alerts to all other systems and these notifications will get filtered through the system to a central location so that the alert can be handled. Using newer wireless technology LoRa and the power of machine learning, the system will accurately and efficiently monitor these large areas and assist in preventing the devastation caused by a fire.

2. Product Description

The following sections cover items relating to the product. The motivation, goals, and objectives preface everything as the project must fall back on them to complete its goal. Furthermore, this section covers the Requirements for the system and the "House of Quality" which helps product development by showing the relationship between customer requirements and design requirements.

2.1. Motivation

Over 100,000 forest fires have occurred worldwide. In the past, forest fires were considered a natural cycle and were ignored [1, 2]. However, with increasing awareness emphasizing the preservation of natural resources, as well as recent forest fires, have put forest fires at the forefront of global environmental concerns especially due to the fires Australia in 2001 and 2002 and USA in 2002 [2]. Forest fires not only increase the levels of carbon dioxide in the atmosphere, but also burn vegetation and plants that act as nature's CO2 sinks.

The increased carbon dioxide impacts air quality leading to smog and escalates the rate of global warming [3, 4]. In addition, humans and endangered animals' fatalities have been reported due to forest fires. As a result, forest fire detection and monitoring systems have sparked the interests of scientists and researchers worldwide.

In this paper a prototype forest fire detection and monitoring system is proposed as a solution. The purpose of this project is to design and build a solar powered forest fire detection and monitoring system that will serve as a preventive measure for forest fires. This device would ideally be used in areas where human activity is present such as campsites especially parts of the forest that are highly susceptible to forest fires. This device can also be used to monitor and detect forest fires to help researchers and firefighters determine incoming fires or the severity of the existing fires. Thus, the device is aimed for prevention and to facilitate extinction of forest fires.

2.2. Goals and Objectives

The main goal for this project is to design a system composed of devices whose main purpose is detecting and monitoring the environment for forest fires. The devices are portable so that it can be mounted on trees and can communicate and send data to the main hub where a forest ranger can monitor forest conditions. Moreover, the system can be calibrated to work under various forest environments.

Hardware: The hardware of the system includes a solar panel system, power regulation system, sensors for flame, smoke, and gas detection, antenna and radio frequency hardware, and processor for network and sensor data.

Software: There are two parts to the software of the system: Network and Fire Detection. The Network software manages and maintains the mesh network and allows for sending messages through the network to a “root node”. The Fire Detection software uses sensor data to determine, through image processing and machine learning, if there is a fire. The two software sub-systems communicate with each other so a message can be sent through the network.

Control: To process and control the data, the system includes a low power microcontroller and a Raspberry Pi that work together to achieve the goals of the system. The microcontroller handles the wireless communication and joining and maintaining the mesh network. The Raspberry Pi handles sensor data and determines if there is a need to send a message across the network.

Communication: A mesh network was created for the project to allow the devices to be scattered in a forest to communicate dynamically and send data to be processed at the root node.

Power Supply: The system is powered by solar panels mounted to the top of the tree or device. Since each device will draw modest current, the solar system is capable of supplying power and allowing the devices to function autonomously without significant human intervention.

2.3. Requirements and Specifications

Table 1 and Table 2 below show the requirements and constraints as determined by the project specification. Some ID numbers are maintained from old revisions, therefore some of the numbers are missing in this table. The requirements and constraints shown here are that of the final product.

Table 1: Project Requirements

ID	Category	Requirement
R1	System	The system shall detect the presence of a fire within 100m
R2	Electrical	The system shall be able to draw power from a battery or solar panel at any time
R3	Electrical	The system shall charge a battery with solar panel
R4	Electrical	The system battery shall last 36 hours without charging
R5	Electrical	The system shall communicate wirelessly to nearby nodes
R6	Software	The system shall differentiate other nodes and determine how to send data to the root node
R7	Software	The system shall read all sensors periodically and store data internally
R8	Software	The system shall process all sensor data to determine if a fire has started
R11	Software	The system shall store configuration and user defined data in non-volatile memory
R15	System	Average installation time should not exceed 30 minutes
R17	Electrical	The system shall verify environment with temperature and humidity sensors

Table 2: Project Constraints

ID	Category	Requirement
C1	Electrical	The system shall use solar power when available instead of the battery
C2	Mechanical	The system shall not be bigger than a bird's nest.
C3	Mechanical	The system shall be mounted to a tree

2.4 House of Quality

The house of quality is a product planning matrix that shows how customer requirements relate to engineering requirements [5]. The House of Quality is mostly used to identify the customer's needs and improving the development engineers' understanding of the customer's intentions. By creating an understanding between the customer and the engineers who develop the product, the product is designed correctly and efficiently while maintaining the original "market" requirements that got the project started in the first place. **Figure 1** below is the "House of Quality" for this project.

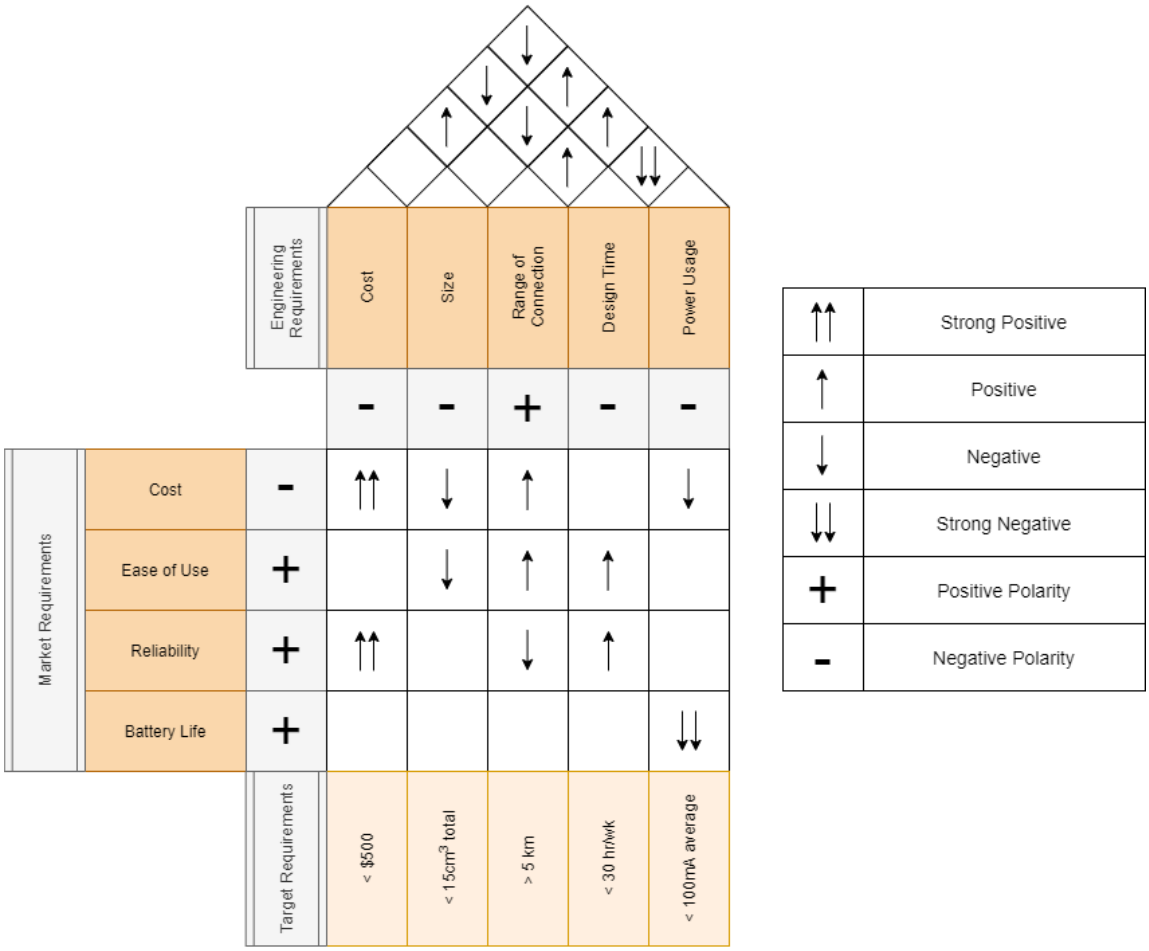


Figure 1: House of Quality

This House of Quality has four major relationships displayed. These four relationships are the cornerstone to the design. The most obvious relationship is the connection between the Cost of the product and the cost to design the product. If it costs more money to design the project, then it will cost more money for a consumer to buy it. The next relationship is directly between reliability and cost. As the reliability increases, it stands to reason that the cost will also increase. This may be due to purchasing better materials or adding in additional components or modules to improve the reliability in the design. The next relationship that matters greatly to the overall project is the inverse relationship of battery life and power usage which is directly related to design time. If the time to design this project is increased, it is likely that we will discover more efficient and better techniques to save on power consumption. Lower power consumption will increase battery life. This relationship is important because it shows that with enough time, we can make a very efficient product. Other relationships exist on the House of Quality, but they are less “powerful” correlations than the four previously mentioned. During the design and implementation of the final project special care was given to make sure that too much time was not spent on different design elements, but the more time spent definitely improved the reliability and ease of use of the product in the long run.

3. Design Constraints and Standards

Design Constraints and Standards are important to every project as they define how this project relates to the world around it. When a project follows a standard, others can define how the product behaves or is designed based on what kind of standard is followed. Furthermore, a product might be approved or denied in certain markets based on which standards it conforms to. Nonetheless, this section covers the constraints and standards that this project is designed to.

3.1. Table of Standards

The table below shows standards that could be applicable to the project and what part of the project would follow those standards. Regulations that could apply (i.e. from the FCC) will also be listed in this table.

Table 3: Table of Standards and Regulations

Standard or Regulation	Application	Where it applies
RoHS – Directive 2002/95/EC	Restriction of using hazardous materials	Entire Project
IEEE C2-2012	Safeguarding persons from hazards during installation	Entire Project – Mechanical Housing
IPC-2220 (IPC-2221)	Series of standards built around IPC-2221. Related to PCB design.	Electrical PCBs
IEEE 802.11ah	Amendment to IEEE802.11. Wi-Fi HaLow.	Research Considerations
47 CFR 18 and 47 CFR 15	Wireless communications and ISM band	Using the 900MHz bands for wireless communications
IEEE 802.15.4 and .5	WPAN and Mesh Networking standards + Chirp Spread Spectrum	Research and Design Considerations
UL 2054	Safety requirements and tests for batteries	Batteries

3.2. Other Safety Concerns

The following section discusses what kind of safety concerns in the design of the project. The project uses electricity meaning that there are some important considerations that must be made.

3.2.1. RoHS

Some materials are hazardous and harmful to the environment. To mitigate effects on the environment, this project is RoHS compliant to the best of our ability. This means that the components chosen for the project had no lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ethers, and some phthalates [6].

3.2.2. Battery Safety

Battery monitoring is an important aspect for this project as most battery types available, due to sizing constraints, are prone to self-ignition which would ultimately defeat the purpose of this project. The final revision of the project relies on a smart charging IC that manages the batteries and charges them based on battery health and voltage. The IC monitors the charge current and battery voltage and controls the charging process accordingly.

Since Li-Ion batteries are being used in this project it was important to understand their thermal limitations. Li-Ion has an issue with thermal runaway which is when a battery reaches a certain temperature and crosses a threshold that will cause the battery to rapidly rise in temperature. The battery will ultimately fail and catch fire and due to the chemical makeup of the battery the fire cannot be extinguished easily and normally burns until the fuel source, the chemicals and metals in the battery, burns out.

3.2.3. Electrical Safety

The system shall take advice from IEEE C2-2012 for Information Technology Safety and will also follow guidelines of IPC-2220 Generic Standard on Printed Board Design [7, 8].

4. Research and Background Information

The following sections discuss the research into this project idea. The project itself contains many different technologies and designs independently from each other. To make sure everything works together, research was completed to understand each part of the project before going into detail and designing the final system.

4.1. Current Fire Detection Systems

The first step was to look into current fire detection systems. These systems are based on a variety of technologies. Some of these technologies will be used by us as well but

some will be skipped over if they are not pertinent to our design goals. The following subsections discuss the products used in the industry today or that have been designed before as well as similar projects using an Arduino. Each of these projects have different costs and requirements associated. By understanding what sets these systems apart from each other, good designs can be created that meet our needs.

4.1.1. Products Used in the Industry

Current forest fire detection and monitoring systems use video cameras to recognize smoke spectrum, thermal cameras to detect heat glow, IR spectrometers, and LIDAR (detection of light and range) to detect smoke particles using reflected laser [9]. These systems are costly due to the nature of the technology. Our objective is to design a system that can accomplish its goal while driving cost down significantly through careful electronic design and component selection.

The following forest fire detection and monitoring systems exist in the market [9]:

1. AlarmEYE:
   a. Video and infrared system using black and white color frequency.
2. EYEfi SPARC:
   a. Optical sensors that includes camera, light sensors, communication, weather, power system, option for tilt zoom camera.
   b. Does not include smoke detection
3. UraFire:
   a. Smoke detection system focused on “clustering motions and a time input”
4. Forest Fire Finder:
   a. Analyzes how atmosphere absorbs light and differentiates absorption behavior
   b. Can detect smoke in a range of 15km
5. ForestWatch:
   a. Sensor camera mounted on a tower using a using a 360° pan tilt camera that scans the forest in a range of 16-20km for smoke in the daytime and flame at night.
6. FireWatch:
   a. Optical sensor system that scans the forest using a 360° camera with a central office for monitoring and data processing.
7. FireHawk:
   a. Cameras stationed strategically in the forest, the system uses GIS mapping and ForestWatch software to calculate the shortest distance to the fire.

4.1.2. Similar Project

The *Arduino fire alarm system using temperature and smoke sensor with Android connectivity* is a product that exits in the market for $5,900 USD and serves a similar purpose to the final product aimed to design [10].
A major drawback of this kit is the high market value price despite the product using

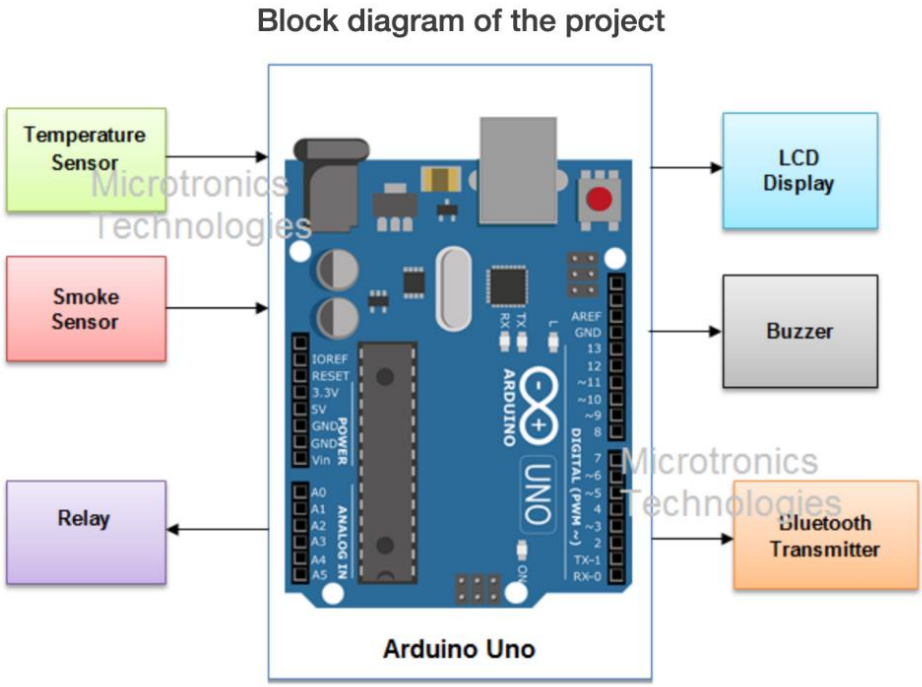


Figure 2: Arduino Uno being used in a similar project [10]

straightforward components. This price can be attributed to the fire-proof enclosure, which typically raises the cost of the system. Moreover, the product uses Bluetooth technology to communicate an alert through a mobile app. Bluetooth technology can range from 30m to 100m, which could function in an indoor environment but is not ideal for an outdoor environment that is intended [11]. Moreover, it is not clear if this product will communicate with other fire systems around it, such as in a mesh network.

However, this product contains many of the features intended to use for this project and the strategic placement of parts will be useful when designing the printed circuit board for this project. The temperature sensor, smoke sensor, and microcontroller are components that would be implemented in this project. Thus, our achieved a similar objective to the Arduino system; however, most importantly the cost is significantly lower with wider range and similar fire detection technologies.

4.2. Background Research

After looking at systems that already exist to detect fires, an investigation on other kinds of technologies that the project will use. Without an understanding of these individual parts, the system will not function properly. In this section, a narrower view is taken such

that individual components, sensors, and protocols are examined for their efficacy in the project.

4.2.1. Serial Communication Protocols

Serial communication is dependent on the type of microcontroller used and the communication protocol of the chosen sensors. Based on the research, the likely protocols to be used for this project will be SPI or I2C.

SPI or Serial Peripheral Interface requires a 4-wire connection: a clock signal (SCLK), a slave select signal (SSn), Master Out Slave In (MOSI), Master In Slave Out (MISO) [12]. SPI uses a protocol where a single device sends the communication to the slave devices, thus it uses the single-master communication protocol [12]. In order for communication to occur, the master and slave must use SCLK frequency, CPOL, and CPHA [12]. In the event when multiple slaves exist, the master will reconfigure itself each time to initiate the communication with each slave [12]. SPI does not have a maximum data rate, nor does it use a specific addressing structure. In addition, SPI does not have a system to acknowledge that the device received data or options to control the flow of data [12]. Therefore, if SPI is used in command type applications, an additional structure would need to be incorporated.

The physical interface of SPI is flexible in the sense that many variants currently use a continuous clock signal and random lengths compared to past types that were non-continuous clocks and used a single byte scheme.

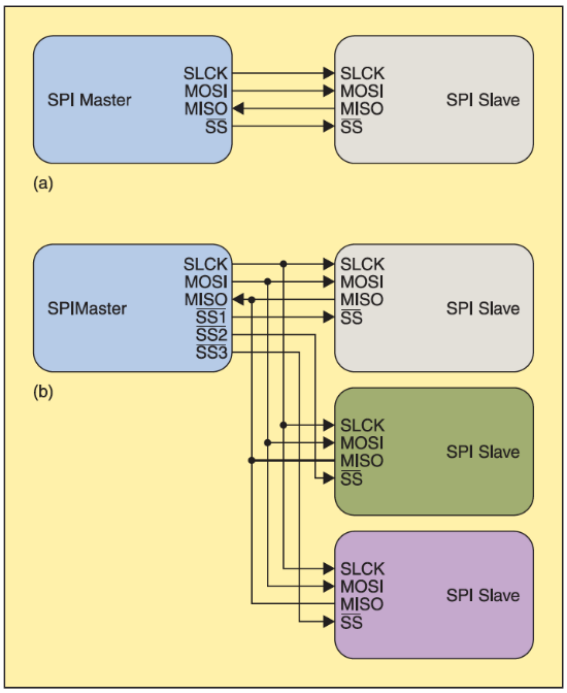


Figure 3: SPI Topology [12]

I2C or Inter-integrated circuit is known for requiring a 2-wire connection between the peripherals and the microcontroller [12]. The two signals are called serial data (SDA) and serial clock (SCL) [12]. I2C allows multiple slaves and masters to be connected and communicate (bi-directionally) between the two lines using a protocol that includes 7-bit slave addresses and data divided into 8-bit bytes [12]. The bus master is the IC that initiates the data transfer, while the remaining IC are considered bus slaves [12]. The data rate should be between 100kb/s, 400kb/s and 3.4 Mb/s for standard mode, fast mode, and high-speed mode, respectively [12]. There some variants of I2C that include a low speed mode at 1kb/s and fast mode + at 1Mb/s [12].

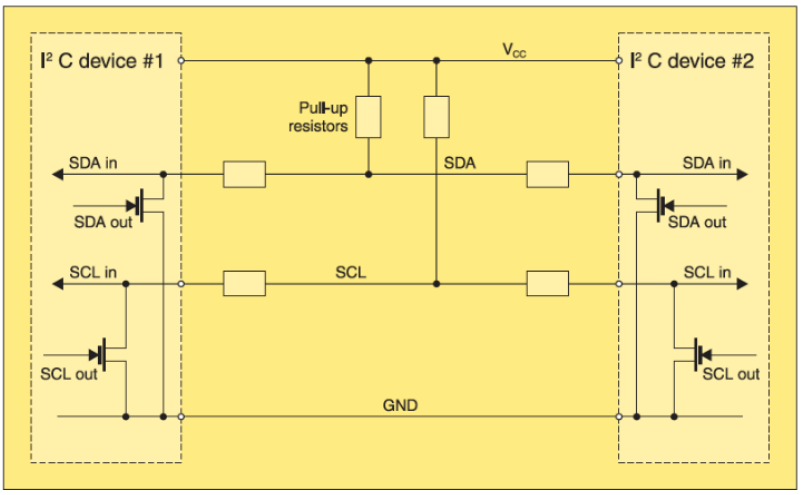


Figure 4: I2C Topology

The physical interface of I2C is compose d of SCL and SDA lines as open drain I/Os with pull-up resistors; while grounded it is a logic zero and while released is a logic one [12]. Due to the physical structure of I2C, communication can occur without conflict even if multiple two devices are continuously sending information on the SDA and SCL lines; there is no electrical interruption due to the open-drain and pull-up setup. This is illustrated in Figure 5 [12].

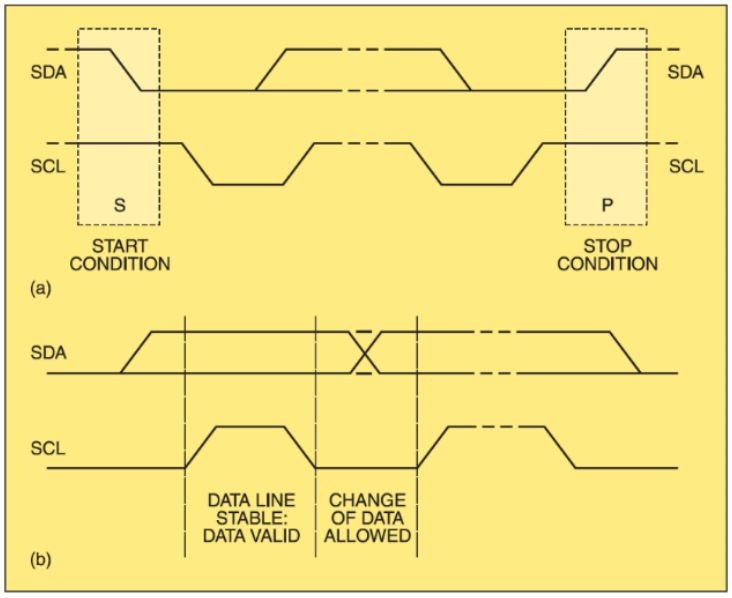


Figure 5: Timing Diagram of I2C [12]

I2C has several advantages over SPI. Firstly, since I2C only uses 2-line connections, this allows easier implementation since less pins are required. Moreover, I2C allows for smooth communication with is advance feature of resolving multi-master communication conflicts on a simple physical structure [12]. I2C's setbacks in comparison with SPI is with data rate; SPI is a full-duplex which means simultaneous communication is possible. Moreover, SPI does not define a speed limit for transmitting data [12].

After examining both protocols, I2C is the ideal communication protocol between the microcontroller and the sensors; however, SPI is not completely ruled out. The advantages I2C provides helps achieve the purpose of the project in a straightforward manner.

4.2.2. Sensors

4.2.2.1. Gas Sensors

When reviewing gas sensor types, the important parameters to consider are sensitivity and selectivity. Additional parameters to consider are response time, stability, reversibility, energy consumption, fabrication cost, and adsorptive capacity according to IEEE fellow researchers investigating fire sensing technologies [13]. Gas sensors detect gases by observing for variation in the sensor output, which typically is an analog value; however, some gas sensors send a digital signal out.

Table 4: Gas Measurements in the Atmosphere During a Fire [14]

Gas	IST	Alphasense	GfG
NH_3	√ 10 ppm	√ 100 ppm	√ 200 ppm
CO	√ 300 ppm	√ 500 ppm	√ 300 ppm
H_2	√ 2000 ppm	√ 2000 ppm	√ 2000 ppm
HCl	√ 30 ppm	√ 100 ppm	√ 30 ppm
HCN	√ 30 ppm	√ 100 ppm	√ 50 ppm
HF	√ 10 ppm		√ 10 ppm
HBr			√ 30 ppm
H_2S	√ 30 ppm	√ 100 ppm	√ 100 ppm
NO	√ 100 ppm	√ 100 ppm	√ 100 ppm
NO_2	√ 50 ppm	√ 20 ppm	√ 30 ppm
SO_2	√ 100 ppm	√ 20 ppm	√ 10 ppm
O_2			25%

Sensors vary by the material used; existing materials in the market include semiconductor, catalytic bead, photoionization, infrared, and electrochemical. Additional gas sensor types include optical, acoustic, gas chromatograph, and calorimetric [13].

In the event of a fire, the air quality changes; the severity depends on the severity of the fire and the environmental conditions. Forest fires tend to release high levels of N2, O_2, CO, CO_2, H_2 gasses [13]. Changes in oxygen levels can provide indication of the type of fire. A low change in concentration suggests a smoldering fire while large changes suggest liquid fuel fires that rapidly burning fires [13].

Gas sensors made with semiconductor metal oxide are an ideal choice of materials however they come with disadvantages namely with stability issues that lead to false alarms [13]. However, despite this issue, zeolites have been used instead of metal oxides to compensate for this issue [13]. Moreover, gas sensors that use polymers have shown to enhance sensitivity [13].

Based on spectroscopy laws, gas sensors that use optical methods are more stable, sensitive, possess better selectivity, and have a low response time [13]. However, optical gas sensors come with the disadvantage of higher costs [13].

A novel method of gas detection uses acoustic waves by detecting the change in velocity of the wave due to adjusting a parameter of the sensor's material, for example the mass [13]. A laser beam is shined through the gas. The gas molecules absorb the beam and releases the beam's energy resulting in an acoustic wave which is detected using an acoustic sensor. The magnitude of the wave is used to identify the concentration of the gas in the atmosphere. The figure below provides a depiction of how this is achieved.

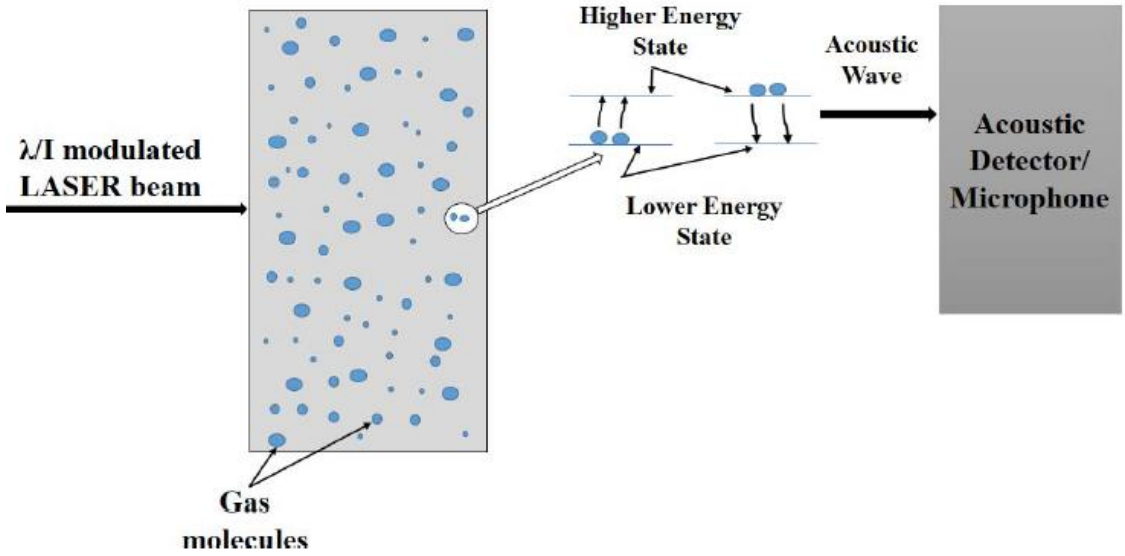


Figure 6: Acoustic gas detection method [13]

Other methods of gas detection use a combination of sensors to detect temperature and humidity and an algorithm to detect gases such as CO and CO2 [13]. These gas sensors use metal oxide or n-LTPS MOS Schottky diode on a glass substrate [13]. SnO2 provides the highest quality in terms of sensitivity ratio; this was used for gas sensor to detect gasses emitted during fires by detecting the smells from cotton and the printed circuit board when it is heated at 200 degree Celsius [13]. This is achieved by measuring the change in resistance of the parts due to gas emission.

4.2.2.2.Smoke Sensors

Understanding smoke characteristics and causes helps understand how smoke sensors function in order to choose an appropriate smoke sensor for forest fire applications. Smoke is produced when a fire is burning and materials are combusted; it is composed of airborne solid, liquid particulates, and gases, which deems it an unwanted element in the atmosphere since it reduces the air quality in the environment.

Smoke detection uses two techniques to detect its presence: non-visual and visual [13]. In a non-visual method, the detection technique looks smoke combustion conditions such as pyrolysis, smoldering, and flaming; these conditions are contingent on the type of fire and the environmental surrounding [13].

Smoke detection methods that use the photoelectric principle are primarily used for smoldering conditions and is effective in doing so; response times are quick [13]. In this method, the ionization smoke sensor measures smoke relative to the ionization levels in the air [13]. A potential difference is applied through a chamber and the output current is measured as a result [13]. Moreover, photoelectric method dictates that the concentration of smoke in the air will proportionally increase the light scattering capacity [13]. Thus, this method measures the variation in light scattered using optical science and technology to detect the smoke levels in each area. It is also common to combine this method with gas sensing technology for better results.

Other smoke detectors use alpha particles to the gate of a MOSFET which induces a positive charge [13]. When the smoke concentrations are high, smoke particles decrease the number of alpha particles in the gate terminal which then reduces the current [13]. Other photoelectrical methods investigated the range of transmission for wood smoke using a white polychromatic LED, an optical fiber, pyrex glass window, and photodiodes [13]. This could be implemented in a forest environment. The figure below provides a visual of how photoelectrical smoke sensors works.

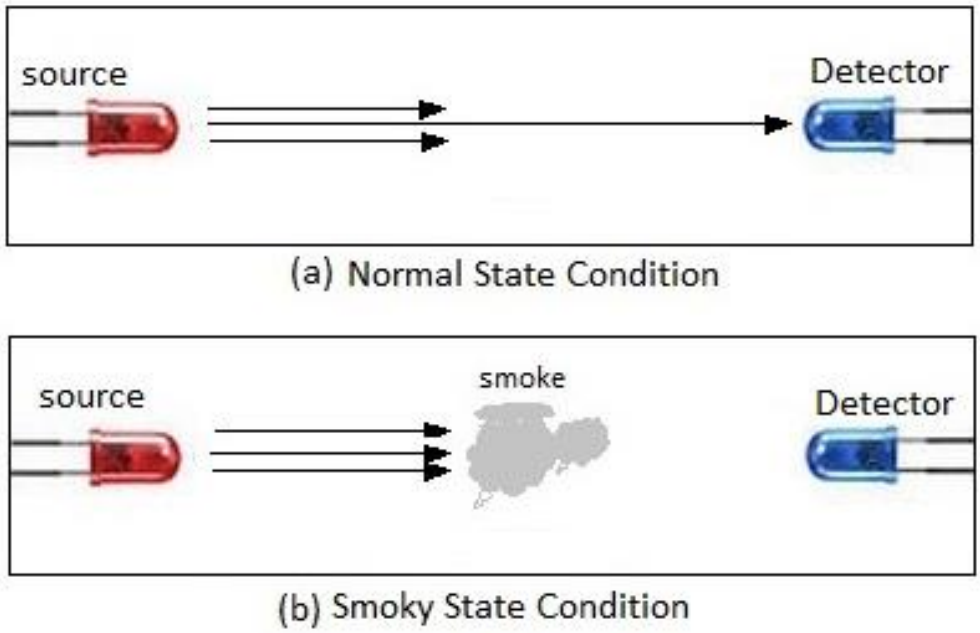


Figure 7: Visual representation of photoelectric smoke detection [13]

Visual techniques mostly use cameras which can detect both flame and smoke [13]. The nature of smoke is that it exists at the beginning of the fire which is crucial when designing fire-detection strategies. Smoke detection uses color space, specifically RGB or YUV. With RGB, pixel rules must be used; however, with YUV, the rules are dictated by looking at chrominance and luminance values [13]. To overcome false alarms, luminance mapping is used paired with support vector machines (SVM) algorithm, and Bayesian network algorithm. Other techniques to detect smoke use Adaboost with staircase searching [13].

Yet, detecting smoke at the early stage can be difficult when comparing it flame detection; it is very common for smoke and flame characteristics to be used when creating algorithms. However, smoke direction can be detected using cameras and various algorithms.

4.2.2.3.Flame Sensors

In order to understand flame detection to choose a suitable sensor, it is important to understand the nature and characteristics of a flame. Flame is a visible exothermic reaction that occurs in a fire due to fuel and oxidants interacting, thus flames emit radiation and chromatic properties. Flame temperature is dependent on the material that is burning.

There are two methods of flame detection: non-visual and visual flame techniques [13]. Non-visual flame sensors use ultra-violet, visible, and infrared rays [13]. This is because flames emit a radiation whose intensity is determined by the flame temperature and the type of fuel burning [13]. An ultra-violet sensor is used to measure the brightness since UV sensors are not impacted by interreferences from other radiations such as infrared [13]. Additionally, infrared and visible light sensors are used to measure flame. However, IR and visible light sensors are more effective than ultra-violet sensors [13]. UV sensors tend give out more false positive alerts due UV sensors emitting sparks of UV spectra that essential interferes with the signal [13]. To overcome this effect, a near infrared photodetector (NIR) can be used for flame detection. NIRs are made of Pb semiconductor using Colloidal Quantum Dots (CQD) technique [13].

Visual techniques for detecting flame can be difficult because standard heat, smoke flame, and gas sensors can delay in receiving a response [13]. This is because the particles must reach the sensors in order for the sensor to trigger a response signal [13]. Moreover, the range of detection tends to have a small radius. As a result, this issue is typically resolved by installing many sensors to cover a large area [13]. Moreover, the nature of fires come with various characteristics such as shape, size, color, location, growth, degree of burning, and dynamic texture and typical sensors are not capable of measuring each of these characteristics and their parameters accurately [13]. Thus, flame sensors that depend on these techniques give false alarms whose validity can only be evaluated by an experienced individual.

A device to solve this issue is using a camera that can capture images of fire and analyze them accordingly to establish fire detection. Such cameras tend to be very high cost; thus, it is more common to see surveillance cameras being used instead. IR cameras have been used for flame detection by using the Markov model to detect flame flicker [13]. The figure below is a flow chart that explains how this works.

Once a camera records data and provides it in the RAW, RGB, YUV, JPEG formats, algorithms can be used to examine the images and deduce if the image frame has the visual characteristics of a fire or not. There are two main methods of designing the algorithm. The first approach analyzes characteristics such as color, shape, flickering

frequency, and dynamic texture of the fire [13]. This requires the use of color spaces; YCbCr color space showed to be the most effective for flame detection [13]. Other color spaces that can also been used are RGB, CIE L∗a∗b∗, YUV, or HIS [13].

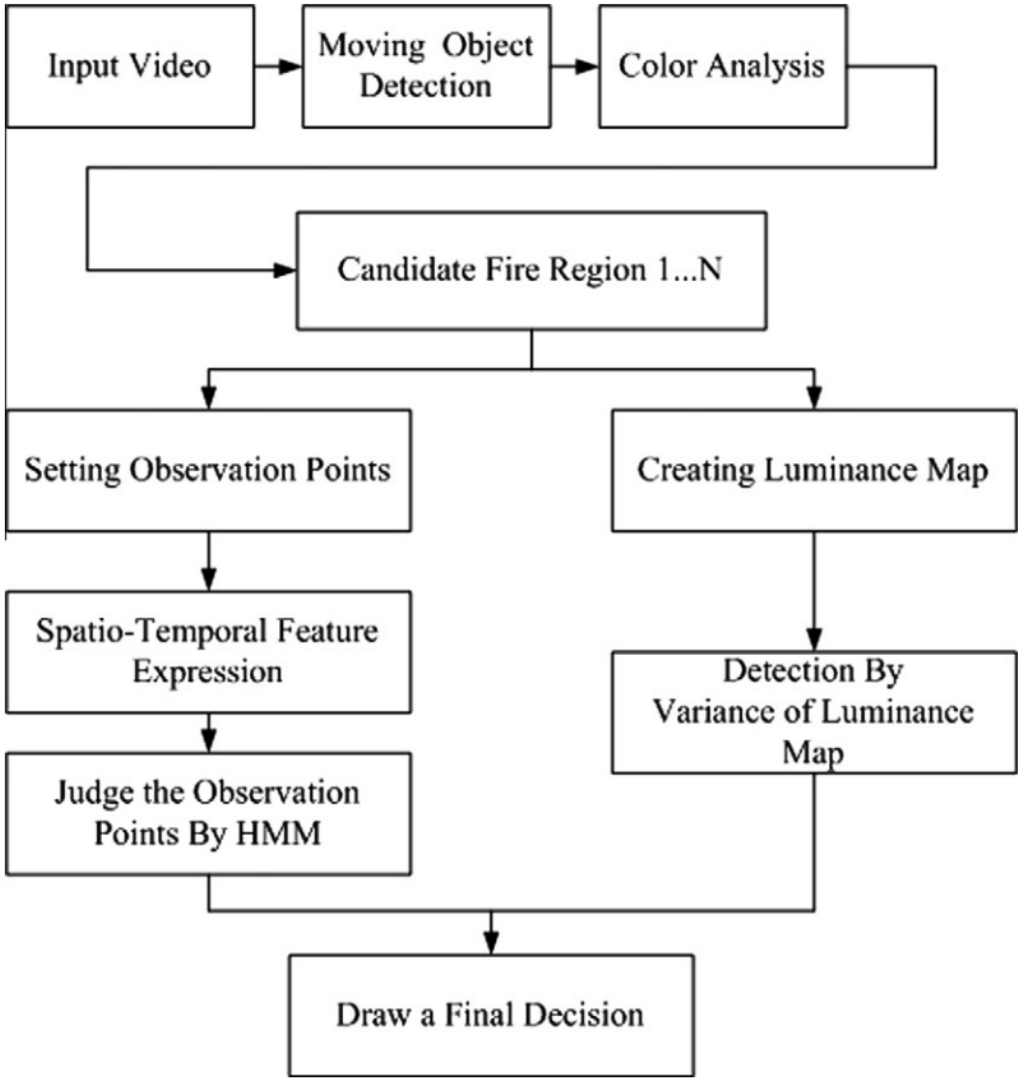


Figure 8: Hidden Markov model used to detect flame flicker [15]

Color information is not enough to provide accurate results [13]. Movement of fire has also been examined for fire detection techniques by using background subtraction method, temporal differencing, and optical flow analysis [13]. The Markov model can be used to detect flame movement for object that have flame-like colors as well as flame boundaries using temporal wavelet analysis [13]. Moreover, a moving camera can be used to observe moving flame pixels without using background subtraction [13]. This can be paired with detecting color, temporal, and spatial information in each spatiotemporal area. However, this method can slow the fire-detecting process since the range is weak.

Another method utilized the Wald-Wolfwitz algorithm for flame detection looking a parameter such as color and predictive motion movement [13]. The reliability of the results was increased using a “convolution operation” [13].

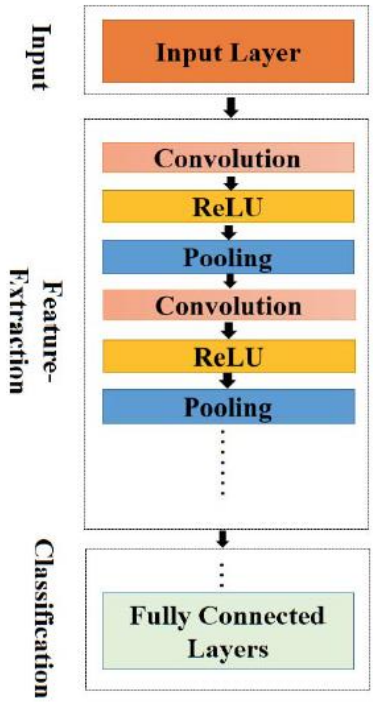
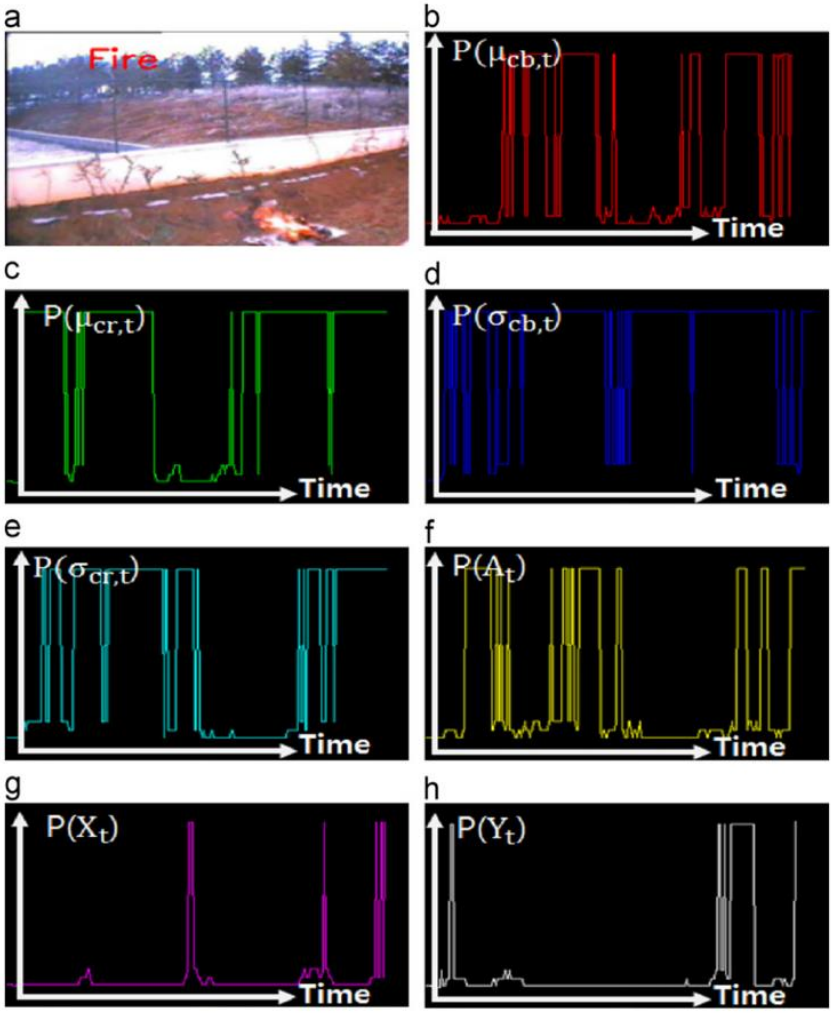


Figure 9: Convolution neural networks approach layers [13]

The second approach of designing the fire detection algorithm utilizes a learning-based approach [13]. In this method, the system is provided a dataset of fire and non-fire images and is “trained” to make an appropriate judgement by analyzing for specific fire features. Convolution neural networks approach is a common approach that achieves this, as well as You Only Look Once (YOLO), and is discussed later in the paper. The figure below provides a visual of the layers involved



.

Figure 10: Example of the Wald-Wolfwitz randomness test being used for flame detection

4.2.3. Fire Resistant Enclosure Materials

There are a multitude of fire resistive materials to choose from but what was need for this project is something that is light and offers the most fire resistance possible while not being excessively expensive. For the purpose of this project the enclosure that will be used for the prototype will not be fire resistant. This was done to minimize the cost and manufacturing processes for senior design two. If this project was to be mass produced, then a fire protective encloser would be utilized to protect the devices in case the fire it has warned about has climbed up to wherever the device is located.

A few choice materials have been selected for their fire resistive properties and their ease of implementation into a manufacturing process. The first material is Kevlar; Kevlar is a synthetic material developed by DuPont and is extremely shock resistant and fire resistant [16]. It is not very abrasion resistant but that will not be an issue as the Kevlar would be manufactured into a composite material consisting of a resin and the Kevlar woven cloth.

The issue with making an enclosure this way is the fire resistance is now going to be limited to the resin is used to cast the composite into shape using a mold. The Kevlar itself has some draw backs, it is very expensive and being a synthetic fiber, it can cause some medical issues if the individual fibers are inhaled [16].

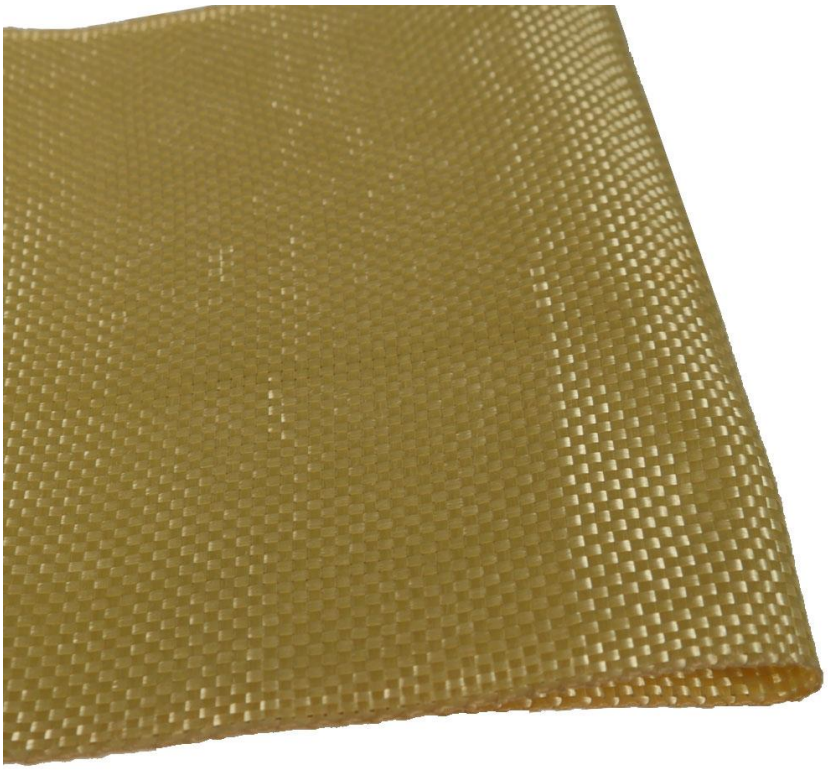


Figure 11: Synthetic Fiber Kevlar [17]

Another choice material is carbon fiber weave. Carbon itself is very fire resistive and in its pure form is used for nearly all castings for materials that need to be heated to extremely high temperatures, temperatures way hotter than a normal wood fire could ever reach. It has the same drawbacks as Kevlar when it comes to cost and handling of the raw material. The carbon fiber weave would also need to be made into a composite with a high temperature resin which would limit the fire resistance to according to the resin is limited to.

The easiest material for manufacturing would be plastic. Most plastics are not fire resistant at all. They have many failure modes from melting to ignition. For this project a plastic compound that does neither is preferred. Luckily, there are plastics that only burn up and off gas when they do so, but they do not ignite or melt. These plastics could have additives put in them to increase their fire resistance; an example is any compound that is a brominated flame retardant (BRF). These compounds burn up in the fire creating a sort of sublimating coating around the plastic which the fire must get through first to burn the plastic.

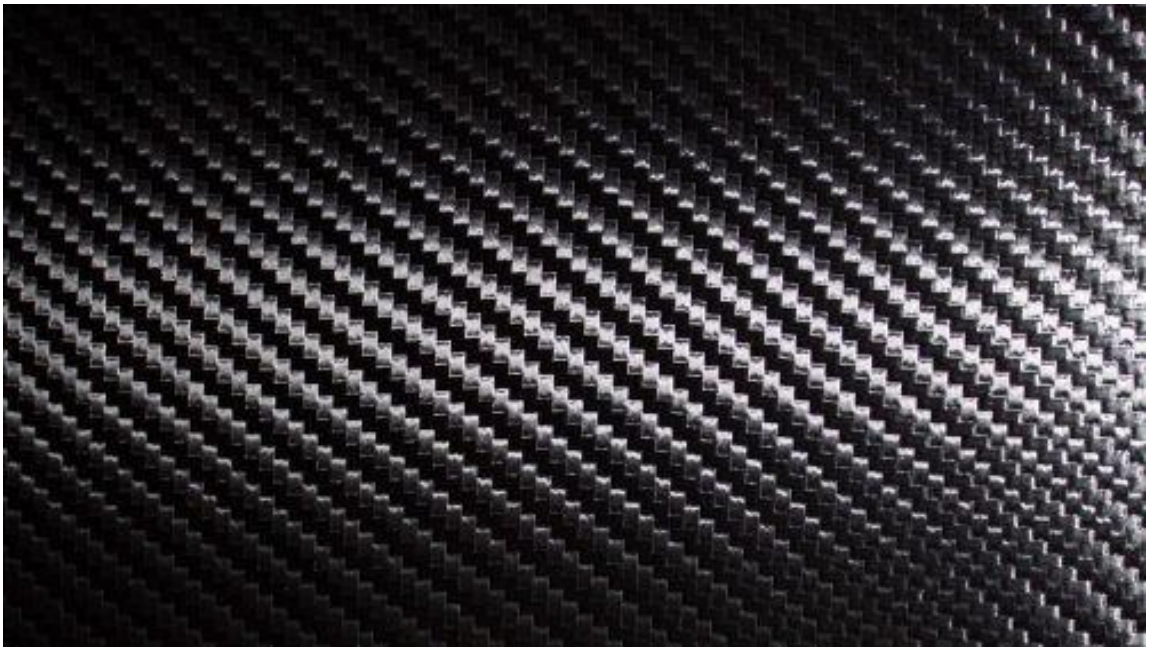


Figure 12: Carbon Fiber Weave [18]

4.2.4. Battery Charging and Battery Chemistries

The battery for the project was capable of lasting throughout the night and when the solar radiation is low on average for the winter months it could handle not being at full capacity during the day. There are many battery chemistries to choose from with a few types of batteries not being viable at all for the system. Lead acid and absorbent glass mat car or RV style batteries could not be used due to sizing and weight. Lighter smaller batteries were the only batteries available to be used so a NiCad or Li-Ion battery style was used. For this project, the best option was to go with a battery within budget that was the most power dense and the chemical makeup of said battery allows for the most charge cycles.

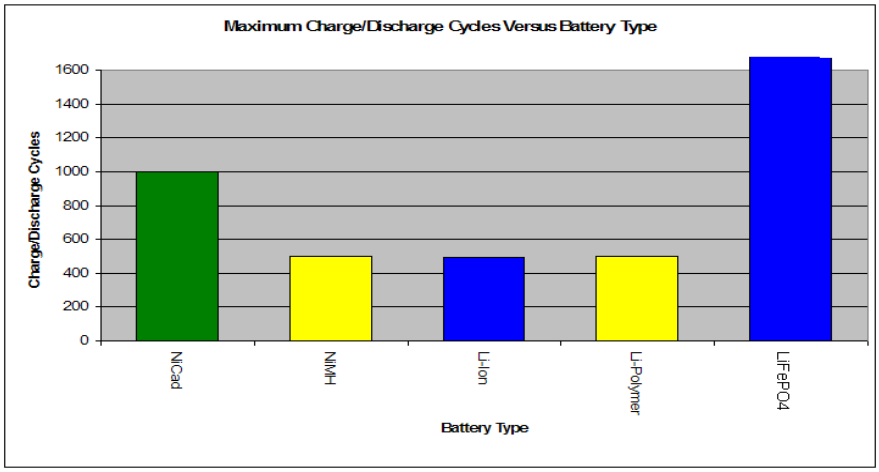


Figure 13: Maximum Charge/Discharge Cycles Versus Battery Type [19]

Researching different types of batteries lead to an issue arising between them. The max depth of discharge had to be accounted for along with how many charge cycles the batteries could handle. NiCad, or nickel cadmium batteries, were a cheaper option for the project but the weight and size made them a bad option. Nickel-metal hydride batteries come at a high price and lithium ion batteries, along with lithium polymer batteries, have the same level of charge cycles so it made sense to go with which ever one was the cheaper option. The best option at first seemed to be lithium iron phosphate, or LiFeP4, but this one is the most expensive out of all the options but did allow for the most charge cycles out of them all. For a final product it may be a better option to go with this battery but for the sake of cost, weight, and size Li-Ion batteries were the best option for the prototyping of this system.

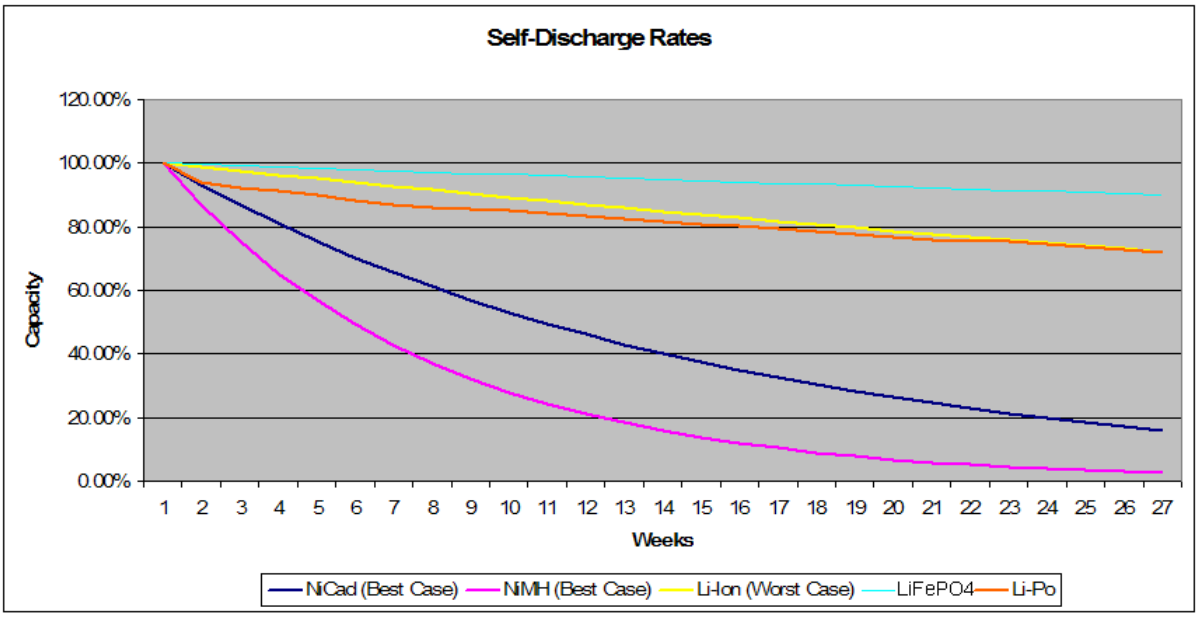


Figure 14: Self-Discharge Rates of Batteries [19]

To further cause issues in choosing a battery for the project the self-discharge rate was a factor to deal with. Self-discharge is when a battery will slowly lose charge over time when not in use. Since the solar panels were going to charge the batteries every day the discharge rate wasn't a massive impact but the battery that was chosen needed to be able to handle a day of not being charged and remain in stand by for when the weather blocks out the sun for a while like in the case of a thunderstorm or hurricane. The only battery that could be ruled out this way was NiMH as it had the highest self-discharge rate of all the batteries being looked at.

4.2.4.1. Effect of Temperature on Batteries

For this project, the batteries are going to experience high ambient temperature in the summertime and most likely very low wintertime temperature due to being placed in higher locations out in the open. These conditions change how a batteries chemistry works and will change the overall life and performance of the battery. Wintertime temperatures shorten the charge life of a battery by slowing down the chemical reaction

happening in the cell when power is being drawn from it. When in standby and not being used cold temperatures will increase the self-discharge rate of the cells which is only worsened by the fact that wintertime conditions lessen the output of a solar array because the solar radiation isn't as condensed as in the summertime. This will cause the overall max amp to draw to be less as well and could cause total system failure due to over drawing the battery.

The figure bellow is a graph that highlights what happens to the battery maximum charge storage capacity if the temperature is increased like in summertime conditions. As shown the max storage of the battery is not affected until the natural battery charge cycle lifespan starts to end. The high temperatures only increase the damage done by having a battery go through many charge cycles with the peak temp of 55C having the most affect it can be postulated that further increase would cause even more damage but seeing how 55C is 131F it is unlikely ambient temperatures will exceed this unless the device is currently engulfed in a fire.

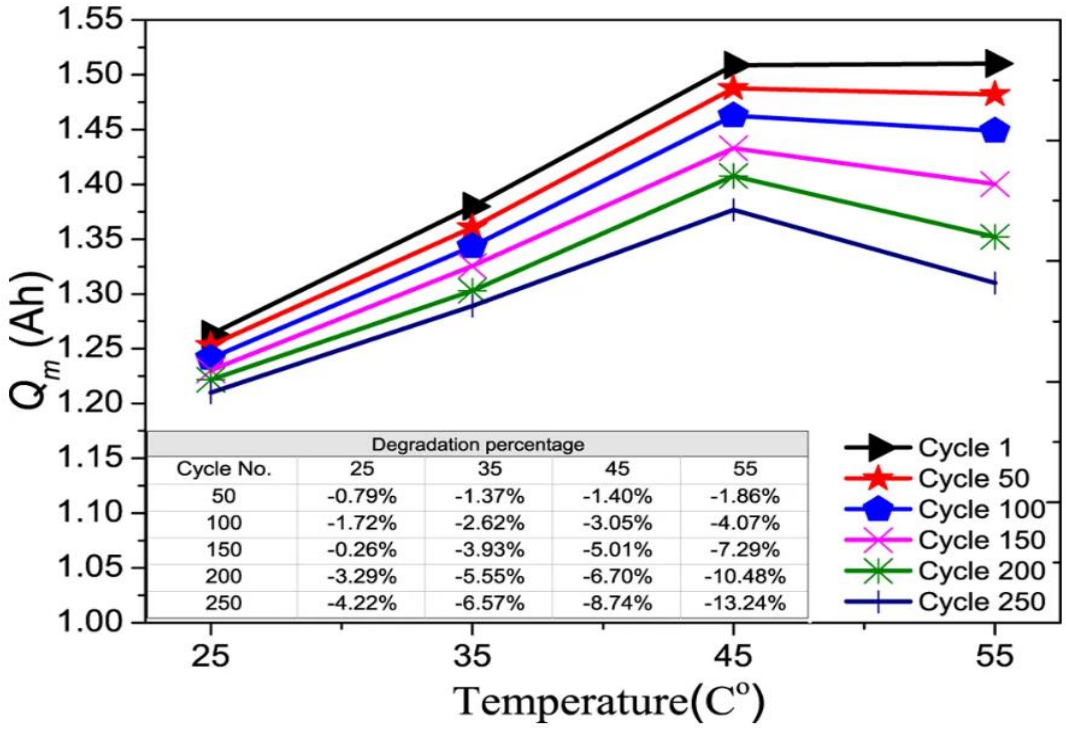
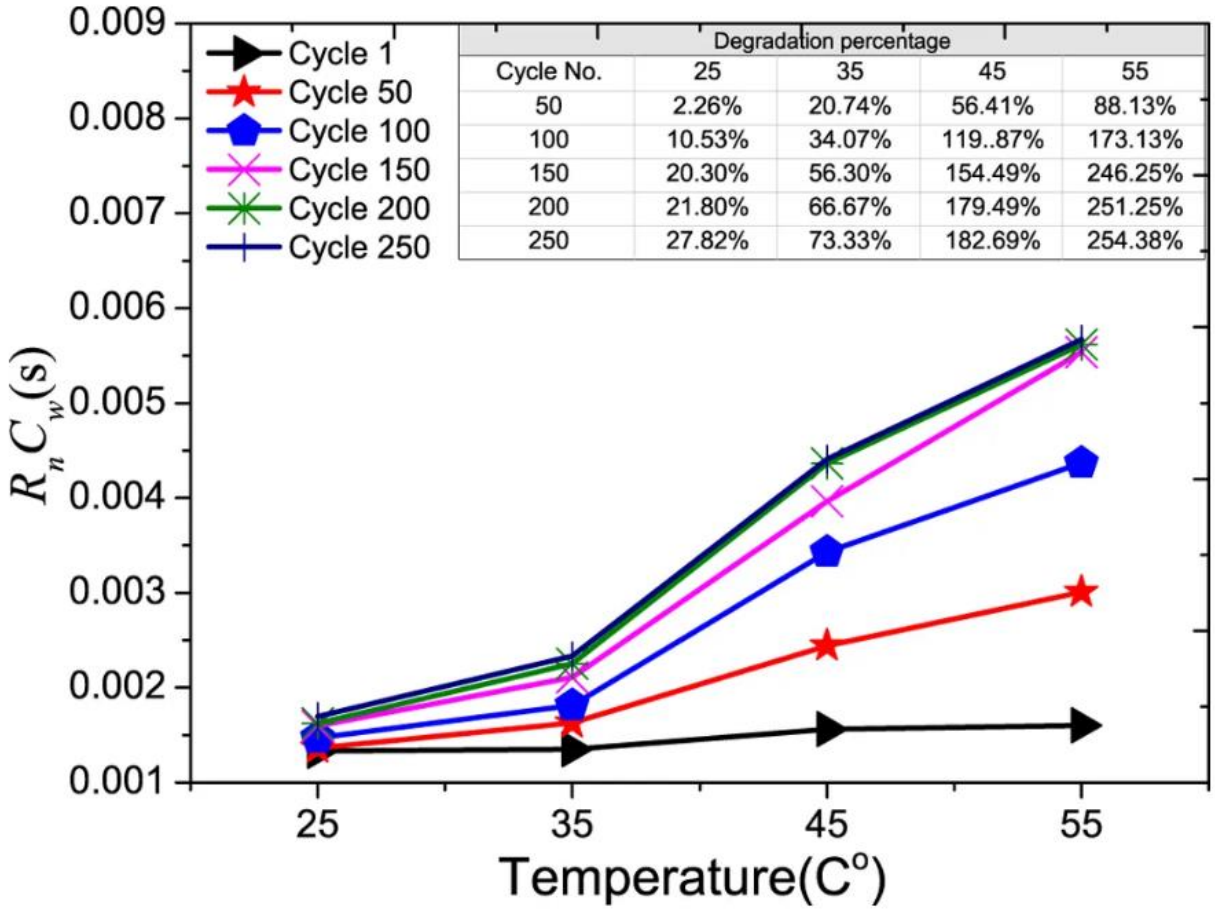


Figure 15: Temperature Vs Charge [20]

The other figure, shown below, is a graph that shows how an increase in temperature on a battery will lower the max amp output of a battery. R_nC_w is the current flowing across a resistor and capacitor in parallel and demonstrates how the battery has an exponential threshold at 45C and any increase beyond this will drastically decrease the max amp draw of a battery. This could cause the same issue as the wintertime conditions in where a total system failure is cause due to pulling to many amps from the batteries.



Degradation percentage				
Cycle No.	25	35	45	55
50	2.26%	20.74%	56.41%	88.13%
100	10.53%	34.07%	119..87%	173.13%
150	20.30%	56.30%	154.49%	246.25%
200	21.80%	66.67%	179.49%	251.25%
250	27.82%	73.33%	182.69%	254.38%

Figure 16: Temperature Vs Time (cycles) [20]

All this means the system needed to be designed to be able to handle the drastic temperature differences seasonal changes brings which was accounted for by doubling the power supply system. The number of battery cells was doubled along with the solar array. This safety factor of two will provide a hefty cushion of protection any temperature change might cause on the system by having the system draw minimum power in the most optimal conditions.

4.2.5. Li-Ion for System Power

The final decision to use Li-Ion batteries for this system was made from multiple different decisions. The major decision that this battery was chosen for this project was cost and availability. This means that size and configuration for the battery was found and sourced for the constraints in the project. Many of the students working on this system had the standard 18650 sized li-ion cell battery to use for any testing of the system so it made the most sense to go with this battery chemistry for the system.

A few smaller reasons is the way li-ions operate such as how the power output doesn't drop as much when the battery is depleted. The chemical make-up of the battery also

allows for a minimum for 70 full charge cycles. That means the batteries can be fully discharged and then recharged to full 70 times before the battery health starts to deteriorate. This does not prevent the battery from operating anymore it only causes the battery to discharge faster than when it was new.

4.2.6. Power Supply Topology

The power supply for this system was a solar array that is 12 volts nominally and is hooked up to a LT3652 solar lithium charging IC that took the 12 volts and regulated it to the most optimal voltage, 16.6 volts, with a mix current draw of 2 amps. This was then hooked up to lithium batteries in series. The battery and solar panels are then hooked to a buck converter that maintained 3.3-volt and 5-volt rails for the Pi and sensors for the system. This power system was able to utilize the solar cells and battery at the same time as to not put undue stress on the batteries.

4.2.3.1 Linear vs Switching Regulators

There are two common types of power converters: Linear and Switching. Linear regulators are the simplest form of regulator as they directly convert power in to power out. That is, there is no complex operation internal to the regulator. This simple conversion, however, comes at a price. Linear regulators dissipate a lot of heat when used and are generally inefficient. As a result, a linear regulator will require a heat sink if a lot of power is expected to be converted to heat. This will add weight and cost to the design. Luckily, a linear regulator is generally cheaper and has less components to support it than a switching regulator. A downside to using linear regulators is that they must always step-down voltage. There is not a way to step up the voltage through a linear regulator. Switching regulators provide many different topologies that can, in some cases, raise the voltage.

It is even possible to design a switching regulator that can lower or raise the input voltage if it is unstable and is sometimes higher or lower than the desired voltage. This does not mean that linear regulators do not have their use. Linear regulators are great when there is a decent amount of power coming in and lower power draw on the other side. An advantage to using them is when there is a small difference in voltage going in and voltage coming out. If the desired voltage is just slightly lower than the input voltage, then the efficiency can be greater than 97%, but only in this case. Usually, it is lower. Considering our design with batteries: two batteries in series will generate around 8 volts. If we use a linear regulator to step down to 5 volts or 3.3 volts, there is a significant (greater than a volt) decrease in voltage. I can be expected, in this case, that the linear regulator will be much less efficient than a switching regulator. Since our design is purely powered from a solar panel and a battery, it important to ensure must that the system is efficiently transferring power between different parts of the circuit and not wasting any power in heat dissipation.

Switching regulators, on the other hand, are the more likely solution that will be implemented in the final design. Compared to linear regulators, they are more expensive

and require more support circuitry causing them to be a bit more complex than a linear regulator. The benefit is in efficiency. With a circuit designed properly, the power coming in can be efficiently converted to the proper power going out. Sometimes even greater than 99%! Since the system is charging the battery and running the circuit off of the solar panel, the system will need to efficiently step down the voltage from the panel to the battery charge voltage, and then from the battery voltage to the circuit voltage. This means the system will need 2 to 3 switching regulators in our final design. The high efficiency implies that there is very little heat dissipation. This is good, as our ambient temperature is going to be higher (or lower in some cases!) than room temperature. The device is designed to operate outside. In turn, it not ideal to have the device to have too much effect on the heat around it such that it does not exceed specific component heating constraints. Since the switching regulator has more components to support it, it will take up more board space. This is a price that must be paid for the efficiency boost. In the end, the board space is not critical if designed appropriately. Since our project does not have any externally imposed sizing constraints, it was possible to move forward with switching regulators.

4.2.3.2 Buck and Boost Converter

While discussing switching regulators, it is good to have a general idea on how the major topologies of different switching regulator designs operate. In the design, the system does not expect to be raising voltages. Therefore, we need to step-down a voltage. This kind of converter is known as a Buck Converter. They are configured only to step-down DC voltages. Buck converters store energy in a passive component, usually an inductor, and uses that stored energy to output a specific value. To store the energy, a pulse width modulator can be used to charge and discharge the passive component as necessary. The duty cycle of this pulse determines the voltage that is output since the passive component must be discharging to provide current. While charging, the passive component is usually supported by a capacitor on the output end of the regulator. The passive component is in series with the load, causing a voltage drop due to the impedance of the device and the time that can charge the passive device. Even though the voltage is lower, the charging/discharging of the device still keeps the average power equivalent (or nearly so) while in operation.

In juxtaposition to a Buck Converter, a Boost Converter is designed to increase voltages. This step-up behavior works in the same way as the Buck Converter: A transistor works as a switch and at the right switching frequency it charges and discharges a passive component, usually an inductor. Since current cannot change instantaneously across an inductor, when the switch is open, the energy stored in the inductor elevates the voltage level above the input of the storage component to keep power consistent on each leg of the power network. This action compensates for the lower voltage on the other side at the cost of lower currents on the high side. The load now sees a higher voltage than the input source has.

In both designs, heavy filtering may be necessary for sensitive components to avoid issues with the switching action. This “On-Off” methodology introduces noise into the

power system which can be detrimental to digital logic devices (like microcontrollers, processors, or DSP devices). In many cases, a switching regulator will feature strong capacitors in the input and output stage to help compensate for the noise and filter it out.

4.2.7. Solar Array Design

Assuming a 12v system that draws an average of 100mA, it was determined that the needs of the system to be almost 29Wh per day. For five hours of maximum power output with an MPPT 5.76W is needed, so a 500mA MPPT would be needed. The MPPT would have to be a custom made one as the ones available for purchase are over specifications and would not be able to handle the low amount of power flowing through them efficiently.

To make sure the system could handle any solar radiation issues that could arise for a few reasons such as weather or the panels getting dirty the entire system had a safety factor of 15% was applied. The needed battery size was doubled to account for unforeseen issues. Two 1.5 watt panels were chosen and these two panels were wired in parallel to increase the amperage and to not exceed the 40 volt max of the LT3652 IC.

4.2.7.1. Azimuth and Elevation/Tilt Angle

To properly arrange a solar array to absorb the most solar radiation as possible it needed to have the proper azimuth and inclination angle for the location it was being used at. Azimuth is the direction the panels are pointed, measured in degrees with 0 degrees being south and north being 180 degrees, and inclination is how far a panel is tilted up. The change is performance of the system if not properly set up is drastic and the location of the solar panels changes both these parameters which can make it difficult to calculate the proper setup. If the location is west of the Mississippi river then magnetic declination needs to be considered for the azimuth angle.

For panels in the Northern hemisphere if the magnetic declination is positive, or east, the panels need to be rotated eastward at the angle of magnetic declination that accounts for the change in the Earth’s magnetic field lines. If it is negative, or west, the panels need to be rotated westward to account for the declination. This must be done because the earth’s magnetic field is not constant with what true north or true south is. A compass has slight errors in it due to magnetic declination and must be taken into account when setting up a panel as all calculations are based off true north.

Setting the tilt of a panel is simple if the panel is to be ridged mounted and not to be moved. The tilt angle is simply equal to what the latitude is in the location the panels are being set up. Depending on the location though the panels might need to be tilted plus or minus a few degrees to optimize them for certain seasons. If the system produces more power than it consumes in the summer than the system may struggle in the winter and if this is the case then the tilt should be angled up, plus, about 10 to 15 degrees to account for low winter production. This will affect the summertime production of the panels but for this system the summertime solar energy production will be more than what is needed and the locations in which the system might be placed is expected to have harsh winters.

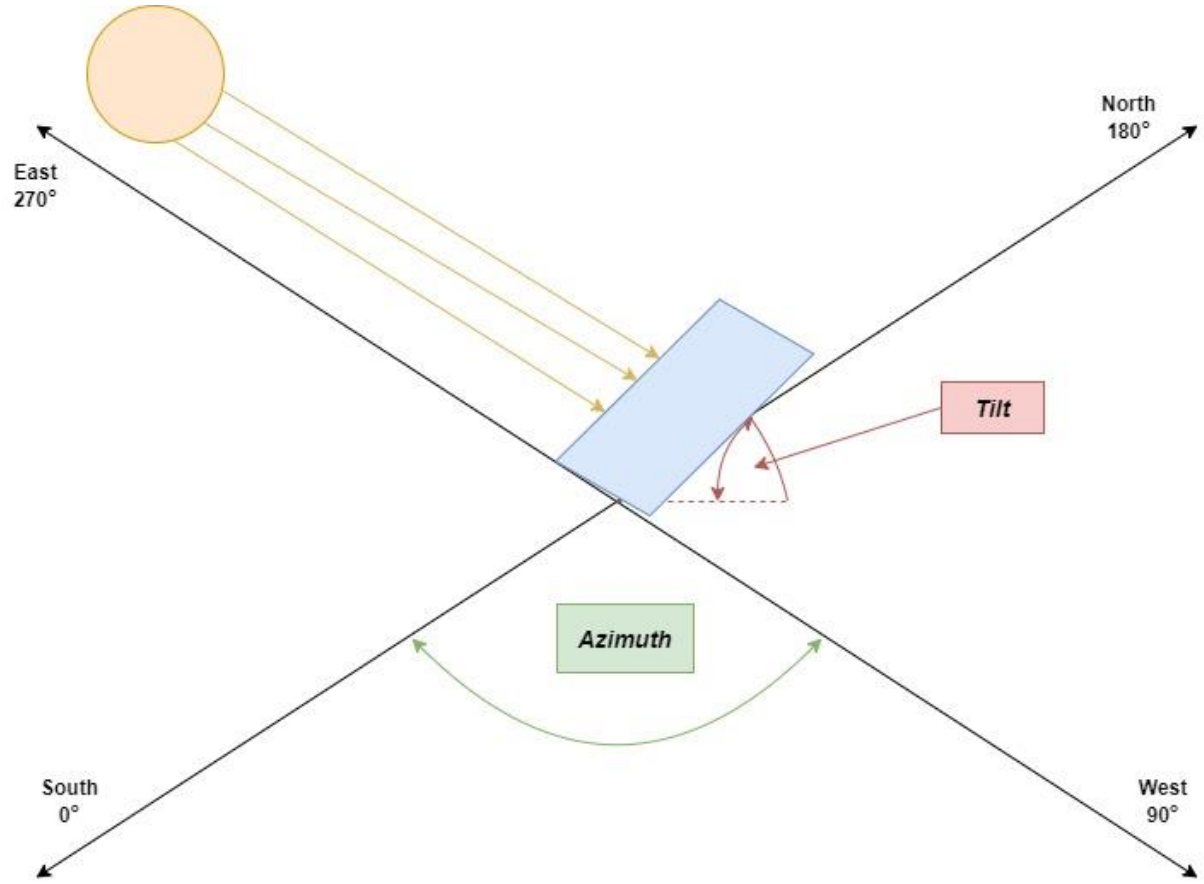


Figure 17: Azimuth and Elevation

A small note on solar tracking systems and their value to this project. While solar tracking technology does exist and could be designed and manufactured into this system it adds significant complexity and design, along with weight. Solar tracking systems cost between $600 to $1000 per panel depending on size and, at best, increase production by about 20%. For our project, the cost and added complexity means that they will not be used.

4.2.8. RF Design and Frequency Selection

To work effectively over a large area, the system needs to wirelessly transmit data. Different antenna designs and frequencies play a role in designing the circuits and choosing the components that will work with wireless technologies. This section investigates the different issues with RF design and some decision-making processes that took place to choose frequencies and other design elements.

4.2.8.1. RF Considerations

RF and wireless applications require that some specific design rules be followed. If these rules are not taken into consideration, then significant power and range issues may

present themselves. At low power levels, avoiding signal loss is very important. Any piece of wire can be considered an antenna. How well that wire radiates energy is dependent on the wire being resonant at the same frequency as the signal applied and that the feed point of the antenna is matched to the impedance of the attached transmitter. Direction and range are then determined by the design and shape of the antenna. It is possible that the radiated energy be aimed at a single point or that it radiates out in a sphere or doughnut shape.

The antenna must be the correct length at the frequency of operation, and it must have its impedance matched by the transmitter or receiver to operate correctly. Impedance matching maximizes the power transfer from the transmitter or receiver to the antenna.

4.2.8.2. Frequency Selection

The main discussion of Radio Frequency technologies comes down to range. Due to its massive support, the ideal choice is Wi-Fi. Wi-Fi, however, has limited range. The numbers found in the table below are averages. Actual distances depend on a variety of variables. The following numbers were compared to get the maximum range out of the device.

Table 5: Comparison Between Technology Ranges and Frequencies

Technology	Frequency	Range
Bluetooth	2.45 GHz	30 Feet
Wi-Fi	2.45 GHz (or 5GHz)	100 Feet
Zigbee	2.4 GHz	1000 Feet
FSK Modulation @ 900MHz	900 MHz	2+ Miles
LoRa	400 MHz / 900 MHz	10 Miles

Using the Free-Space Path Loss equation, the attenuation of radio energy between two antennas is determined.

$$FSPL = \left(\frac{4\pi df}{c}\right)^2$$

where d is the distance between antennas, f is the frequency, and c is the speed of light.

A graph that shows the best option for longer ranges and frequencies to minimize attenuation is generated by trying different frequencies and ranges.

Finally, due to regulations of how much power can be dissipated in the 400MHz bands, 900MHz is a good solution for global use and higher power dissipation.

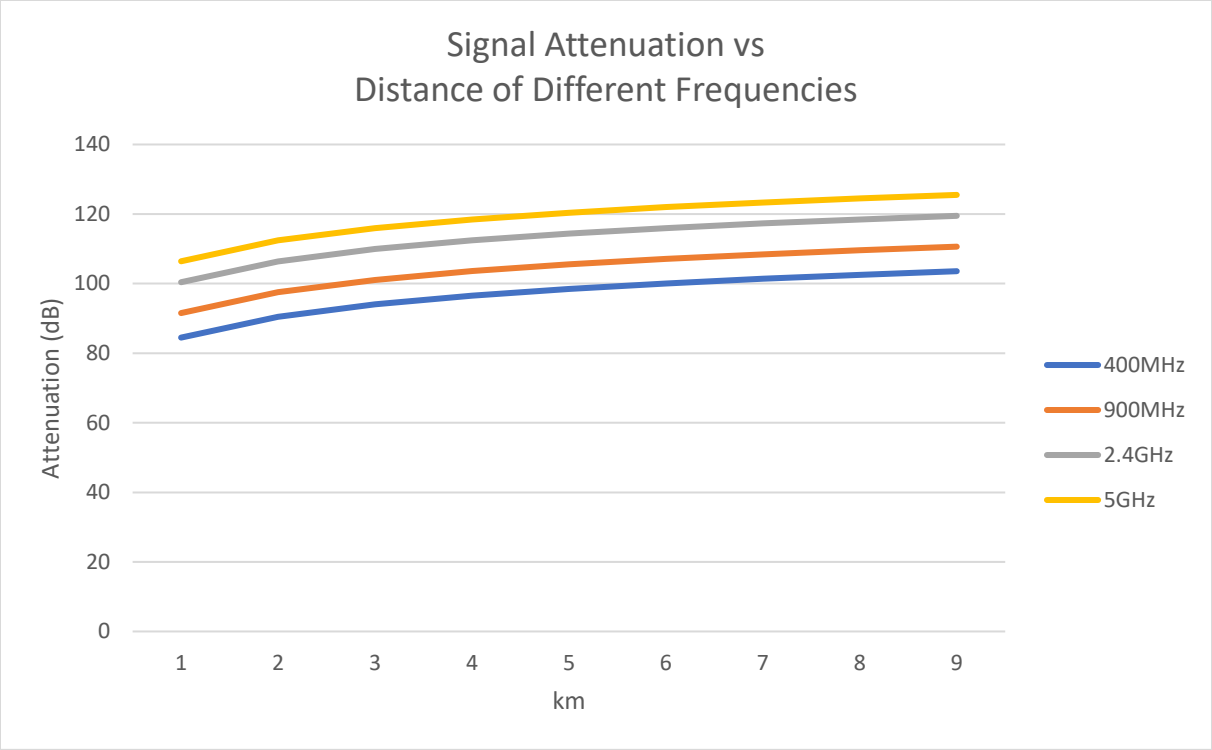


Figure 18: Signal Attenuation Vs Distance of Different Frequencies

4.2.8.3. Antenna Design

Antenna design plays a big role in RF applications. Without a proper antenna design the range and sensitivity of the device will be severely impacted. This section outlines some research in antenna design and considerations.

Whip Antennas

These kinds of antennas are designed for machine-to-machine communication but are not used in portable designs much anymore.[45] They are externally mounted, so they do not suffer interference issues from a PCB as much as other designs and they are not easy to detune. They are very useful for certain applications that could benefit from an external antenna.

Helical Antennas

These kinds of antennas are similar to whip antennas but instead of being a strand of wire externally mounted, they are copper that's wound in a helix shape. Since the frequency band is selected by the length of the antenna (among other factors), the antenna can take up less space since more of the copper takes up less area being wound in a loop. Due to their size and mounting style, they are fairly rugged [21] which means they can be put inside the mechanical housing of the device and can be hidden from view.

Chip Antennas

Chip antennas (Usually made from ceramic) are small and easy to put into a design. They have several advantages compared to larger antennas. They are not as sensitive to proximity interference and from other components. Furthermore, they are easier to

accommodate without simulation [22]. A downside as these are more expensive than a trace PCB but they are generally cheaper than other alternatives.

Trace Antennas
Trace antennas seem to be the cheapest but most difficult antenna to design. They are included in the cost of manufacturing the PCB. This means that, if designed correctly, the antenna is free! Furthermore, they are more tamper proof since it is embedded into the PCB. When tuned correctly they can operate in a wide bandwidth and have a good amount of network reliability [22]. A downside to these kinds of antennas are that they cannot be modified after manufacturing. Any changes to the antenna require redoing the board layout and having new boards manufactured again.

4.2.9. Machine Learning/ Computer Vision

This section will cover different methods/models that was considered to use for detecting fire using our system. These will cover different filters and adjustments that can be done to the images to help the system learn and identify the fire in an image or sequence of images. There are multiple ways to help the system identify the fire. It could be using deep learning through available pre-trained models or having functions such as optical flow or color classification to help identify the area of the fire.

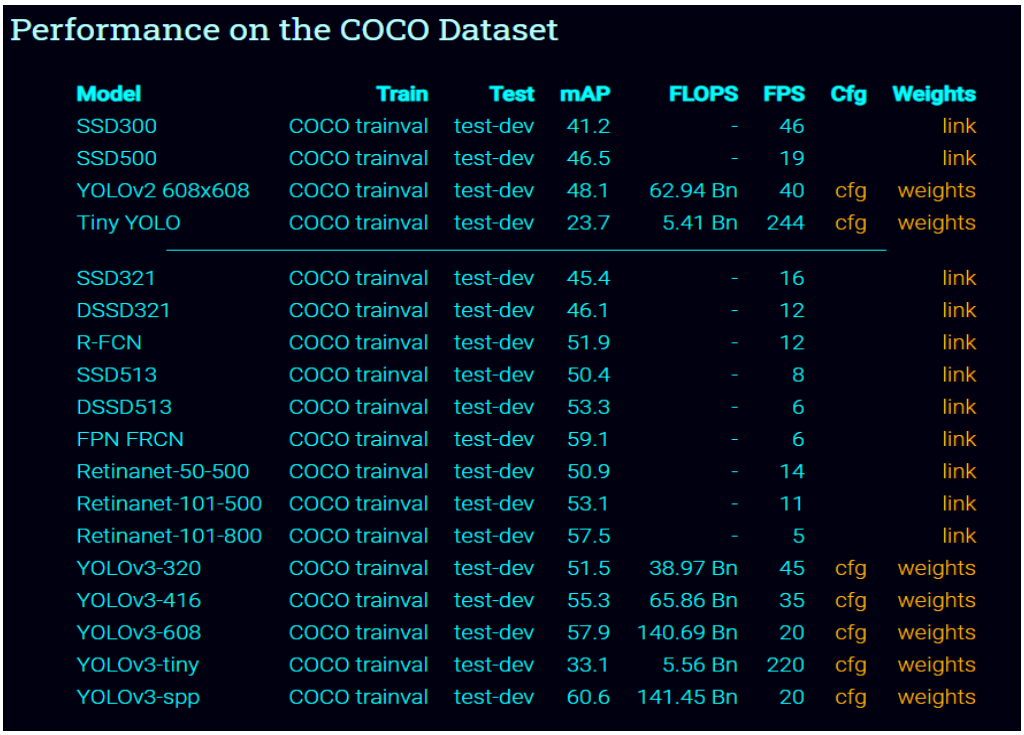
Our system will be able to ignore the background and its noises and identify the fire that is within an image or a sequence of images with minimal processing power via Raspberry Pi inspired by these methods/models.

4.2.9.1. Generic Object Detectors

There are several accessible neural networks such as YOLO and Faster RCNN via GitHub. Other neural networks were discovered that focuses on detecting fire instead of having functions such as object classifier. This subsection covers the comparison between them.

4.2.9.1.1. YOLOv3

YOLO (You Only Look Once) is one of the popular object detection methods. In fact, it is a state-of-art, real-time object detection system. It is a fully convolutional neural network (FCN) and has no pooling used [23]. By having no pooling, it avoids loss of minor features. It has great speed and accuracy compared to other state-of-art methods as seen in the figure 51 below which is the reason behind its popularity. For these reasons, YOLO is one of the top methods that come into our minds to implement in our system [23]. Thanks to its popularity, there are many tutorials as well as resources and forums available for this model which can help us understand and use it better.



Performance on the COCO Dataset

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

Figure 19: Comparison of other state-of-art models on the COCO dataset [23]

YOLO predicts and outputs feature map that has box coordinates, object score, and class scores as shown in the figure below. This means that it can classify and detect object at the same time. Since YOLO is a fully convolutional network, it can adapt to different sizes of images. However, it is recommended to have a constant input size to avoid adding complexity and issues during implementation. Since its accuracy and speed is applicable for real-time detection, this model is one of our top choices to implement. The reason behind the speed of YOLO compared to Faster RCNN is its use of confidence score to eliminate many of the predicted bounding boxes per object.

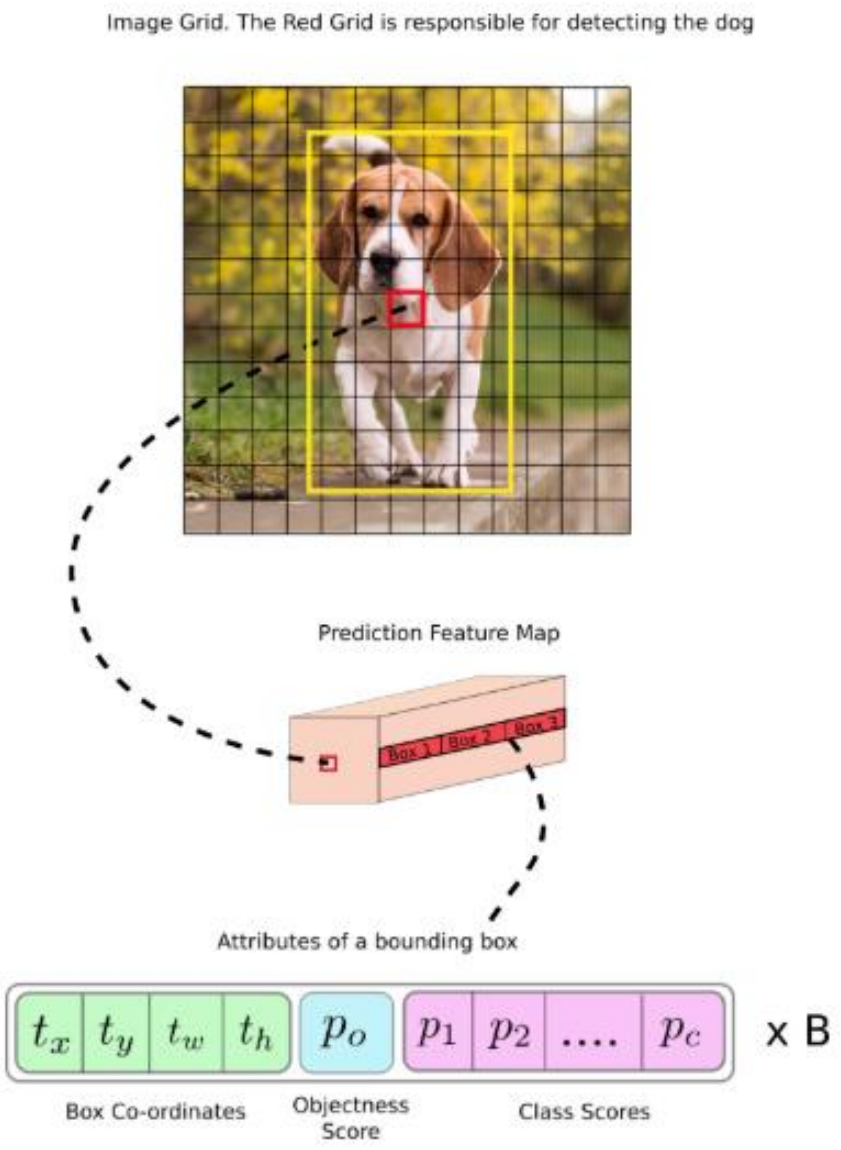


Figure 20: YOLO Bounding Boxes [24]

For an image of size 416x416, YOLO predicts 10647 bounding boxes. To reduce these boxes in order to detect a dog in the picture as shown in the figure above, it uses thresholding by object confidence score and non-maximum suppression as shown in the figure above.

YOLO offers great information such as bounding boxes for the detection of objects. It even classifies the detected objects from one another. However, most of these features are not needed for the purpose of our system. Our system focuses on binary classification of whether the flame exists in the image or not. There is a great possibility to utilize YOLO (or TinyYOLO) in Raspberry Pi Zero, but this required many adjustments. Through our

experiments, implementing YOLO is difficult and tricky due to the limitation of the Raspberry Pi Zero.

4.2.9.1.2. MobileNetV2

The main concern lies in whether our embedded system can handle all the computations and processing fast enough to detect the fire. Thus, other models were investigated that are commonly used in a similar set up as our system.

One of the models found that is commonly used with Raspberry Pi is MobileNet. This is another model that is accessible and most optimal for our hardware, Raspberry Pi Zero. MobileNets are low-latency, low-power models for mobile applications to perform object detection, classification, and segmentations. There are MobileNetV2 and MobileNetV3 available through GitHub [25]. These can also be used for real-time object detection and can be easily implemented by Raspberry Pis. Many examples are available online.

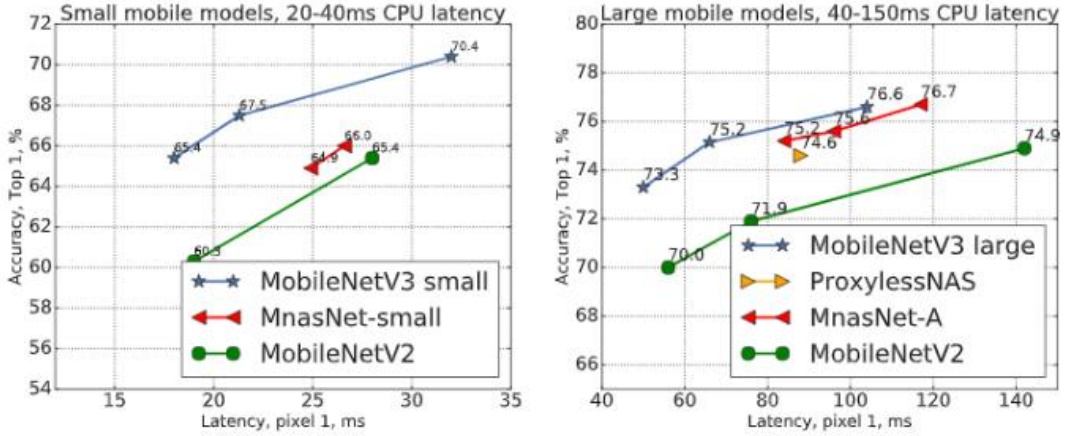


Figure 21: Comparison between MobileNetV2 and MobileNetV3 [26]

As seen in the figure above, MobileNetV3 has better accuracy compared to MobileNetV2. However, the MobileNetV3 being new makes the MobileNetV2 preferable to use when it comes to training the models ourselves because of the details that could be found about the hyperparameters of MobileNetV2 in the GitHub and not of MobileNetV3.

4.2.9.2. Neural Network Frameworks/ Services

Different frameworks exist to help with the implementation of neural networks. When designing software around a neural network framework, it is important to discover the differences in each framework. The strengths and weaknesses of each framework will determine which framework is used for the project and how well it performs. The best framework, which most optimal for the capacity of the Raspberry Pi Zero, was taken into consideration.

4.2.9.2.1. Keras

Keras is a high-level neural networks API, written in python. [27] It allows easy prototyping of a model and runs on both CPU and GPU. It is easy to use and beginner-friendly, but it does not allow many modifications to the model like Pytorch does. There are simple examples available online to test and create your own neural network architecture quickly. There are some models such as Faster RCNN that are coded using Keras. Keras can also accommodate to Raspberry Pi which makes Keras one of our top frameworks to utilize.

4.2.9.2.2. PyTorch

PyTorch is an open source machine learning framework that excels in researching prototyping and production. [28] Pytorch is known to be harder to implement than Keras, but it provides more flexibility and features. Most of the models available publicly are coded using PyTorch as it is one of the top used frameworks when it comes to machine learning researches as it offers fast and dynamic training.
Many industries also look for proficiency in this framework as they also use this as their main framework. Understanding and being able to use PyTorch should be useful for us in long term and a good skill to have. Knowing how to use this framework should indicate a good understanding of machine learning.

4.2.9.2.3. TensorFlow/TensorFlow Lite

TensorFlow/ TensorFlow Lite is an open source deep learning framework for on-device inference. [29] It is commonly used with the Raspberry Pi Zero. To implement deep learning, we would need to install this to the system. Most of the tutorials we encountered use this framework especially for Raspberry Pi. It is also most optimal for integrating AI into a product.

4.2.9.2.4. OpenCV

OpenCV is an open source computer vision library [30] that has many computer vision applications. OpenCV is the best choice among the others we mentioned when utilizing CPU. This is because it has many libraries and models that are optimized for CPU use. The models mentioned earlier can be implemented using OpenCV using their pre-trained models. OpenCV has many libraries available for us to use and is popular enough to have many resources to help us guide through the process.

4.2.9.2.5. Google Colab

To train the model, we need to perform it outside of Raspberry Pi Zero as it takes too much processing power. Google Colab allows usage of GPU for faster training and has many necessary packages installed already. It is user friendly and has easy access to Google Drive which makes retrieving dataset easier if it is saved there.

4.2.9.3. Frame Differencing

Frame differencing is where we simply take a difference of values between the two images to see where the movement is significant. This is a way of capturing the temporal information between the images. Since flames will flicker and spread, it should have much more movements compared to the background. Thus, we expect the frame differencing to be effective in identifying the flame. One of the main concerns for using frame differencing is distinguishing the flames from other movements such as leaves swaying or movements of animals in the forest. These can be solved by eliminating background noises and noticing a big difference if an animal is detected. Applying additional filters such as blurring, or normalizing will help the model detect the fire from those subtracted images.

An example of frame differencing can be seen in the figure below. The white values indicate high differences or movements between the frames. In the example, it is noticeable that the movement was significantly recorded for the human and the flame.

Figure 22: Frame Differencing [31]

There are other researches done on computer vision with fire detection using frame differencing. The method by the Ministry of Public Security of Shenyang Fire Research Institute shows how the smoke is also being detected via frame differencing as seen in the figure below. This is an interesting concept as we initially disregarded the idea of smoke being detected in our system. However, it is one of great indicators of forest fires and identifying smoke in our system can warn or notice fire in great distance.

An example of frame differencing from the Ministry of Public Security of Shenyang Fire Research Institute is shown in the figure below. It shows both cases where the smoke is detected, and the flame is also detected by itself through the help of frame differencing. As seen in the figure, the smoke can be large and seem to be easily recognizable. This may mean that the smoke detection may possibly be added to our system. Furthermore, these examples show how effectively can frame differencing isolate desired subjects of fire and smoke in order to alert the system. This is because the fire and smoke have distinct movements compared to the other movements in the background. They also have patterns that can be recognized using frame differencing which also helps distinguishing them.

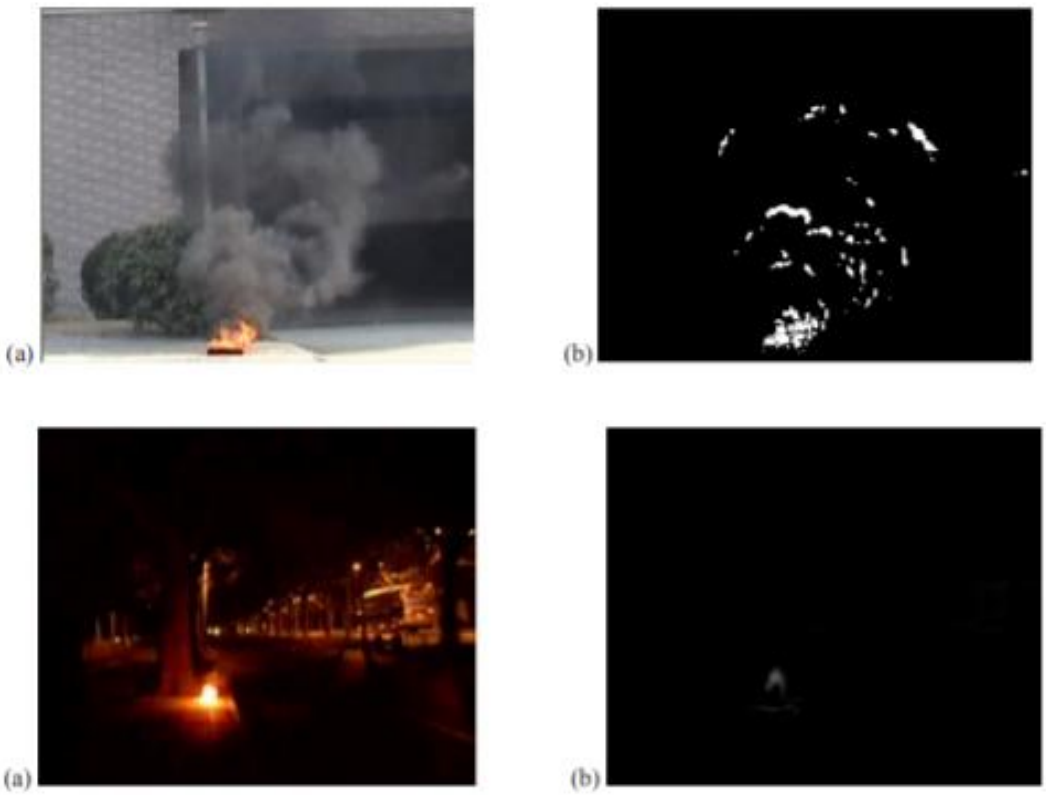


Figure 23: Frame differencing continued [32]

Frame differencing can greatly help in distinguishing the background from the flame and smoke by capturing the flickering movements as seen in the previous examples. Being able to detect smoke is an additional feature that may significantly improve the system's effectiveness and utility. For instance, even if small flame was missed, the smoke can trigger the system earlier rather than waiting for the flame to be large enough to be recognizable.

4.2.9.4. Color Classification

Color classification is a method to classify area of an image by its color values. It can distinguish different color values, hues, and saturation. It is a simple yet effective method to add into a system. Since our focus is forest fires, the fires should have significantly different color from the other objects in the background. Thus, we expect the system to be able to provide better results by applying color classification as part of its identification process. Adding color classification should further help in narrowing the computation time as well as increased accuracy by providing better predictions of where the fire may be.

Compared to frame differencing, color classification may run into more ambiguous detections, since forest contains many objects that can be detected as false positives

such as red flowers, woods, and leaves. To avoid confusion between different objects, focusing on specific unique color that is most applicable to flames should increase accuracy. However, choosing a specific value of color to detect fire as a threshold is tricky as it increases false alarm rate or lower overall accuracy.

But the application of color classification to a system is much easier compared to frame differencing. OpenCV's functions help ease the application of color classification or dissect images into the RGB layers and focus on the R layer alone to help the model detect the fire. OpenCV is also a reliable library that provides color classification and shape detector. It also has a packages for Raspberry Pi. The figure below shows an example of color classification and shape detector using OpenCV libraries alone. It computes the center of the contour, perform shape detection and identification, and color labeling by taking averages of a particular image region.

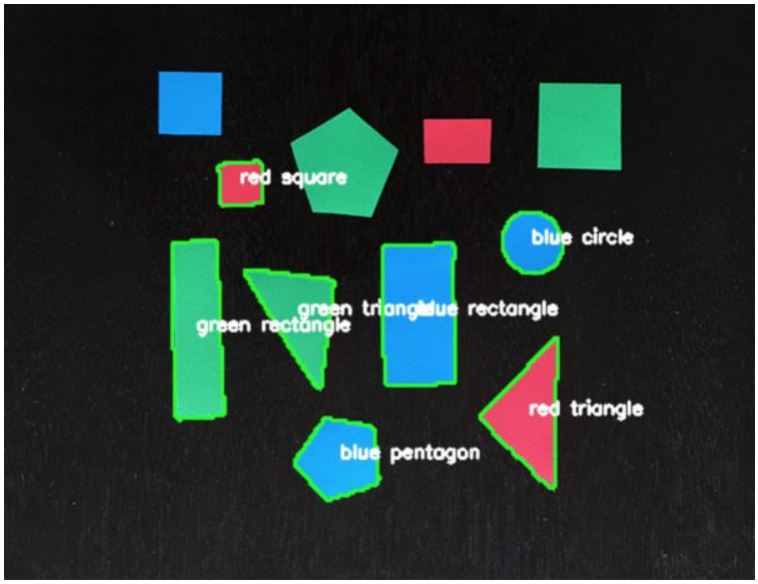
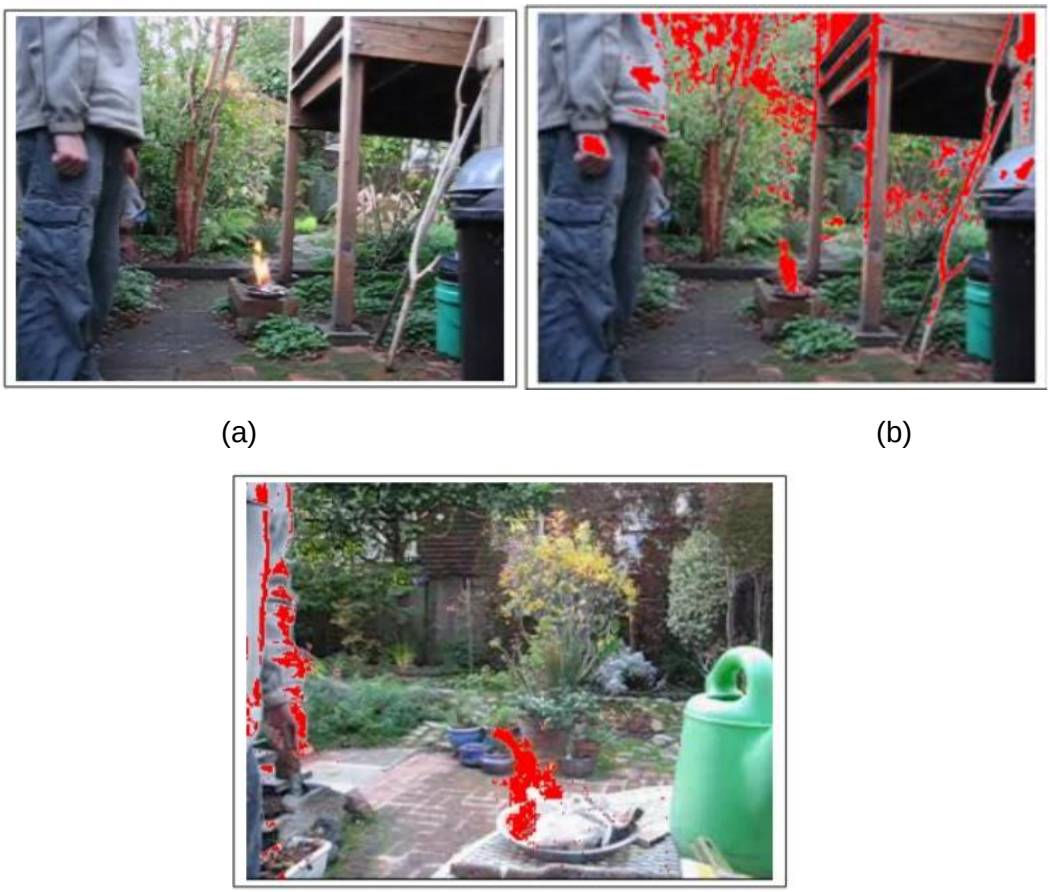


Figure 24: Color Classification and finding contours using OpenCV [30]

There are other researches that also implemented color classification in order to detect fire. Figure 25 is an example of a research that utilized the color classification in order to detect the fire. As you can see in the example, flame is within the predicted area using the color classification. However, other objects such as wood or person are also detected as false positives. In the research, they were able to minimize these false positives by adding motion along with the color classification to narrow the predictions down.

Color classification may greatly be enhanced by adding other methods such as frame differencing to minimize false positives by isolating objects that have motion and desired color value. By doing so, leaves, trees, and structures may easily be distinguished from the fire as it will not have as much of flickering movements as the flames will have. By eliminating most of it, it should significantly help our system to learn or identify the flames from others.



(a) (b)

(c)

Figure 25: Color of Fire Classification [31]

(a) original image (b) red denotes pixels that were classified as being the color of fire (c) color classification with motion

Since we are aiming to detect fire in a forest setting, these flames will be very distinguishable compared to the other objects in terms of color. Adjusting different settings such as saturation and filters may also help in identifying the fire and distinguish it from the rest of the background.

4.2.9.5. Optical Flow

Another method to detect motion is optical flow. This method can be implemented using OpenCV [33]. Optical flow shows the vector or density of an object's movement between two consecutive frames. The dense optical flow in OpenCV uses Gunner Farneback's algorithm. In this method, the direction corresponds to hue value while the magnitude corresponds to the value plane. An example output can be seen in the figure below.

Figure 26: Dense Optical Flow [33]

Top image in the figure above is the original image while the bottom image shows the result of dense optical flow via OpenCV. Optical flow is another method to detect motion like frame differencing. Optical flow adds a bit more complication than the frame differencing, but the available OpenCV library helps us implement this method with ease much like color classification.

By taking in consideration that the flames flicker in concentrated area spread slowly, optical flow can best illustrate this dense movement in the sequence of images and identify flame. This is an easier implementation than frame differencing as we do not need to experiment our own filters and thresholding as much as frame differencing would need.

Optical flow also helps us distinguish other movements such as animals moving by comparing the density and the vector of the movement. OpenCV optical flow is able to ignore the background noises which can help reduce false positives. Thus, this method is very effective in detection motion while identifying its density and vector.

4.2.9.6. Superpixel Localization

Another method we found interesting and effective is superpixel localization. Instead of looking at the whole image, pixel by pixel, or by looking at bounding boxes, it localizes objects by segmenting the image into perceptually meaningful regions similar in texture and color.

A research from Durham University [34] shows how they were able to effectively detect fire using superpixel localization and a network architecture with reduced complexity. By using superpixel, they were able to increase accuracy without adding complexity to the network architecture and with no temporal information. Their research shows that using superpixel significantly outperformed other works in the non-temporal fire detection.

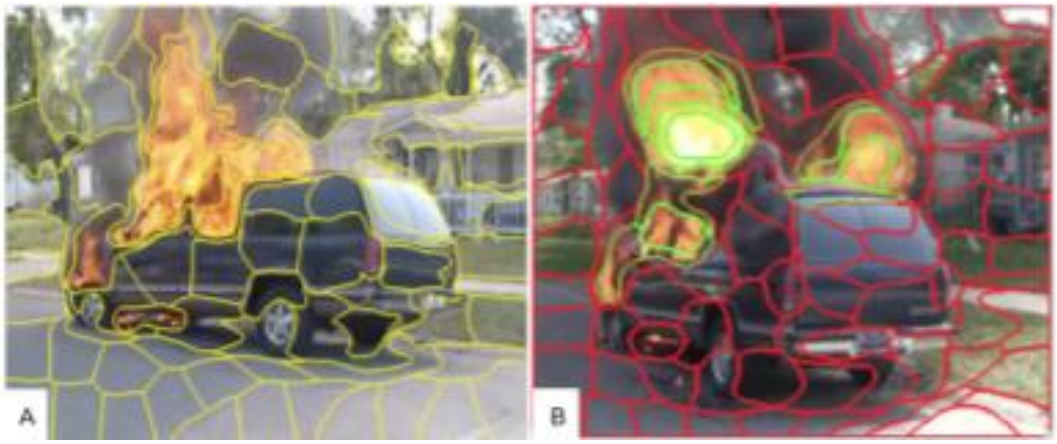


Figure 27: Superpixel Localization from Durham University [34]

This method is also available in GitHub created by Toby Breckon [35]. He uses FireNet and InceptionV1-OnFireNet architecture shown in the figures below along with the superpixel localization explained in the research. These netowrks have binary detection architectures that determine whether an image frame contains fire globally. However, by adding superpixel localization, it breaks down the frame into segments and performs classification on each superpixel segment to provide in-frame localization. The superpixel localization uses SLIC algorithm. For the best performance and throughtput, use the FireNet model.

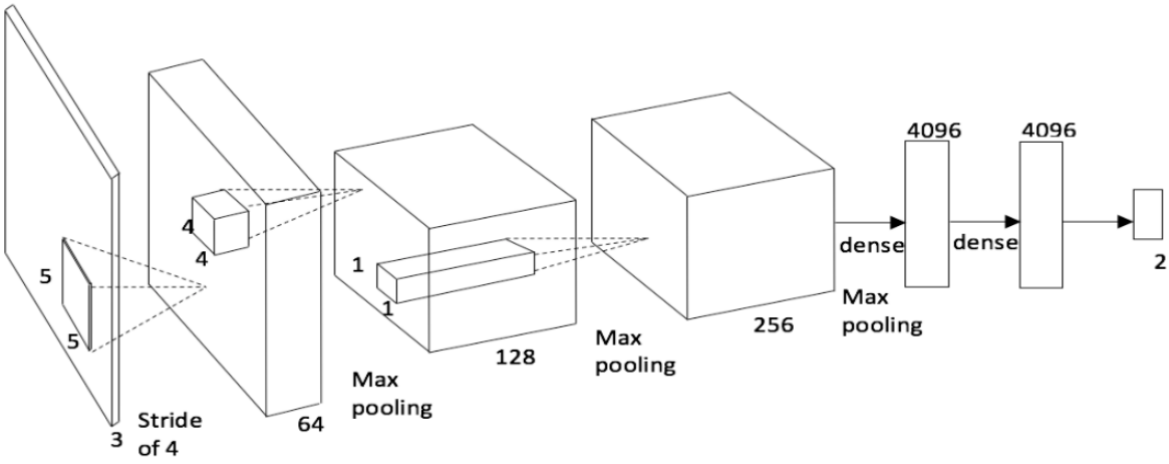


Figure 28: FireNet Architecture [35]

If slightly lower false alarm rate is desired despite having lower throughtput, then use the InceptionV1-OnFire model shown in the figure below.

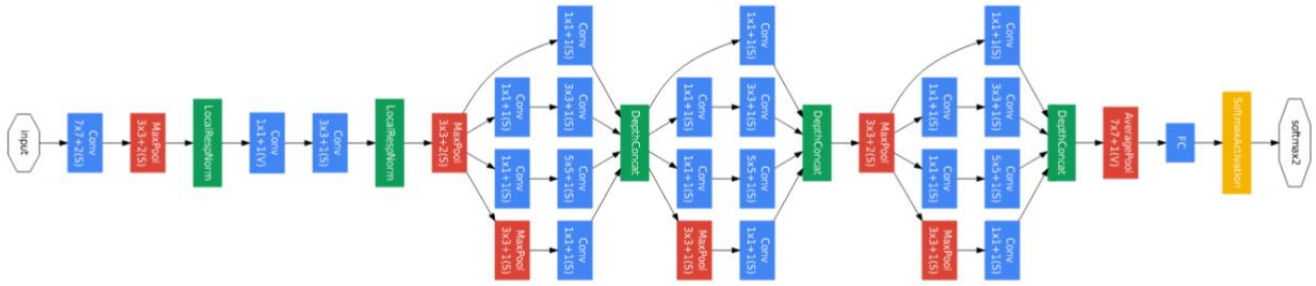


Figure 29: InceptionV1-OnFireNet Architecture [35]

An example output is shown in the figure below. As seen in the figrue, it was able to successfully identify the fire in the given image by selecting the correct superpixel regions associated to the fire.

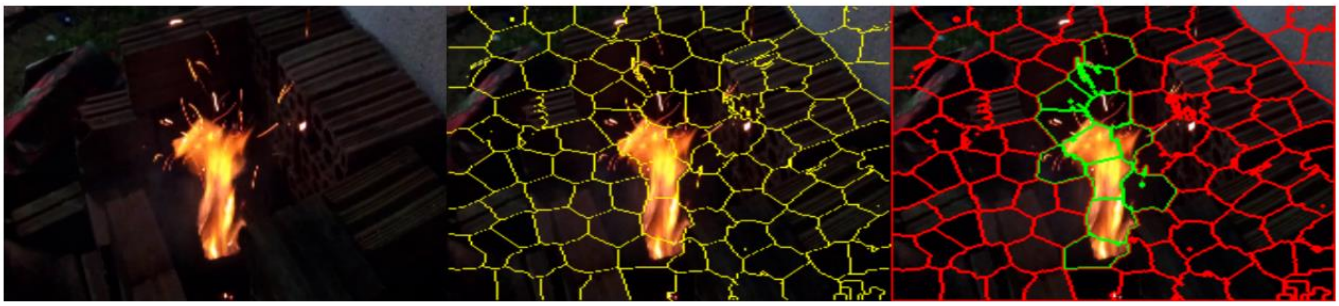


Figure 30: Implementation of Superpixel Localization with CNN [35]

Left Image: Original image, Middle Image: Superpixel Localization, Right Image: Predicted Fire Regions (Green)

These models are available in pre-trained form using the dataset found in the Durham Collections. Both models were able to achieve over 90% accuracy using that dataset according to the Durham University's research paper.

The superpixel localization can be applied by using the OpenCV as it provides three different algorithms we can choose from to perform superpixel. They are SLIC, SLICO, and MSLIC as shown in the figure below. [30]

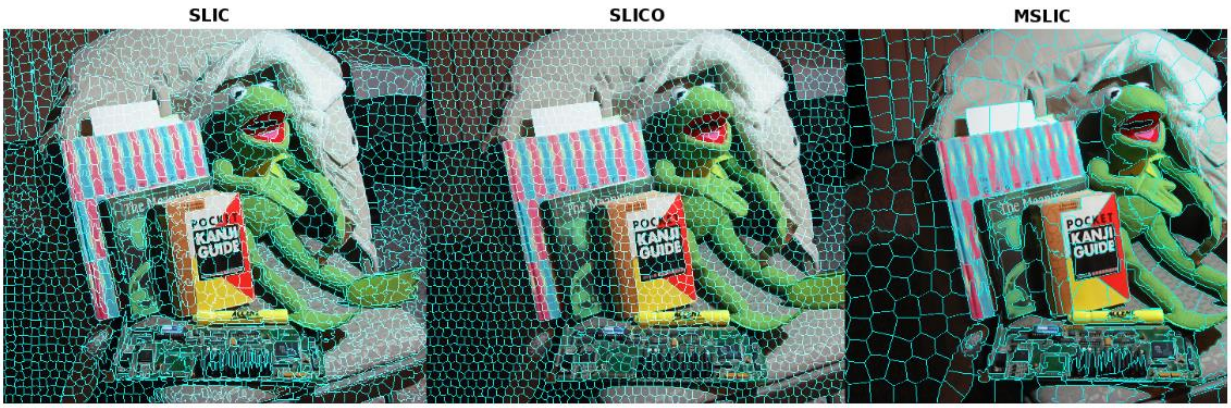


Figure 31: Superpixel Localization using OpenCV

4.2.9.7. Original CNN (Convolutional Neural Network) Design

There are other available models or neural networks out there but not much that are publicly available for us to implement especially with the Raspberry Pi Zero. To solve this problem, we can create our own CNN architecture. However, this means finding a novel way in such a short amount of time with limited resources such as accessible GPU and datasets. This route seems very impractical for our project considering there are already readily available CNN that may fit fine with our goals.

Combining all these methods' advantages, we may be able to create something novel and much more effective system than what is out there. For example, combining faster RCNN and YOLO may result in better model. Utilizing color classification and frame differencing will also help in creating better accuracy for the model [10]. Or optical flow with one of the generic object detectors may work fine as well. Another design we may add to our model is changing its hyperparameters to optimize for detecting fires. Reduction of latency is also possible by eliminating unnecessary features that come with the pre-trained models and libraries such as classification and segmentation.

However, due to time constraint, testing all the methods mentioned above is not possible as it will take time to design, program, train, and test the model. Tweaking the hyperparameters alone would be tedious and take tremendous amount of time to find the best values for the models to perform. In addition, it does not provide much scalability and promising improvement.

Another option to ease the heavy computation usually brought from the models is to create a much simpler CNN architecture concentrating on binary classification and just identifying whether the fire exists or not. This model can be improved by adding the other methods mentioned such as color classification or motion detector to eliminate false positives in the early stage.

4.2.9.8. Settings for Machine Learning

4.2.9.8.1. Programing Languages

Python is used for our programming language to utilize machine learning. Many resources use Python as well which makes it easier and efficient for us to use. Python is also easy to learn and ideal for people who wants to learn a new programming language. Thus, making it easy to follow or understand the code by itself too.

4.2.9.8.2. Hardware

We chose Raspberry Pi Zero to implement computer vision in our system. It has low cost while providing decent amount of memory and speed for our system. Another system we were considering was Raspberry Pi3. It was widely used and had several tutorials available online to implement computer vision. However, it had much higher cost compared to raspberry Zero. Furthermore, our system does not require the output to be instantaneous. Our priority for our system's goal is for it to be able to detect fire fast enough to relay the message to the other systems. Thus, it was decided that the Raspberry Pi Zero should sufficiently perform and meet our goals while saving us significant cost.

4.2.9.9. Dataset

For our model to have a good performance, it is optimal for us to train our models instead of utilizing the pre-trained models available online. However, this requires us to create our own dataset to train and test our models. Usually, a dataset would contain thousands of images for the model to learn from. The number of images can be increased improve its performance if adjusting its hyperparameters seems ineffective. Another possible source of a dataset would be utilizing similar dataset other research mentioned earlier has used.

4.2.9.10. Summary and Update

Utilizing Raspberry Pi Zero caused unexpected obstacles during our development of the software for machine learning. It has limited access to many packages that TensorFlow offered. The processing is too slow to install many of the packages that are needed to implement most of the pre-trained models available online such as YOLO. Furthermore, Raspberry Pi Zero is not optimal for real-time object detection due to its limited memory and processing power. It cannot handle complicated neural network structures with many parameters. Thus, simple convolutional neural network is best used for this set up unlike YOLO. There are many complications just to install the right packages for Raspberry Pi Zero. However, OpenCV can be successfully installed and it had enough processing power to run a pre-trained model for one or two images (sacrificing some processing time).

As mentioned earlier, Raspberry Pi Zero cannot handle much processing and kept running into issues with necessary packages. Therefore, the generic object detectors mentioned earlier were not successfully implemented. To handle this issue, resources

designed specifically for Raspberry Pi were tackled first. Upon searching for more, Fire Detection Net [36] were discovered which has simpler neural network architecture compared to the generic object detectors and other fire detection computer vision research to accommodate for Raspberry Pi capabilities. From there, we ended up adjusting the layers and hyperparameters to achieve acceptable performance. The designs and architectures of the neural network is discussed later in the section 5.4.2. This was time consuming and did not allow us to further implement the other networks.

Keras, TensorFlow, and OpenCV are sufficient and can be operated in Raspberry Pi Zero for application of computer vision to the system. Although not all the versions worked well with it. Keras 2.1.5 and TensorFlow 1.8.0 were the only ones that properly worked during our development. This was easier to install via Python 2.7 version. Different versions would have issues with installing or managing the code. The whole process for the TensorFlow and model to load would take few minutes. However, the Raspberry Pi is not capable of training the model. Therefore, Google Colab is used to train and test the model before implementing into the Raspberry Pi. Google Colab can be either Python 2.7 or 3 as long as the model is trained with the same version of Keras and TensorFlow. Otherwise, there will be problems in loading the model in Raspberry Pi.

The color classification went smoothly as researched beforehand. However, to detect motion, dense optical flow was utilized instead of the frame differencing. This is mainly due to the time constraint. Optical flow is much easier and efficient to use compared to frame differencing. Through testing, it showed it can detect motion accurately. In contrast, frame differencing requires more refining after subtracting the images without the assurance of improvements. The same reasoning applies to superpixel localization. There was not enough time to test without guarantee of improvements.

For the dataset, there was an easier way to create an original dataset consisting of few hundred images by using the Microsoft API which is explained more in 5.4.2.1.

4.3. Component Research

An even narrower view than before is the selection of individual components. In the following subsections, different components are compared to see if they make a good fit for the system. These components, and their selection, will take parts of the previous, higher level, sections and focus on individual aspects that set the components apart from the others and lend themselves to a good design.

4.3.1. Controller Selection

The fire detection system operates in two parts: Process the sensor data and process the network data. Since these devices are wireless and need to be put out over a wide area, controllers that can support wireless communication and also process the sensor data

are needed. Some options, initially investigated, were the MSP430 family, ATMega family, and Espressif ESP32 controllers as they are popular options for controllers and have a wide variety of resources. As discussed in the previous section about Radio Frequency communication Technologies, LoRa modulation was chosen as the physical layer for communication. Because of this, the MSP430 and ATMega microcontrollers fell out of favor since they do not inherently handle RF communications. They would have to be interfaced with another circuit to implement the RF design and that would increase cost and complexity. To remove this complexity, the SAMR35 was chosen. It contains a LoRa and FSK modulator/demodulator built into the chip. Combined with its low power usage and high RAM options, it is a good choice.

Note: All values are shown at their highest possible offering if multiple values are given.

Table 6: Comparison of Microcontrollers

Controller	RAM	Flash	Avg Power (mA) (Tx/Rx/Run/Sleep)	Sleep mode	Wireless
MSP430	66KB	512KB	-/-/2.36/.00045	Yes	No
ATMEGA4808	48KB	6KB	-/-/11/.0001	Yes	No
ESP32	520KB	4-16MB	240/100/30/.005	Yes	Yes
SAMR3x	32KB	256KB	95/16/4.5/.0008	Yes	Yes

Finally, the matter of sensor reading, machine learning, and image processing was discussed. To handle this simply and quickly, the Raspberry Pi Zero was chosen since it can run Python code (making the software easier to write and maintain) and it has a relatively low power consumption: about 100mA.

4.3.2. Radio Frequency Communication Technology

Since the 900MHz band was chosen as our frequency of choice, there are only a few simple to integrate solutions on the market. IEEE 802.11ah would be ideal, however it is not quite ready for the industry just yet.

This leaves only a few viable options like XBee and LoRa.

LoRa appears as the best modulation technique as it is simple, has some examples, and has some resources to pull from. Furthermore, the SAMR35 microcontroller already has the LoRa modulation scheme built into the chip. Therefore, LoRa was chosen as the modulation technique. Finishing out the project, however, showed that the LoRa implementations shown online used external transceiver ICs (like the SX127x series chips) or were made for LoRaWAN. This is different than LoRa as it is a MAC layer built on top of LoRa. This made for some interesting challenges as discussed later in the paper.

LoRa is still a good choice and so more time would have been needed to complete the project as intended.

4.3.3. Fire Detection Sensors

The solution chosen to tackle to the environmental issue of forest fires has been narrowed down to sensor detection based on past research and implementation and understanding of forest fire behavior. The devices detect fire using three main approaches: flame detection, gas detection, and smoke detection.

Figure 32: Hydrogen sensor mounted to a tree during an experiment done in Humboldt University in Berlin, Germany. [9]

This approach provides an efficient method of detecting frame light in the nighttime, by identifying infrared radiation, but also during the day by sensing specific gasses, smoke particles, and beams of bright light (flame) [37]. Flame sensors observe the wavelength of a burning flame using infrared sensors as transducers. Gas sensors are designed to detect the concentration of specific gases in the atmosphere also using infrared sensors. When the concentrations reach the sensor's maximum reading, an alarm is triggered. The common gasses released during a fire emission include carbon monoxide, carbon dioxide, hydrogen, nitrogen dioxide, sulfur dioxide, and volatile organic compounds [14]. Smoke detectors work by emitting alpha particles to the atmosphere. When smoke is present, the ionized air molecules interact with the smoke. Other smoke detectors function by emitting light to its surrounding; the presence of smoke will cause light shattering which sends the signal of a smoke alarm [14].

Camera surveillance is also another technique that has been used by other systems. In this case a video camera system is set up in the forest and used to recognize a spectrum of smoke and fire during the day and night [3]. Other techniques use thermal cameras to detect heat and glow of a fire. Moreover, Infrared spectrometers have been used to observe specific visual characteristics of smoke. Lastly, LIDAR (light detection and ranging) have been used to measure reflected rays from smoke particles [3]. Cameras provide a range of opportunities however there are challenges associated with using them; cameras record images with a number of pixels and observes the motion between the images to compare pixels to the characteristics seen in a fire. This comparison is done through an algorithm [3]. Such optical systems are usually integrated with local maps.

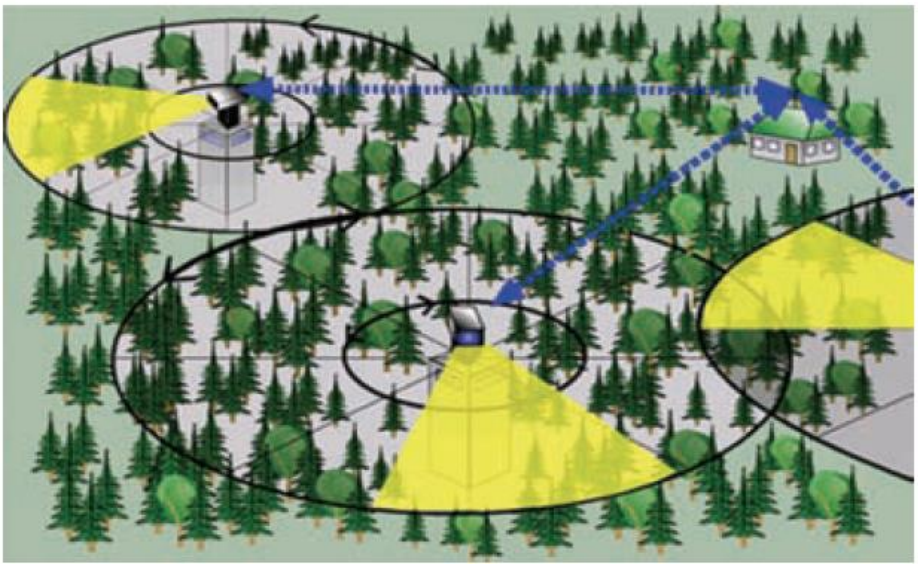


Figure 33: FireWatch adopts a similar concept to our method of scattering sensors in a forest, except their system uses cameras [3]

An additional advantage to our approach is the use of wireless sensor networks. A wireless sensor network is a cluster of “low- cost battery-powered sensor nodes” that uses wireless communication [38]. A wireless sensor network mainly includes numerous sensor devices that typically use low power, low processing memory, and low bandwidths [39] Within this network will be a wireless mesh network, which is defined as a “multi-hop wireless network formed by a number of stationary wireless mesh routers” [38, 39]. By creating a network of sensors that communicate with each other and send updates to the central hub, we are able to identify localized and sweeping fires occurring in a forest. Long Range Wireless Data Telemetry, which uses bi-directional VHF / UHF radio frequencies, has been studied and suggested to connect multi-node fire sensors and GPS to create a fire detection prototype with promising results due to its wide range [40].

4.3.3.1. Sensor components.

Based on the research, the following components were considered as sensors to be used in the system based on electrical characteristics (supply voltage), cost, I2C compatibility,

and principle of detection. Each sensor listed provides an advantage considerable enough to be worth exploring.

Table 7: Gas Sensors

Gas Sensor Name	Op. temp.	Comm. protocol/ Output type	Op. Voltage	Cost	Notes
Renesas Electronics America Inc. e ZMOD4510 Gas Sensor Platform Smoke Sensors	-40 ~ +65 C	I2C interface Up to 400kHz	1.7V – 3.6V	5 for $56.13	Displays air quality index (NOx) and ozone (O3) (20 – 500ppb).
AMU gas sensor	-5°C ~ + 50°C	Analog output with Analog to Digital Convertor	1.4V	5 for $40	CO2 (eCO2) range from 400ppm up to 29206ppm. eTVOC range for CCS811 is from 0ppb up to 32768ppb.
Senseair CO2 sensor	0 – 50°C	UART, Modbus protocol	4.5-5.25V	5 for $211.05	Non-dispersive infrared (NDIR) principle. Signals alarm output. CO2 400–2000ppm. Can go up to 10,000ppm in extended range
AS-MLV-P2 Air Quality Sensor	up to 300°C	Analog output, requires ADC	3V	5 for $84	Sensitive to humidity changes and temperature changes. CO, butane, methane, ethanol, hydrogen from 0 to 6000 ppm
Multi-gas, humidity and temperature sensor combo module	5 - 55°C	Digital I2C interface	5V	10 for $20	Measures indoor air quality parameters total VOC (tVOC), CO2-equivalent (CO2eq), relative humidity RH and temperature. a typical accuracy of ±5 %RH and ±1°C. Gasses: 0 – 60000ppm Humidity: 0 to 100 %RH

					Temperature: –20 to 85 °C
Sparkfun Gas Sensors	5 - 55°C	Resistor to Analog to Digital conversion needed.	5V	7 for $30	Alcohol, LPG, Methane, Carbon Monoxide, Hydrogen. Gas concentrations 200 to 10000ppm.
Gravity: Analog Gas Sensor (MQ2)	20℃-50℃	Analog output		$6.9 for 1.	Application gas leakage detecting equipment in family and industry, are suitable for detecting of LPG, i-butane, propane, methane, alcohol, hydrogen, smoke.
Renesas Gas Sensor Module for TVOC and Indoor Air Quality	Up to 300 ℃	I2C		10 for $83	Detecting total volatile organic compounds (TVOC) and monitoring indoor air quality (IAQ) in different use cases. Measurement range: 200ppm-5000ppm LPG and propane 300ppm-5000ppm butane 5000ppm-20000ppm methane 300ppm-5000ppm H2 100ppm-2000ppm Alcohol
Adafruit MiCS5524 CO, Alcohol and VOC Gas Sensor Breakout	Up to 80℃	Output is a resistance, analog voltage proportional to gasses detected	5 V	1 for $20.	Output does not identify gas detected. CO (~ 1 to 1000 ppm), Ammonia (~ 1 to 500 ppm), Ethanol (~ 10 to 500 ppm), H2 (~ 1 - 1000 ppm), and Methane/Propane/Iso-Butane (~ 1,000++ ppm)
Adafruit BME680 - Temperature, Humidity, Pressure and Gas Sensor	Up to 80℃	SPI or I2C		1 for $22.	Temperature, humidity, barometric pressure, and VOC gas. Must be calibrated. Detect gasses & alcohols such as Ethanol, Alcohol and Carbon. Must be calibrated Humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy.

Selected sensor:
In the end, the Adafruit BME680 was chosen since it is a 4 in 1 sensor that provides temperature, humidity, pressure, and gas measurements. It used an I2C protocol and requires a supply voltage of 3.3V to 5V. The sensor also had an IIR filter that is used for temperature compensation and to provide accurate measurements for gas, temperature, humidity, and pressure. The cost of the sensor also falls within a reasonable range.

Table 8: Smoke Sensors

Sensor	Op. temp.	Op. Voltage	Output	Cost	Notes
CMOS Photoelectric Smoke Detector ASIC with Interconnect	-25°C to 75°C	12V	Output local alarm	25 for $17	An internal oscillator strobes power to the smoke detection circuitry for 100us every 8.1 seconds to keep standby current to a minimum. If smoke is sensed the detection rate is increased to verify an alarm condition. A high gain mode is available for push button chamber testing.
CMOS Ionization Smoke Detector ASIC with R&E International Interconnect and Timer Mode	-10 to 60°C	15V	Output: local alarm	25 for $16.50	The smoke comparator compares the ionization chamber voltage to a voltage derived from a resistor divider across VDD. This divider voltage is available externally on pin 13 (VSEN). When smoke is detected this voltage is internally increased by 130mV nominal to provide hysteresis and make the detector less sensitive to false triggering.
CMOS Low Voltage Photoelectric Smoke Detector ASIC with Interconnect	-10 to +60°C	5V	Output signal: local alarm	25 for $27.25	The RE46C190 is a low power, low voltage CMOS photoelectric type smoke detector IC. With minimal external components, this circuit will provide all the required features for a

and Timer Mode					photoelectric-type smoke detector
PIM-438 Smoker sensor (and oximeter)	-45 to 80°C	3 – 5V	I2C	$16.30	Uses the MAX30105 module and is used for heart rate, oximeter, smoke sensor.

Selected Sensor:
In the end the PIM-438 was used for smoke detection. This sensor has 3 LEDs (Red, Infra-Red (IR), Green), a photodiode, and an analog front end. The sensor uses the photoelectric principle to detect smoke. The sensor uses infrared light (invisible to the human eye) to check for the presence of airborne particles and the visible light to confirm whether the particles are smoke. The sensor also fits the requirements for cost, voltage supply, and signal protocol.

Table 9: Flame Sensors

Name	Op. Temp.	Op. Voltage	Comm. Protocol / Output	Cost	Notes
ezPyroTM I2C Pyroelectric Infrared Flame Sensor (SMD)	-40 to + 85°C	2.7 - 8V	I2C	1 for $33.45	Thin film digital pyroelectric IR sensors. Full frequency range of flame flicker (3-30 Hz).
Thin Film Pyroelectric Flame Sensor	-40 to +85 °C	2.7 - 8V	Analog output	1 for $56	Noise at the signature 8-10 Hz flicker range of a flame Aperture: 5.2 mm x 4.2 mm A wide field of view of typically 100°
QFC Pyroelectric Infrared Flame Sensors, Analog	−40 to +85°C	2.7 - 8V	Analog output	1 for $73.76	In triple IR flame detection Noise characteristic at the signature 8 – 10 Hz flicker range of a flame. Used for forest protection. Wide field of view, typically 100°
KEMET's QFS pyroelectric flame sensors		1.75 – 3.6V	I2C	1 for $24.82	High dynamic range to ensure rapid and accurate detection of small and large

					flames, nearby or over larger distances. Full frequency range of flame flicker from 3 – 30 Hz. 90° field view
Analog's ADPD2140BCPZN-R7 photodiode	-40 to 85 °C	8V	I2C	1 for $2.47	Near Infrared Sensor: IR array primarily used to detect for infrared rays Spectral range from 800nm - 1080nm Compatible with the ADPD1080 photometric front end.
Adafruit AMG8833 8x8 Thermal Camera Sensor	Measuring temps of 0°C to 80°C	3V or 5V microcontroller or computer.		1 for $39.95	8x8 array of IR thermal sensors. 64 individual infrared temperature readings over I2C. Detect a human from a distance of up to 7 meters (23) feet.
Adafruit MLX90640 24x32 IR Thermal Camera Breakout - 110 Degree FoV	Measuring -40°C to 300°C	3V or 5V microcontroller	I2C	1 for $59.95	24x32 array of IR thermal sensors. 110°x70° field of view
Melexis Technology MLX90640 thermal camera	-40°C to 85°C	2.9V to 3.6V	I2C	1 for $39.95	32X24 IR array of pixels. 2 FOV options – 55°x35° and 110°x75°

Selected sensor:
The Pyreos EPY12241 pyroelectric sensor was chosen for flame detection. Some of the characteristics of this sensor include: Output sensitivity, Signal to noise ratio, Noise equivalent power, Specific sensitivity, and Response time. The user is able to adjust the low pass filter, high pass filter, sample rate, capacitance (gain), and trans-impendence. The sensor also fits the requirement for voltage supply, signal protocol, and cost.

4.3.4. Software Tools

Software tools play an important role in the development of a working system. This section discusses the different tools utilized to design, develop, or prepare the system. All of the software used for the purpose of development will be listed here like CAD programs, Administrative tools, chat applications, and software development tools.

4.3.5.1 CAD Tools

Contained in this section are some of the CAD tools used for this project. CAD tools were used for the mechanical design of the structure of the project as well as the PCB and schematic design for the electrical components of the project.

Fusion 360
Fusion 360 Student Edition was used to CAD and render the four preliminary mechanical designs for this project and the final design. Without mechanical designs for the project, the system will not function correctly. The mechanical design is almost as important as the electrical design for this project as the system must operate and exist outdoors with varying weather conditions and other hazards. Fusion 360 can design parts that might need to be 3D printed or machined.

KiCAD
KiCAD is used to make the schematics as it is open source, free, and provides all the tools needed to create any PCBs. Schematics are an important part to the development process as many bugs and errors are found at this stage and designed out of the system. Without schematic software, these problems may manifest into larger problems when the design is put to the test in real life. Furthermore, once money is spent on a faulty design it cannot be recovered. It is essential that designs be worked out before moving from the schematic stage. The program then allows the conversion from schematic to PCB Layout. The data can be sent to a manufacturing facility and they will manufacture the PCBs for the project. Lastly, the program allows the creation and modification of schematic symbols and footprints as well as the ability to import them from vendors or distributors that sell the products.

4.3.5.2 Administrative Tools

Contained in this section are some of the administrative tools used for this project. Administrative tools are any application that helps in the creation of documentation, communication, or organization including, but not limited to, file storage on a computer.

WhatsApp
WhatsApp was chosen as the tool for general communication. It is simple and does not have many integrations as some other chat applications, but it is lightweight and allows for chatting from a computer or smartphone. This means that the team can always communicate if necessary. It does not have any limits on file or image uploads as other chat applications may have. Furthermore, most of the team already had the application, so it was a quicker way to get started than learning or downloading a new application.

Microsoft Teams
Teams is another communication tool chosen to have meetings online. It provides conference calls, video calls, screen sharing, and file sharing to help organize and meet up in an efficient way. It is also accessible through phone and computer which allows the team to communicate easily at any time. The screen sharing feature lets the team communicate and share instructions and work effectively during meetings. It also provides messaging and filing system to keep work organized if needed. It can access calendar information by logging in using UCF knights' email. Since it is accessible by using knights' email, each member of the team has an account already created. Given the COVID-19 limitations, Microsoft teams was useful for team meetings and maintaining productivity despite social distancing.

Trello
Trello allows for task planning and scheduling so that the team knows what project component is due and when. This also allows each group member to schedule individual parts of the project, so the big picture is always in sight. The many integrations of Trello allow the team to do almost anything. Currently, Calendar integration is used so that it formats all the deliverable due dates into a calendar so that everything can be found quickly and easily. There is no confusion on when a deliverable is due.

Microsoft OneDrive
OneDrive is a good place for the team to easily share files together. The ease of OneDrive, compared to other tools like Google Drive, is that OneDrive will sync files from the cloud to any computer directly. This means files can be edited and uploaded/downloaded directly from the file explorer. No need for a browser or external tool. The expansiveness of OneDrive also allows synchronization to smart phones to view documents on the fly.

Microsoft Word
Since the team is relying on OneDrive, it became clear that using Microsoft office tools to work on documentation is a good choice. Word features a very comprehensive (but expansive) collaboration element so that work on a document can be accomplished simultaneously with all the power of offline editing. All the standard formatting tools exist, but in addition to that comments can be added, and collaborative chat exists with users currently editing a document.

4.3.5.3 Software Development Tools

Git
Git is the de-facto tool for version control across software projects. Git works by tracking changes byte-by-byte to files within a directory. This is useful when multiple users are editing a file at the same time. The way Git structures itself is by using "branches" which a user will "checkout" to. When checking out to a branch, the user creates a local copy of the files stored/tracked in the remote repository of code. The code that is changed locally

on the user's computer is not the same as the code that is in the remote repository. This is useful as the user can make any changes they want.

If another user wants to make changes to the same file, they clone those changes which allows them to work on the same file, unencumbered. When these two users complete their modifications, they will "commit" and "push" these changes to the remote repository, allowing their changes to become public and Git will automatically "merge" the changes into the current working branch. As long as there are no conflicts, the changes are accepted and saved in the remote repository. If there are conflicts (i.e. modifications to the same place in the file) then the users must manually accept and merge those changes that are correct. Git can be used to track binary files, but any change to the file usually results in a large change across each byte of the file thus causing the whole file to be updated.

Atmel Studio 7

Atmel Studio 7 is the IDE that is suggested to be used with the SAMR35. It includes a C and C++ compiler for the microcontroller, and so it was used for programming, debugging, and writing code for the SAMR35. Atmel Studio 7 features a programmer which is beneficial since the compiler, code editor, debugger, and programmer are all in one software package.

Python

Python is one of the most popular programming languages known for its ease of use. It has a simpler syntax and format compared to other languages such as C or Java. It is the main language used in recent computer vision applications and offers abundant libraries for implementing them into the project. Other well-known libraries such as OpenCV and Pytorch uses Python to implement computer vision. Most of the CNN (Convolutional Neural Network) models are trained and available in Python via GitHub. Tutorials and other guidance are available due to its popularity which will help in debugging and constructing the code. It is easy to learn which will save time and improve our system further.

C++

C and C++ are the standard languages used in embedded programming. As such, C++ was investigated to be used to program the SAMR35. C++ has a lot of features and syntax taken from the C language but allows for classes and data structures to be built and used from the standard library that C does not. This means that it could have been easier to maintain the software written, and it should be easier to implement.
C always followed the paradigm that nothing should be hidden and that it should have the simplest features so that the programmer is the one to implement all of the functionality. C++ Follows the paradigm of "C is a good foundation, but we can do better" and allows for a lot of expanded functionality that the C language does not provide. Other technologies like Rust were investigated, but C++ is a good mix between object-oriented software principles and embedded systems. In the end, however, C was used to program the SAMR35 since it was faster to get up and running. Due to issues with implementing LoRa, using C++ was not advisable as it took more time to learn and debug.

Atom
Atom is a simple text editor that provides syntax highlighting for different programming languages and formats. It also includes a markdown viewer. Some of the software and text documents were edited in Atom since Atom provides a lot of plugins for productivity.

Putty
Putty is a terminal emulator and serial console that allows for quick and easy connection to serial devices. When communicating with a device over UART, especially for debugging purposes, Putty became invaluable. Since it is a free and open source program, it was easy to get running and didn't cost a penny. Putty supports many different communication protocols other than serial communication like Raw, Telnet, Rlogin, and SSH. Using a Raspberry Pi, it was useful to use the serial console through the UART pins of the Raspberry Pi and the Raspberry Pi Zero W could use SSH to communicate with it via Wi-Fi, assuming we are using windows computers and may not have the ability to SSH through a terminal.

MobaXTerm
MobaXterm was used in addition to putty. It provides a nicer interface for serial and ssh connections and also allows users to save sessions so that information like baud rate or parity does not need to be entered in subsequent connections to the same device.

4.4. LoRa

This section covers the methodologies and lower level implementation of the LoRa modulation scheme.

4.4.1. LoRa Overview and Definition of IoT

A current "Buzz Word" all over the world is "IoT." IoT stands for Internet of Things. A "Thing" in IoT Is some kind of device that is able to sense information about the environment in which it is placed and transfer that data over a network. IoT devices share their data by connecting to an IoT gateway or other edge device where the data is sent to the cloud to be analyzed [41]. This connection together creates better understanding as the data that is collected can be interpreted in many different ways. The ways it is interpreted defines what kind of information someone can learn from that data.

LoRa, literally "**Lo**ng **Ra**nge", is a proprietary spread spectrum modulation scheme that is derivative of Chirp Spread Spectrum modulation (CSS) which trades data rate for sensitivity within a fixed channel bandwidth [42]. The idea is create a physical layer protocol that is separate from higher layer implementations which allow the protocol to be generically used with new and existing devices.
LoRa is bandwidth scalable, low power, and long range modulation technique. It allows a very large link budget that exceeds conventional FSK [42].

4.4.2. Quick Discussion of Common Modulation Techniques

Modulation is the act of changing a carrier signal to transmit information. A Modulator will turn digital data into an analog wireless waveform and a Demodulator will take the wireless waveform and convert it back to a digital signal. The goal is to convert this digital signal into something that can be sent wirelessly without interference to some other device amidst all the electromagnetic signals currently in the air.

This section quickly covers the three prominent modulation techniques. Modulation techniques as a whole are not limited to these three and may, in fact, incorporate multiple different schemes or modifications on these schemes to enhance different features of their wireless network. This section does not compare or contrast the different methods and does not explain the advantages of each, only the different methodologies as a whole to understand LoRa and how its modification on Chirp Spread Spectrum Modulation is relevant.

Amplitude Shift Keying
Amplitude Shift Keying (ASK) works on the principle that a digital 1 map to the presence of a signal at some amplitude while a digital 0 maps to the absence of that signal. A device can send a binary symbol by changing the order of presence to absence of this signal. A simple view of this technique is for every digital “1” that the device sends, it turns a signal on and for every digital “0” the device sends, it turns the signal off.

Frequency Shift Keying
A popular modulation technique. Similar to the above, Frequency Shift Keying (FSK) works on the principle that the two digital states are represented by a constant signal that varies in frequency. By changing between a high frequency signal to a lower frequency signal, the device can transmit a 0 or 1.

Phase Shift Keying
The device, in Phase Shift Keying (PSK), will alter the phase of a signal when trying to transmit information. For example, the signal might be at some frequency constantly, but if it is a positive signal it might mean a digital “0” but when changed to the negative waveform of that signal it means a digital “1”.

4.4.3. Chirp Spread Spectrum Modulation (CSS) & LoRa

LoRa uses a modified version of Chirp Spread Spectrum Modulation (CSS). Chirp Spread Spectrum was developed for radar applications in the 1940’s [42]. It has become more popular recently as it is low power and great sensitivity. Unlike other modulation techniques, it seems to have the inherent ability to resist multipath fading, Doppler effects, and interference in the same bands. The idea is that a “chirp” has a constant amplitude but the frequency passes through the entire bandwidth in a certain time. If the frequency increases it’s called an “up-chirp” and if the frequency changes from highest to lowest it is considered a “down-chirp” [43].

The alteration between up-chirps and down-chirps create the symbols for LoRa.

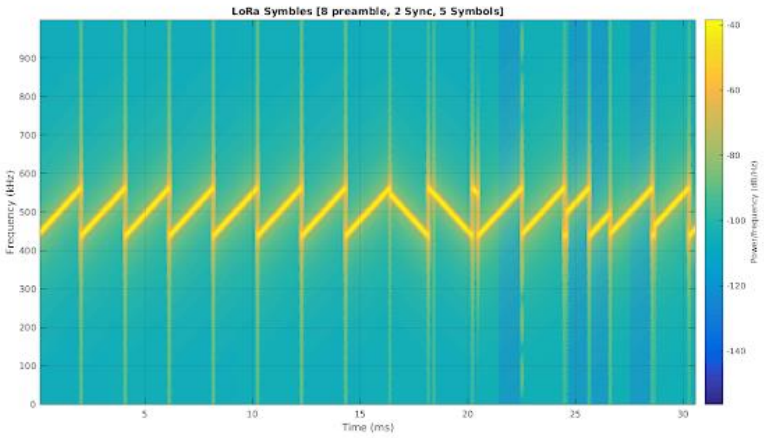


Figure 34: Spectrogram of LoRa physical layer [43]

The image above shows a LoRa frame on the physical layer. The frame consists of 8 preamble symbols, 2 synchronization symbols, the physical payload, and an optional CRC. The symbols are demodulated as 0's and 1's which cat be any kind of packet as defined by the project.

Lastly, an interesting feature of LoRa is the ability to change the Symbol Rate. By changing the "spreading factor" used in the LoRa implementation, the device can change the properties of the signal. LoRa uses three different bandwidths: 125kHz, 250kHz, and 500kHz. As a quick overview of all of this, incrementing the spreading factor by 1 roughly doubles the time to send the symbol. Therefore, a lower spreading factor results in a higher data rate and a higher spreading factor results in a longer transmission. Since there is this relationship, the Symbol Rate can be defined as this relationship here:

$$Symbol\ Rate = \frac{Bandwidth}{2^{Spreading\ Factor}}$$

This means the device should use a higher bandwidth and lower spreading factor to get the highest symbol rate. Doing so, however may affect the power consumption during transmission since time to transmit increases and/or different parts of the internal circuit may be active at different intervals.

5. Design

This section is a high-level overview of the fire detection system. In this section, there is an overview of the major function blocks, the use cases, and descriptions of the hardware and software sub-systems. The design should take components from all the previous sections as well as considering our design goals and motivation to create the final product.

5.1. Use Cases

The system, for all intents and purposes, acts autonomous but users still must interact with the system for the goal of the project to be successful. These uses are shown in the following sections.

5.1.1. Uses Case Diagram

Figure 35: below contains the use case for the fire detection system. There are three uses for the system: the Firewatch Official, the Installer, and a Networked Device. All three of these users will have to interact with the system.

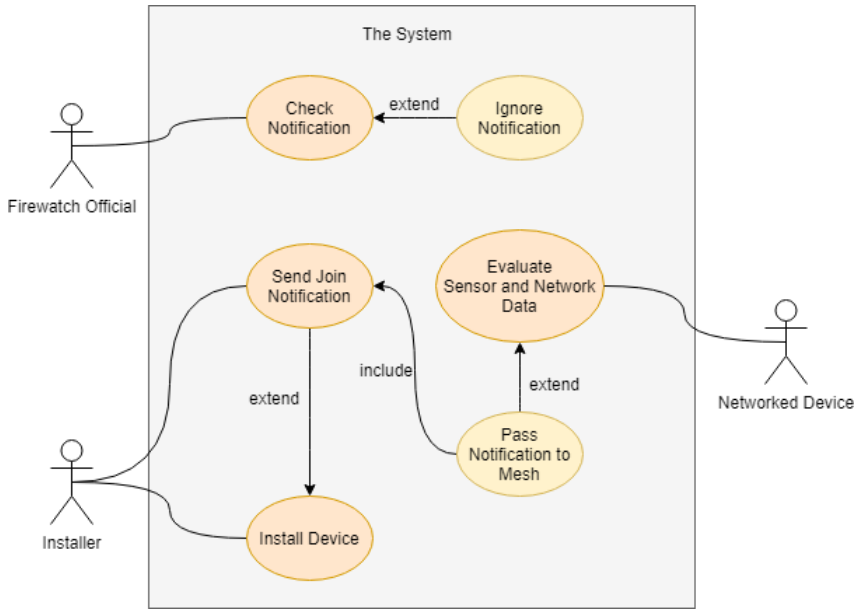


Figure 35: Use Case Diagram

5.1.2. Functional Design

The system is designed so that a fire watch official can check notifications from the system. These notifications will detail information about the mesh network and the individual devices connected to it. The fire-watch official can do no more than check the notifications and ignore them if he chooses. The Networked Devices and Installation personnel are the only users who may send notifications throughout the mesh network. The Installer connects a device to the network by putting the batteries in the device. At that moment, the device sends a join notification through the network. Before it becomes a “networked device” the node must join the network and the installer can reset the node until it joins. A Networked Device will evaluate the sensor and network data and choose to pass that notification to the mesh network if it meets certain criteria

5.2. Hardware Design

The hardware design refers to the electrical hardware that is present within the system. The hardware must work autonomously with very few failures at all times (day and night) to align with the project's design goals and motivations.

5.2.1. Hardware Block Diagram

The following diagram shows the hardware design sub-systems. There are four major subsystems. The top row of blocks shows the power sub-system. This subsystem is comprised of the solar panels, battery charging, and battery protection. This filters down into power regulation to create the specific power rails necessary to power the sensors, controllers, and the RF circuit. The three other subsystems comprise of the Sensor circuits, the Sensor control and processing, and the Network control and processing. These parts of the circuit are dominated by software instead of electrical considerations. If the serial communication circuits and power circuits are fine-tuned, these 3 subsystems will work well. The antenna and RF design must have care taken as RF antenna design must follow specific rules.
In the block diagram, there are some grey blocks that were not completed. This is due to difficulties with the LoRa technologies that will be explained in later sections. In short, the network and sensor processing controller was combined into one device (the Raspberry Pi) and the RF circuit was no longer used.

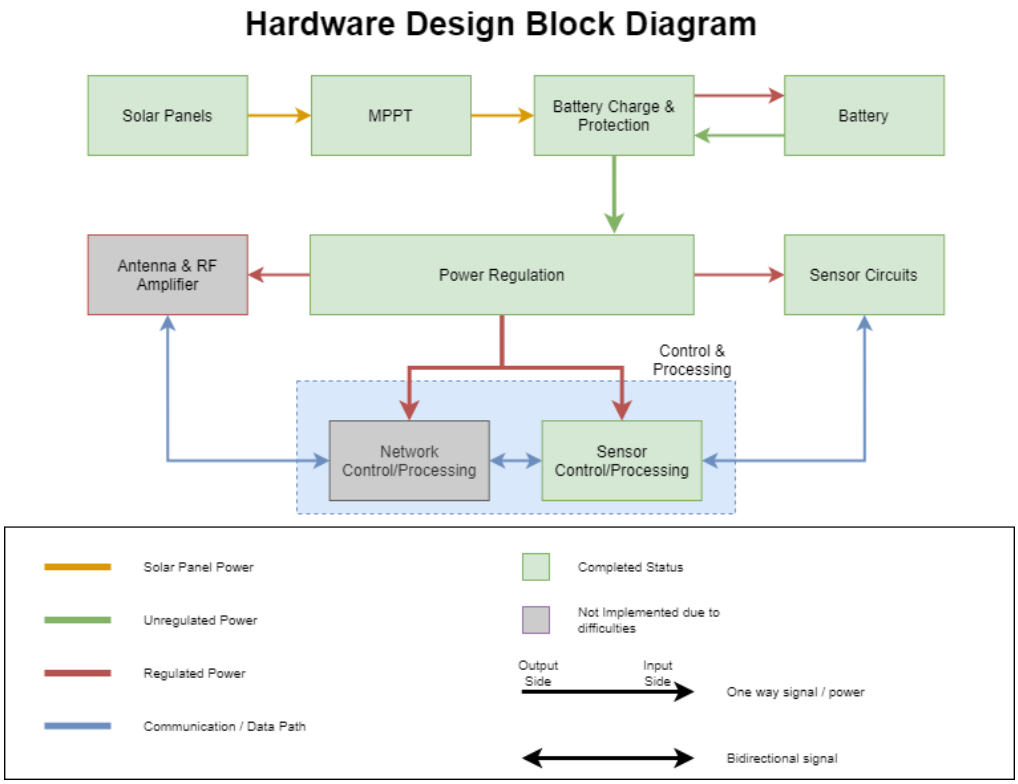


Figure 36: Hardware Design Block Diagram

5.2.2. Microcontroller and Processing Device

The system's original goal was to make use of two processing devices/controllers. The SAMR35 and the Raspberry Pi Zero. These two devices allow for much simpler interfacing and separation of responsibilities for the system. This also allows power consumption to be at a minimum during down time but the simplicity of programming for up time. The SAMR35's responsibilities include the Network control and processing. It should handle communications within the LoRa network and the connection to the network. The Raspberry Pi Zero conducts the sensor readings and return its confidence of how likely a fire is present in the local area. The root node must contain hardware to receive LoRa transmissions and software to process root node packets, but once it retrieves the data out of the network, any device can be used (such as another embedded system or a standalone computer). The SAMR35 internally uses the SX1276 Low Power Long Range Transceiver. This module incorporates an FSK modem and a LoRa modem. The device can operate in the 137MHz to 1020MHz range and is compliant with IEEE802.15.4g. Since this module is built in to the SAMR35, it does not have to be a standalone device.

In the end product, however, the SAMR35 presented some interesting design challenges that had to be overcome to produce a working prototype. The SAMR35 is a BGA Package which means without an x-ray device, the quality of the solder joints could not be confirmed. It is possible that some of the difficulties of implementing the design come from this. Another difficult part to implementing the SAMR35 is how new it is. There are a lot of resources for LoRaWAN, a MAC layer protocol built on top of LoRa. There are not many resources for just LoRa and the SAMR35. This meant that, to get a working product, a standalone SX127x chip was used to conduct LoRa transmissions. This chip was connected to the Raspberry Pi only, and the SAMR35 was not used for network transmissions.

5.2.3. Hardware Schematics

The following sections are descriptions and diagrams of the hardware schematics for the project. There are 4 sub-systems regarding hardware: Network, Raspberry Pi, Sensor, and Power. These 4 sub-systems must work together to do the final goal of detecting a fire.

5.2.3.1. Preliminary RF/Network Sub System Schematics

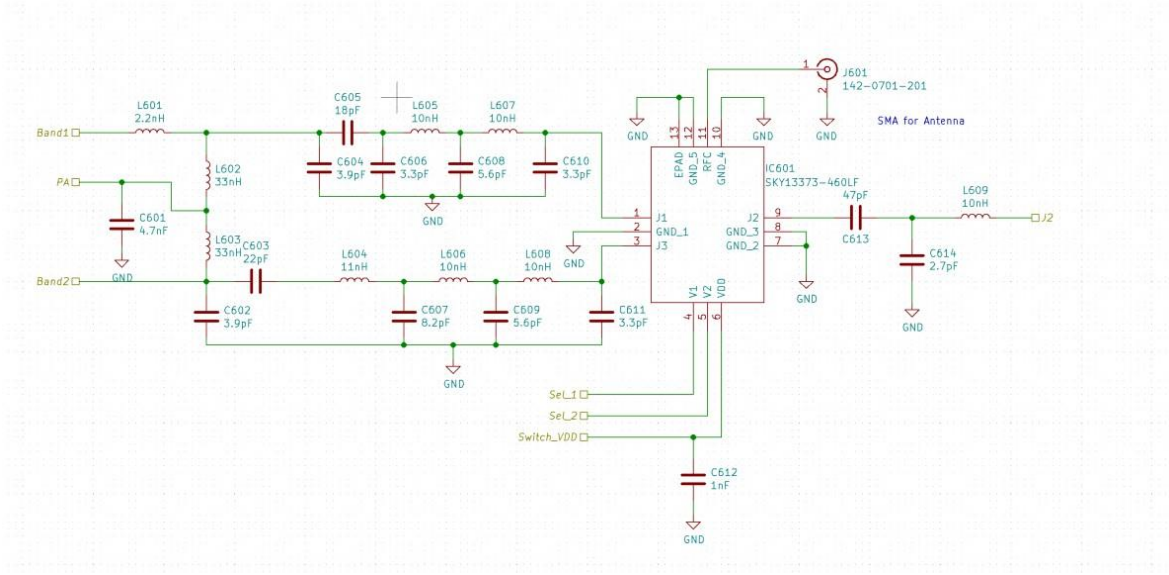


Figure 37: RF Switch Schematic

The schematic above in Figure 37: handles switching the RF signals so that the device can use a single antenna. In RX mode, the circuit is slightly different in the way it filters the signal than in TX mode. The device has two separate circuits on the TX side to select between different bands and power usages. The selection of the band is determined by the SAMR35. The schematic for the SAMR35 in Figure 38: Figure 37: is only the SAMR35 and its immediate connections. Most of the filtering hardware is removed for simplicity. Its UART ports are routed to the Raspberry Pi. Everything must be 50-Ohm impedance matched for the RF circuit. A Software defined status LED was given to the SAMR35 as well.

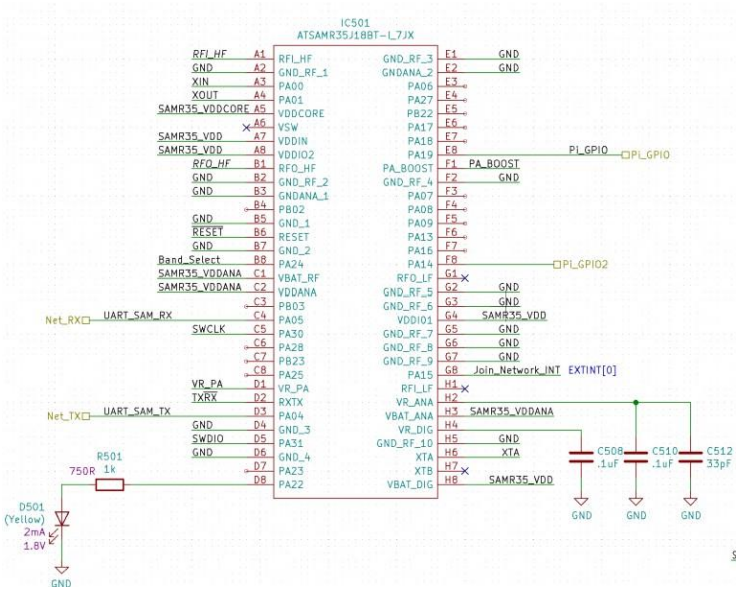


Figure 38: SAMR35 Schematic

Lastly, in Figure 39: there are some oscillators and buttons added for the SAMR35 to make it easier to use the device. The connector for programming and debugging was also necessary to include.

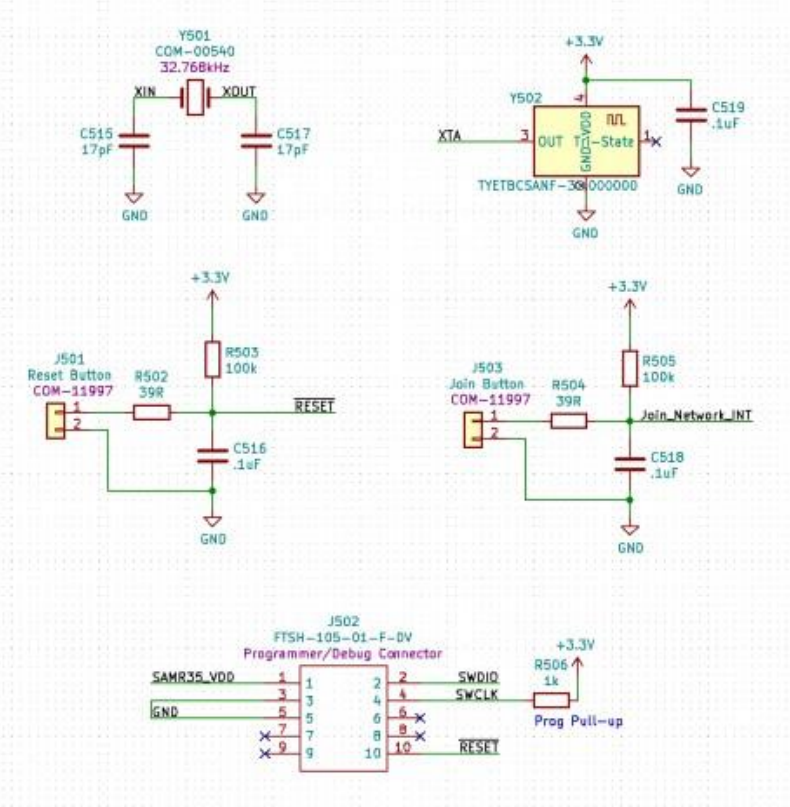


Figure 39: SAMR35 Peripheral Components

5.2.3.2. Preliminary Raspberry Pi System Schematics

The Raspberry Pi ended up doing a lot of the work. In the end prototype, the Raspberry Pi connected to the SAMR35 only through the TX/RX UART pins, however the SPI connections were used to talk to a standalone SX127x component to handle LoRa communication. An I2C bus allowed it to easily collect data from all the sensors. The connections are shown in Figure 40: . Ideally, the Raspberry pi is turned off most of the time and will be powered up only when it needs to do a sensor reading. For the final product the Raspberry Pi was powered ON during the entire operation of the device. The power consumption was more of an issue then but was not detrimental to the requirements. The Raspberry Pi also was given a software defined status LED.

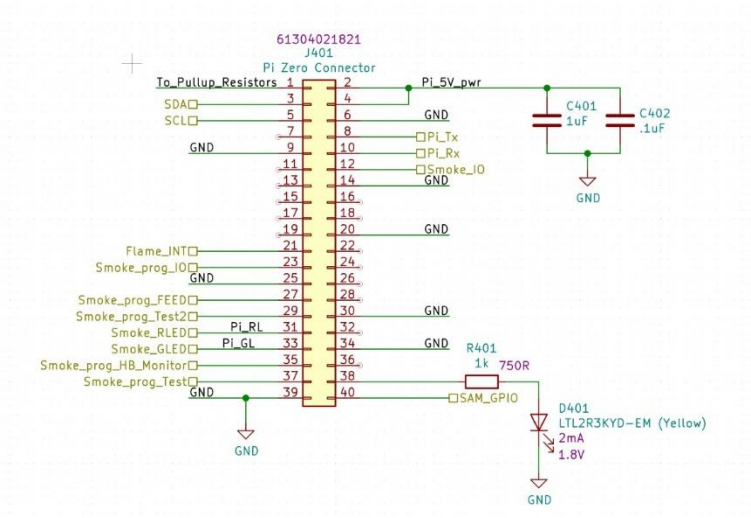


Figure 40: Raspberry Pi Connection Preliminary Schematic

Another important part of the Raspberry Pi circuit is how the Raspberry Pi turns on. The circuit is shown in Figure 41: . The Relay is connected to the SAMR35 and it is able to turn on and off the raspberry pi. In the final prototype, the SAMR35 immediately turns on the Raspberry Pi and does not turn it off. This is because the SAMR35 became redundant but required to turn on the Raspberry Pi unless the device had a pull-up wire soldered onto the PCB.

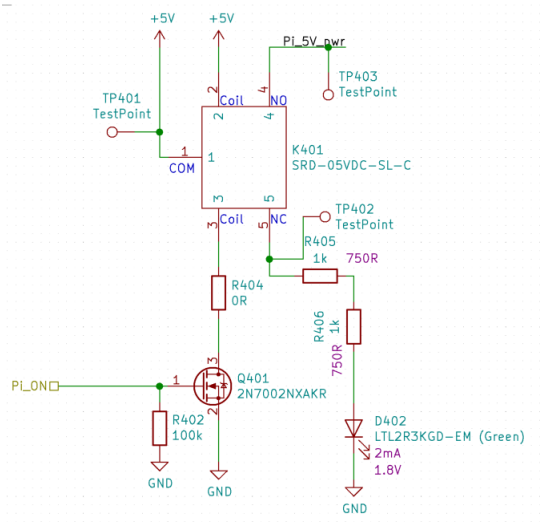


Figure 41: Raspberry Pi Relay Power Circuit

5.2.3.3. Preliminary Power System Schematics

The device is powered with a solar panel that is routed to an LT3652 IC. The 12v panels keep the charging IC constantly running at the most efficient state. To maintain a constant output the right solar panel is needed for the system.

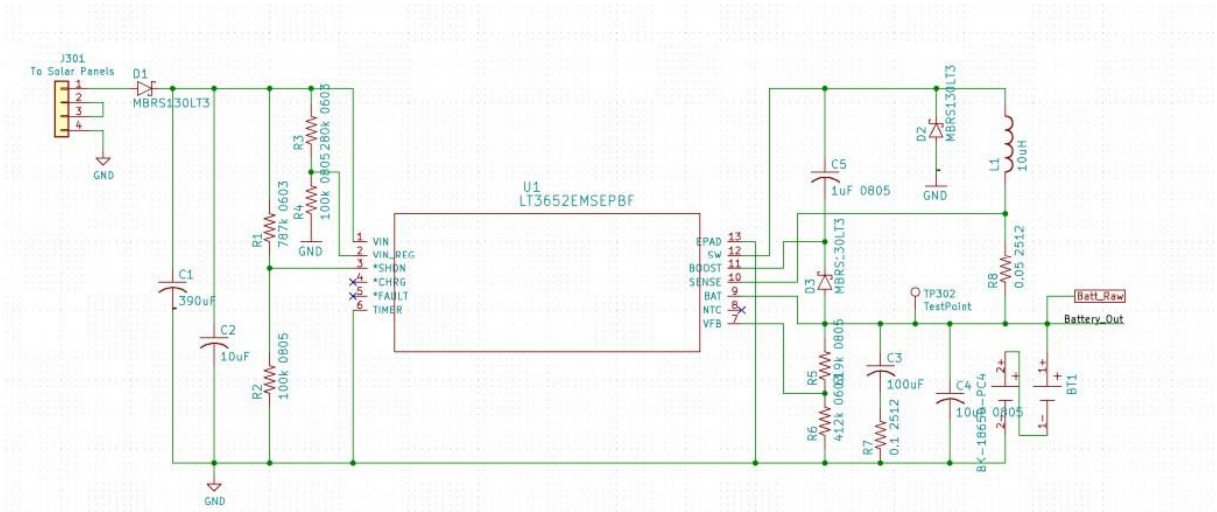


Figure 42: Voltage Regulator Schematic

The solar panel must output 12 volts nominally to supply the regulator and must have redundancies to help handle the changing solar radiation levels throughout the day. For this reason, two panels in parallel with many cells in series is needed to help with these issues. A panel similar in design to the one in Figure 43 is needed for this project. A panel like this has many cells linked up in series and then those sets of cells are then paralleled to prevent one cell getting covered or one cell breaking causing the entire panel to go bad and stop supplying power to the system.

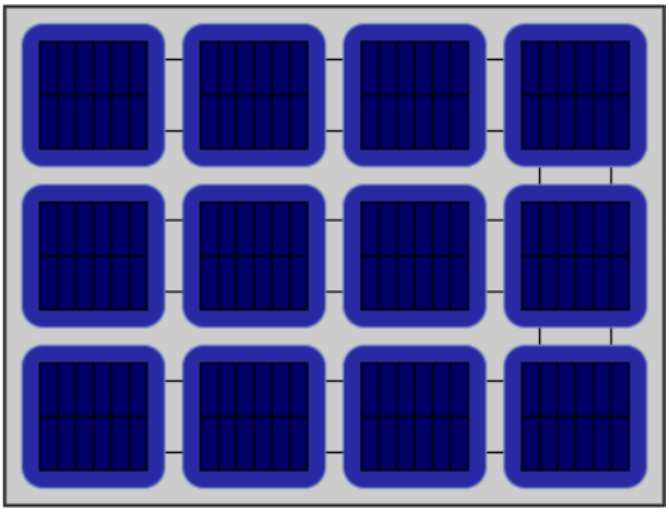


Figure 43: Top View Solar Panels

5.2.3.4. Sensor Circuit Schematics

This section provides an overview of the sensor schematics for gas, smoke, and flame detection.

Gas Sensor

The BME680 by BOSCH can detect ambient temperature, humidity, and barometric pressure and, most importantly, a range of gasses such volatile organic compounds. The sensor is also able to provide the air quality using an index provided below. This gas sensor can use both I2C and SPI communication protocols. However, for the schematic above was designed to select I2C.

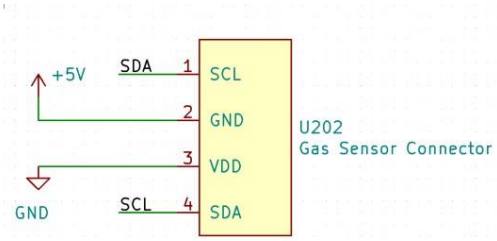


Figure 44: Gas Sensor Connector

The sensor can detect a range of b-VOCs such as Ethane, Isoprene, Ethanol, Acetone, and Carbon Monoxide. The output includes raw pressure, raw temperature, raw relative humidity, raw gas resistance, sensor-compensated temperature in Celsius, sensor-compensated relative humidity (%), sensor compensated gas resistance (Ohm), Index for Air Quality, CO2 equivalent in ppm, b-VOC (ppm), accuracy status of IAQ, gas percentage based on the individual sensor history, as well operational parameters such as stabilization time status and run in status.

IAQ Index	Air Quality	Impact (long-term exposure)	Suggested action
0 – 50	Excellent	Pure air; best for well-being	No measures needed
51 – 100	Good	No irritation or impact on well-being	No measures needed
101 – 150	Lightly polluted	Reduction of well-being possible	Ventilation suggested
151 – 200	Moderately polluted	More significant irritation possible	Increase ventilation with clean air
201 – 250[3]	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs	optimize ventilation
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present	Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible	Contamination needs to be identified; avoid presence in room and maximize ventilation

Figure 45: Air Quality Table [44]

Due to COVID-19, there were limitations on tools and resources and restricted access to the senior design laboratory, the Adafruit BME680 board was used to integrate the BME680 chip to the final PCB.

Smoke Sensor

Initially, the Microchip technology RE46C190S16TF was intended to be used for smoke detection. The design includes a photo amplifier to use with an infrared emitter/detector in pin 3 (Detect). The internal oscillator allows for smoke detection to occur for 100us every 8.1 seconds; this helps to minimize standby current. When smoke is sensed, the detection rate is increased for verification purposes. Every 32 seconds, the device checks for low battery and chamber integrity. The smoke chamber is located between pin 3 and

6 and is illustrated below. The internal comparator compares the photo-amp's output to an internal reference voltage. When the smoke conditions are met, the device triggers the local alarm. Below is the schematic of the sensor.

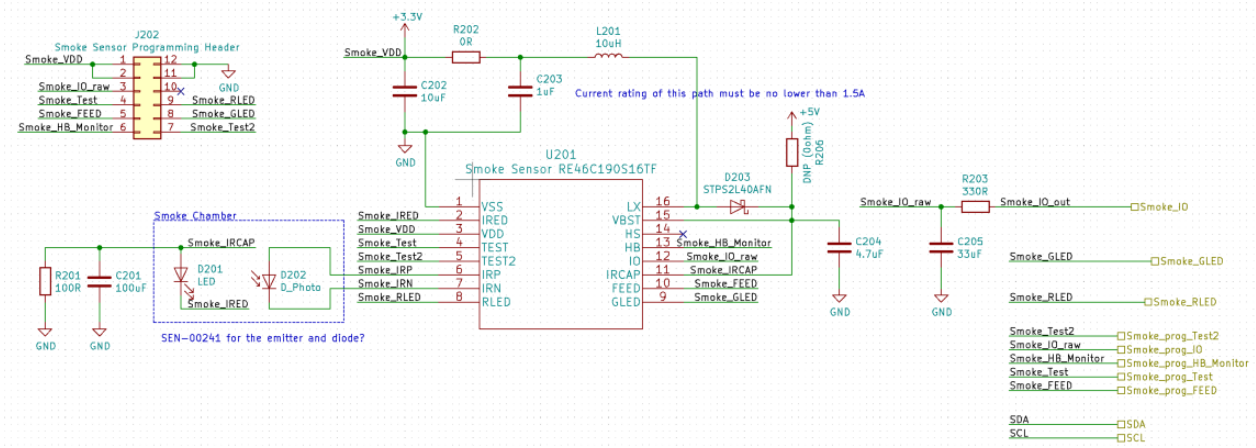


Figure 46: Smoke Sensor Schematic

Due to COVID-19, there were limitations on tools and resources and restricted access to the senior design laboratory, the PIM-438 board was used for smoke detection instead. This sensor also used the photoelectric principle to detect the presence of smoke. The algorithm uses infrared light (invisible to the human eye) to check for the presence of airborne particles and the visible light to confirm whether the particles are smoke. The sensor also works in high ambient light, complete darkness, or artificial light. This sensor was soldered onto the 3.3V, SCL, SDA, and GND pins of the PCB.

Flame Sensor

The flame sensor that was initially intended to use was the EPY12241 SMD chip. However due to COVID-19 there were limitations on tools and resources and restricted access to the senior design laboratory. Instead the breakout board for this chip was used instead. There are five key characteristics to consider when using pyroelectric infrared detector: output sensitivity that depends on narrow infrared band, signal to noise ratio, noise equivalent power (NEP), specific detectivity (D*), and response time [45]. The output sensitivity, D*, SNR, and NEP are dictated by the manufacturer in the module design. The user, however, has the ability to choose the electrical response time by adjusting the impendence and capacitance as well as setting the high pass and low pas filter values and the sample rate of the filter. [46] This sensor will be useful in identifying flames in low light, such as at the nighttime. To use in the daytime, it must be paired with PYREOS' sunlight-rejection sensor.

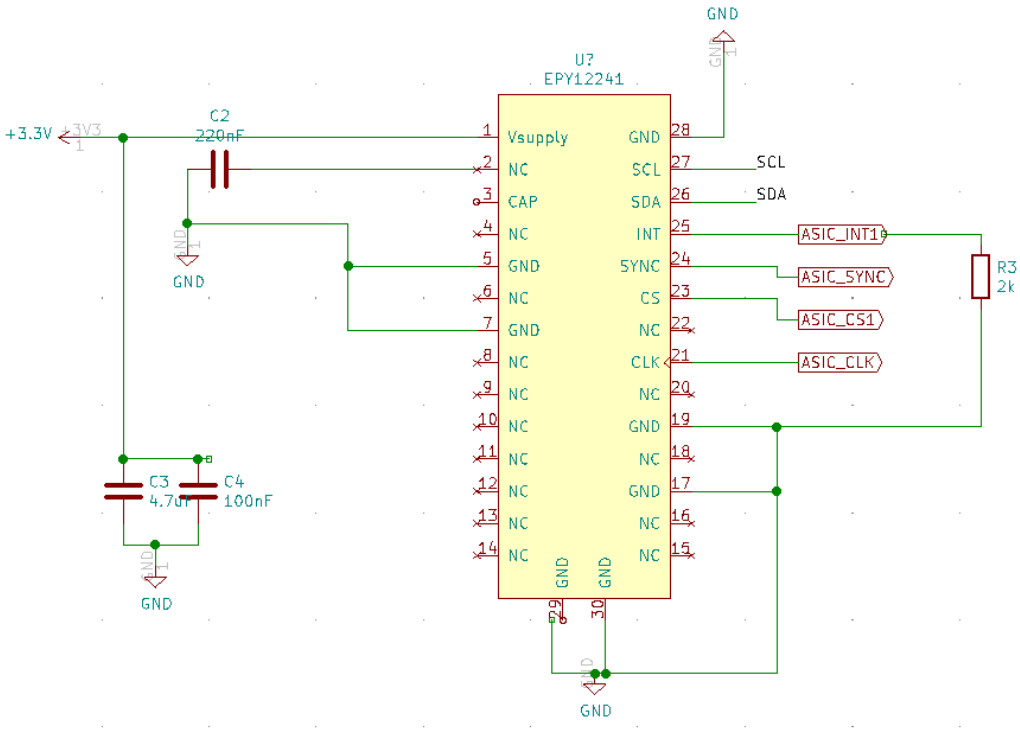


Figure 47: EPY12241 SMD chip

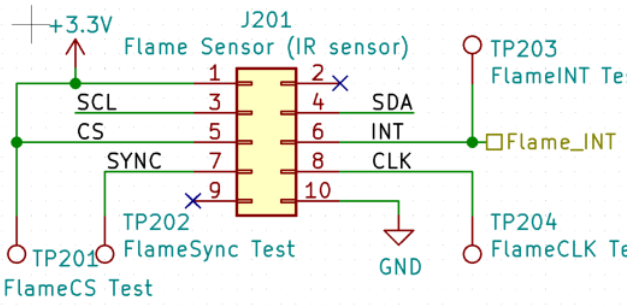


Figure 48: Flame Sensor Schematic

5.2.4. Mechanical Design

The following mechanical designs are potential ideas on how the system will be mounted. The designs each have 2 components: The mounting apparatus and the system functional area. The functional area, for these preliminary designs is represented by a 3D cube. The estimated size of this area is a volume of only 15 by 15 by 15 centimeters. The 3D cube is meant as a guide to see the area in which we expect the structure of the system to occupy, it is not necessarily to scale. The system was designed in such a way that it is mounted to a tree or other tall structure. The first design in **Figure 49** simply gets

mounted to the side of the structure. It has 6 screw holes to allow for mounting. And is the simplest and quickest to install. This works great for a small system and large trees as the tree's trunk will appear as a flat surface at small scales.

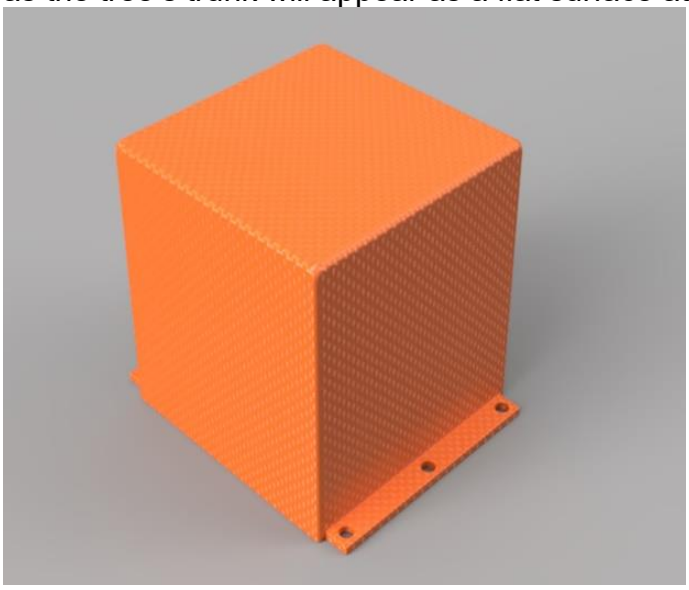


Figure 49: Mechanical Design A

The second design, shown in **Figure 50**, has a large loop that gets wrapped around something that would fasten the system to a tree. This could be the trunk of the tree or a branch. It is simple in design and would be easy to install as it just requires one point to lock the mechanism to the surface. A downside to this design would present itself for trees or branches with a large radius as the band would have to be large enough to support the device. This design is scalable with almost any size of the system, big or small.

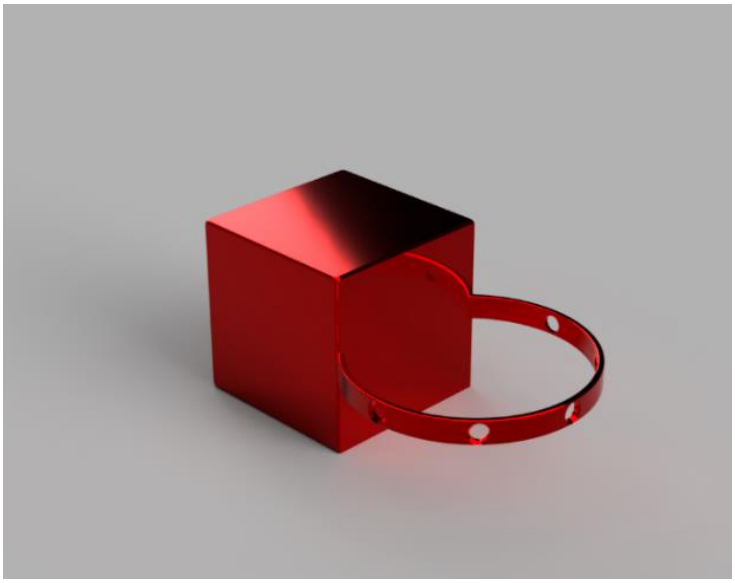


Figure 50: Mechanical Design B

The design idea represented in **Figure 51** would get clamped around a tree branch high in the tree. It is not designed to be clamped around the trunk of a tree. This design would be attached by clamping the bottom part to a branch like a claw. A difficulty of installation might appear when trying to install the system in a high/tall structure and mounting it vertically. Due to the longer arm on the design, it creates a stronger moment of inertia and could prove difficult to implement with a heavy device.

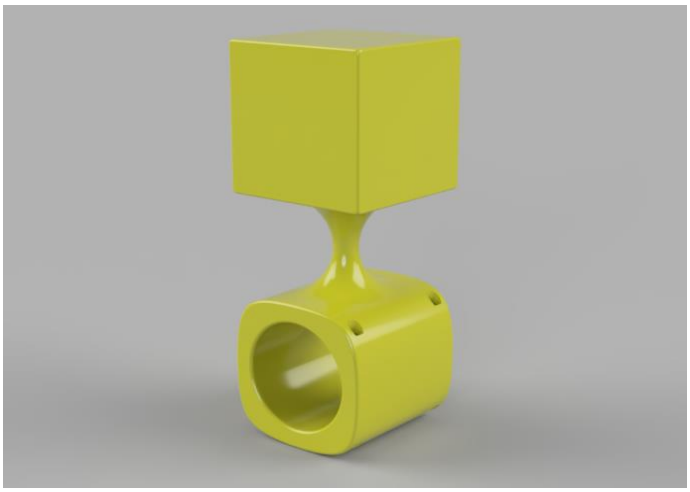


Figure 51: Mechanical Design C

Finally, the last design shown in **Figure 52** is hung on a tree branch or other horizontal structure. This is a great design for simplicity. It allows the installer to simply hang the device wherever it needs to be. For quick/temporary deployments it might be the best solution. For long term deployments, this solution may need some form of bracket or screws to be inserted to lock the device to its structure so uncontrolled scenarios like weather or animals cannot move or knock the device off its structure.

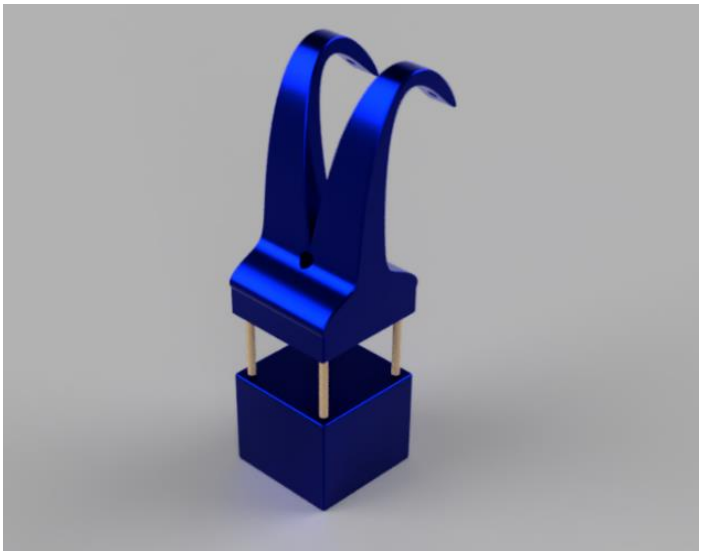


Figure 52: Mechanical Design D

5.2.4.1. Final Design

The final design, shown below, is one that was optimized for 3D printing for the purpose of rapid prototyping. It took 26 hours to print this design in its entirety, but it was perfect for prototyping as it proved that the design had plenty of room for cable management and showed some flaws that wouldn't have been caught otherwise. Flaws such as low airflow in the container when assembled which could be remedied by a simple one inch by one inch five-volt PC fan. The design could also be optimized by having a 'tunnel' through one side of the box which the gas sensor and smoke sensor could be mounted in and a push-pull fan configuration could be mounted to each end of this tunnel to pull fresh air across these two sensors for better readings.

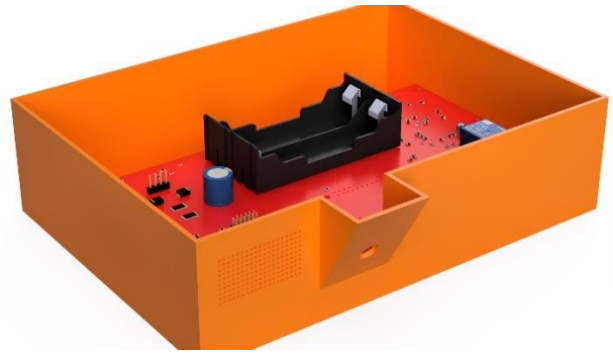


Figure 53: Final Design (open)

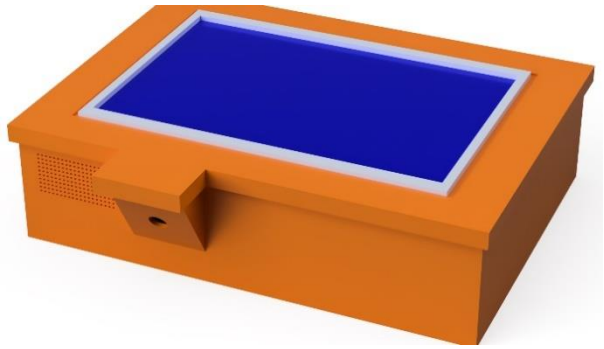


Figure 54: Final Design (closed)

5.3. Software Design

Similar to the hardware design, the software design must be as reliable as possible and have minimal failures. When a failure occurs, it must also be able to correct for those failures or allow for itself to be ignored or disabled until a time when it can be replaced. The following sections discuss the software designs and the methodology in place to let the controllers do their job.

***Note**: In Section 5.3* Software Design*, all software functionality designed for the Network controller has been moved to the Raspberry Pi and the SX127x chip used for LoRa. Due to implementation difficulties the Network software was never implemented on the SAMR35, only the Raspberry Pi. The sections below may reference the "network controller" or SAMR35 and that is to be understood to have become the Raspberry Pi for the prototype implemented in Senior Design 2.*

5.3.1. Design Methodology

The original design methodology implemented for this project is to keep every function compartmentalized to its own controller. There are two controllers: Sensors/Fire detection and Network. The Network controller's job is to join and manage its connection to the

network while the sensor controller's job is only to read all the sensors and determine if there is a fire.

This idea of keeping the functionality partitioned among the hardware allows for simpler software to be written and for the system to use the least amount of power possible during idle. The only interaction between these two controllers is the sensors controller sending a notification to the network controller that there is a fire, and possibly a small message string to send along with it. It is possible that other binary data is sent (such as raw data) and so the two systems will need to be able to communicate simply and effectively (such as through the SPI protocol or UART protocol).

Network Controller:
The network controller is responsible for making a mapping between itself and the other controllers already in the network. When joining a network, it will beacon a join request and the controllers in its vicinity will respond with linking information. This is to ensure that the controllers can be in the same network and to avoid duplicate packets being sent. The controller will then continuously listen for packets of data and will absorb packets until a timeout, or the sending controller decides it is done sending.

At this point, the network controller will send packets to all in its network map (except for the sender) to attempt to get the data back to the root controller. If it hears any repeated packets from another controller, then it will discard them. To arbitrate between a busy network, random delays will be introduced to avoid controllers from responding to other controllers whose messages are already being sent.

A second function of the network controller is to wake up the Sensor Controller, which should be shutdown at all times. The Sensor Controller will be woken up in 2 different cases: A fire has been detected or a timeout has been triggered. This is to avoid unnecessary power consumption. The Network Controller may or may not send a notification through the network that the Sensor Controller is turned on or off at any time.

Sensor Controller:
The sensor controller is to stay asleep/shutdown when not reading sensor data or processing the sensor data. When the Sensor Controller is complete processing its data it will send a shutdown notice to the Network Controller so it may configure its timers or send notifications to the network. The Sensor Controller will read all available sensors on its communication busses and read data from a camera if applicable. Using this data, it will decide if there is a fire in its vicinity or if there is not a fire. It will report this decision to the Network Controller and prepare to shut down.

Final Design Changes:

In the final design, the network controller became the Raspberry Pi and the Network software and Sensor Software were run on the same system. To communicate between the two, a known file was written to or read from between the two processes.

5.3.2. Software Block Diagram

The software block diagram shown in Figure 55: is the basic design that we are following for the full software package of the system. This diagram does not differentiate between the network controller or the sensor controller, so it appears as one conclusive system. This is how the final prototype was implemented. In the original design, the “Main Loop” and the Network side of the diagram is managed by the SAMR35 and the Sensor Data side of the diagram is managed by the Raspberry Pi. The data path between the two systems will be worked out as an “on chip” communication bus between the two systems. In the final design, the data bath between the two “systems”/processes became a known file in the Linux file system.

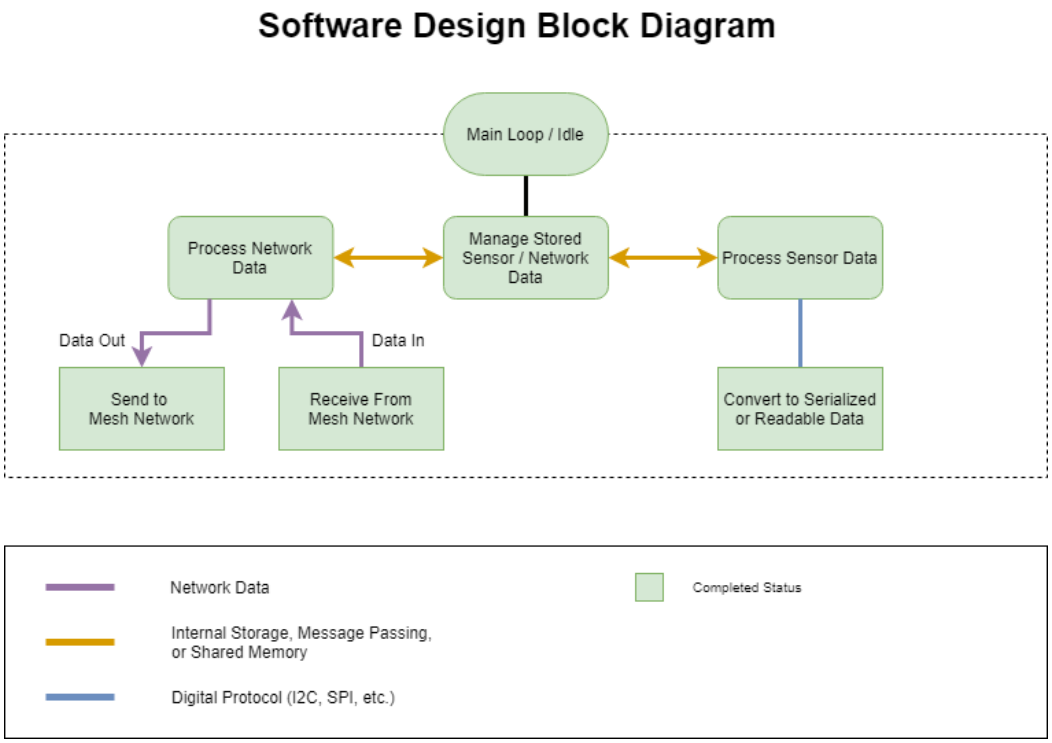


Figure 55: Software Design Block Diagram

5.3.3. Network Software

The network software can be complex and so many diagrams were made to keep track of how it all works. The following sections within Section 5.3.3 discusses the Network Software specifically.

5.3.3.1. Network Flow

This section describes the “Network Flow” of the system. The network, in normal circumstances, is not busy. Most of the traffic exists when broadcasting a message to or from the root node as it must propagate through the network to its destination. Figure 56: describes a Join Request case.

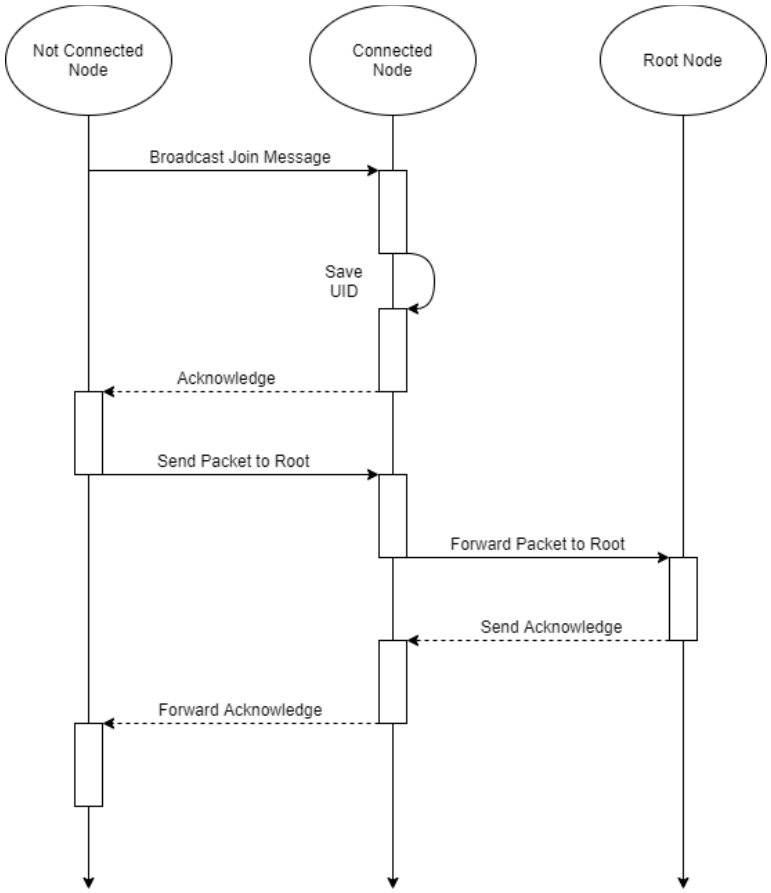


Figure 56: Join Request flow diagram

When a Join Request is received the currently “Not Connected Node” broadcasts a Join Message and all connected nodes in the vicinity respond with an acknowledgement. This acknowledgement is important so the previously disconnected node knows that another node can hear the message. After some time, the node sends a join message to the root node through the network to request for an acknowledgement from the root. As far as all the connected nodes are aware, this new node is sending messages through the network but has not “joined” the network. Once the “not connected” node hears the acknowledgement from the root node, it will consider itself “Joined” to the network and attempts to forward packets through the system like any other node. Furthermore, this acknowledgement contains the time as Unix time so the node can update its real time clock. A “Not Connected Node” can send messages through the network but will not forward messages through it. Acknowledgements are generally not required in the scheme, but without them, there is no way for the node to know that it is correctly connected to the network. This implies that a “not connected” node can still broadcast a fire condition but will not send packets through the network from other nodes.

Other kinds of messages follow similar protocols as these two messages in some way. Described in Section 5.3.4, a generic case is shown where a generic message is received. Figure 57: , below, describes a “Fire Packet” which follows a similar protocol as the generic case, but with some extra work.

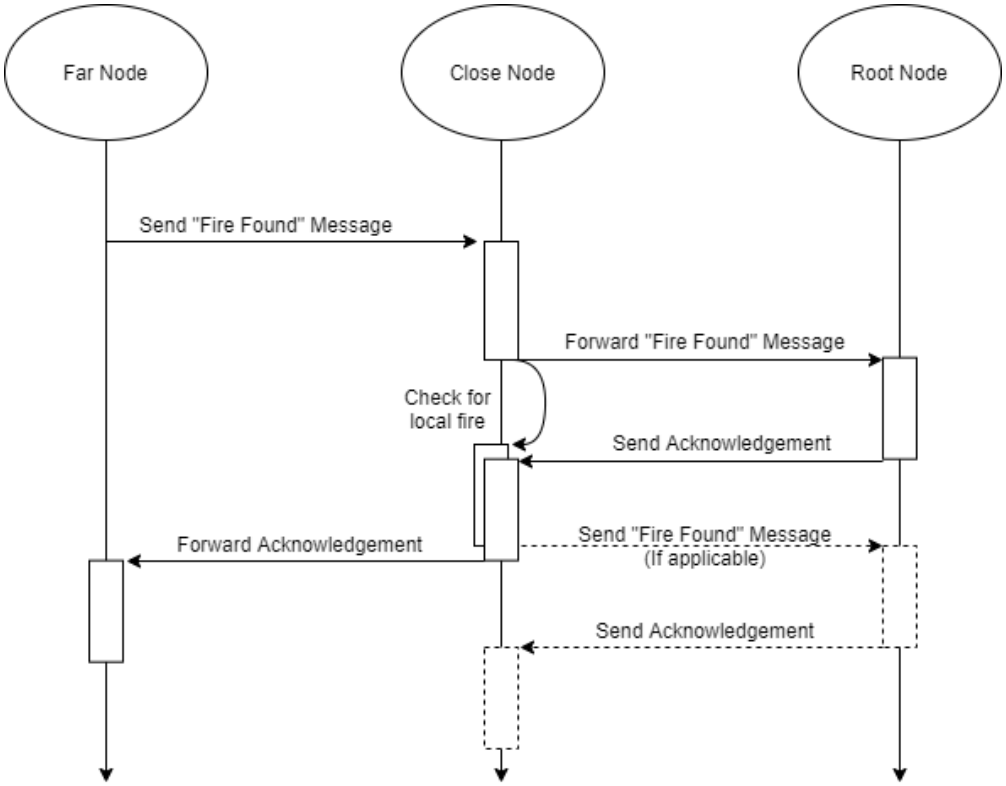


Figure 57: Fire Packet flow diagram

When a fire is detected at a “far node”, that node sends a message to the root alerting of the fire to the nodes closest to it in a broadcast. These “close nodes” forward the fire found message to the root node. For every valid fire packet received, the node wakes up its sensors and begins looking for a fire to report. During this time, it will continue to normally forward packets as necessary. If the “close nodes” detect fires in their area as well, they send the “fire found” message to the root node too. To stop the network from getting busy, the sending nodes of a fire message will stop sending packets after some time. Eventually, all the nodes will stop reporting the fire continuously and will wait some amount of time before reporting the fire again.

So far, all messages have been treated as asynchronous and can send at any time. In the event of a busy network or hot spot (very many nodes in a small area) then some kind of network arbitration is necessary.
First, it is important to understand that each node maintains a packet buffer to store received messages before sending. The node holds onto these packets until it is time to forward them into the network. This buffer only contains packets that need to be transmitted, not packets that are invalid or are internally processed. This buffer needs to

be sufficiently sized to handle a busy network with many forwarded packets. To arbitrate who may send, nodes that receive packets that require transmitting a message (such as forwarding the message) wait a random amount of time before sending their message. Each node will assume that after this amount of time, they may send a packet. Receiving a message during this delay will increase the timer by some factor to ensure that the waiting node does not interrupt a current transaction. Ideally, each node maintains its own state such that it can forward messages without losing state while waiting for an acknowledgement or response. This scheme alone appears like it may work but runs the risk of packets never forwarding through the network if the network is busy. The packets will eventually be forwarded since the network will eventually go silent and packets will slowly trickle through the system until they all go through. In future implementations packets going through the system should be assigned some priority (on a first come first serve bases) that is supplemented by the type of packet that comes through (for example a “Fire Packet” might have higher priority than a “Join Request” Packet). Higher priority messages are sent first before lower priority messages. Lastly, messages that sit in the buffer longer than other messages should accrue a higher priority than their initial priority. This ensures that packets, eventually, get through the system. Some packets may not gain a higher priority past some maximum. This allows for some packets to always have a higher priority overall to other packets (for example a “Join Request” might always have a higher overall priority than binary data).

The last bit of arbitration is to ignore repeated and invalid packets. Since it is a mesh network with different nodes receiving different messages, packets that are repeat packets are to be ignored by the receiving node. In this case, if multiple nodes can hear each other, they do not send packets in a cyclic pattern and then get stuck in a loop of transmissions. If a packet is received from the same origin multiple times it will be considered invalid and will be ignored. This invalid state persists for some time to avoid packets getting through cyclically. After enough time has passed for that packet, the state will no longer be considered invalid and a repeated packet can get through. Packets that come from the same sender will be allowed to pass through multiple times to allow for valid repeat transmissions. This means that before forwarding a packet, the node must check the origin of the packet and the sender. If the sender and origin are the same, or the origin has not been heard from before, then the packet is valid. If the packet has come from that origin before and the sender is different than the first sender, then the packet is ignored and considered invalid. This methodology, in theory, creates multiple paths from the origin point to the root where repeated packets are also sent along this path. The quickest path to the root node is the path that will prevail in transmitting the message along that path. In future implementations, nodes should remember where packets successfully arrive from and acknowledgements should get forwarded along this path. It is a stretch goal that the original join request and subsequent heartbeat packets determine this ideal path and set it up as the primary path that messages get sent along. This may avoid issues with clogging the network and may act as a form of load balancing over time. The system does not support load balancing of this form as of the final prototype implementation.

5.3.3.2. Network State Machine

This section discusses the different states that the network controller has at the time of the final implementation. To make software easier to implement, a state machine will be used for the embedded software design. This state machine will allow the network controller to understand its state and environment and keep the code compartmentalized and simpler to maintain. The state diagram below shows the transitions between states.

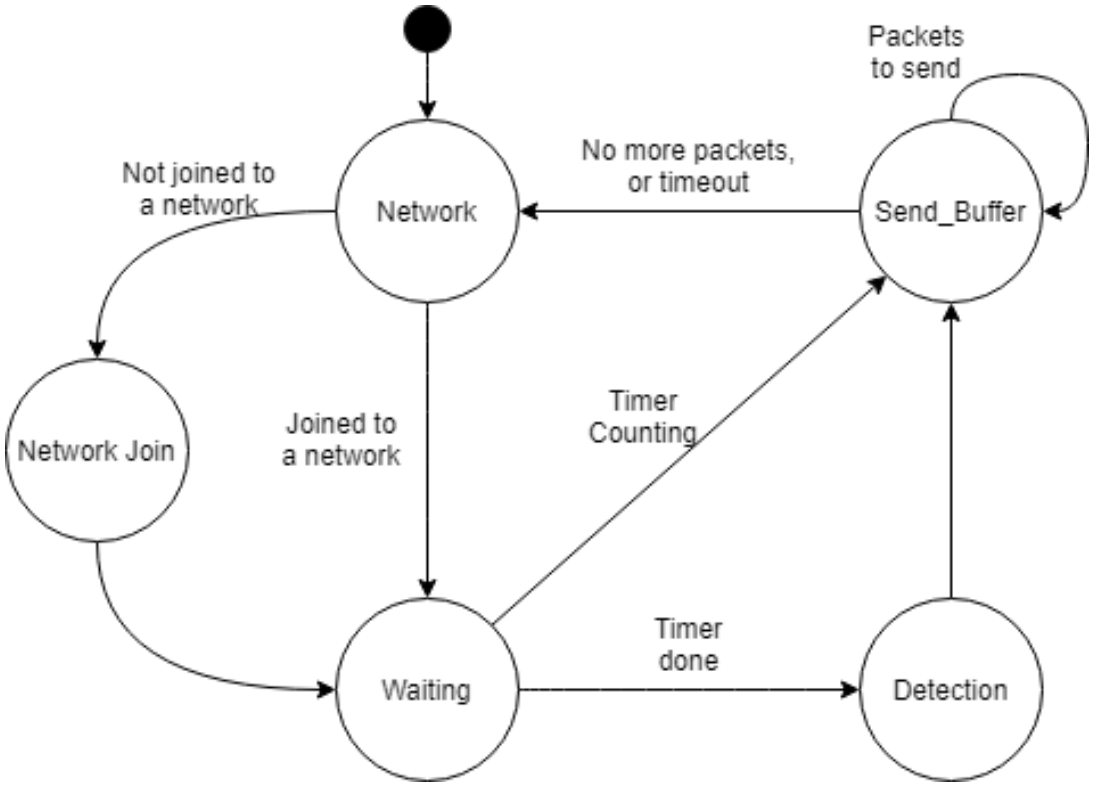


Figure 58: Network Control State Diagram

The state transitions are not too complex. Structuring the software in this way allows each individual state to be simple code compared to complex code as well.
The initial state is the Network state. This state handles storing all network data to non-volatile memory if it changes and to determine if the device is in a valid network. "Heartbeat" packets may be sent in this state as well. Notice that there is no state for receiving packets on the network. This is because receiving packets will be serviced in an interrupt. After the Network state, the system transitions to "Waiting" which resets the timer to its delay time and waits before moving to the "Detection" state.
From here, if there are packets to send then the system will do so in the "Send_Buffer" state. After the timer is finished, the "Detection" state is invoked. During this state, nothing happens. A loop will wait until the timer is done running to move on. This state waits for the Raspberry Pi to finish reading the sensors and to acknowledge that it has detected a fire. The system will wait for the Raspberry Pi to finish for an indefinite amount of time. After the response, the state transitions to "Send_Buffer" again to offload any forwarded packets and fire packets.

State	Next State	Transition Condition	Previous State	Description
Network	Waiting	A network has been joined	Initialization or Send_Buffer	Saves Network State and other tasks
	Network Join	No network has been joined yet		
Network_Join	Waiting		Network	Joins the Network
Waiting	Send_Buffer	Timer is not done	Network or Network Join	Waits for timer to finish.
	Detection	Timer is done		
Send_Buffer	Network	No more packets or Timeout	Waiting or Detection	Send packets from the packet buffer
	Send_Buffer	Packets to send		
Detection	Send_Buffer		Waiting	Waits for the raspberry pi to detect a fire

Table 10: State Transitions

5.3.4. Software Events & Flow

This section discusses the software events and flow that happens in the system. It includes charts and descriptions of both parts of the software system: The SAMR35 and Raspberry Pi.

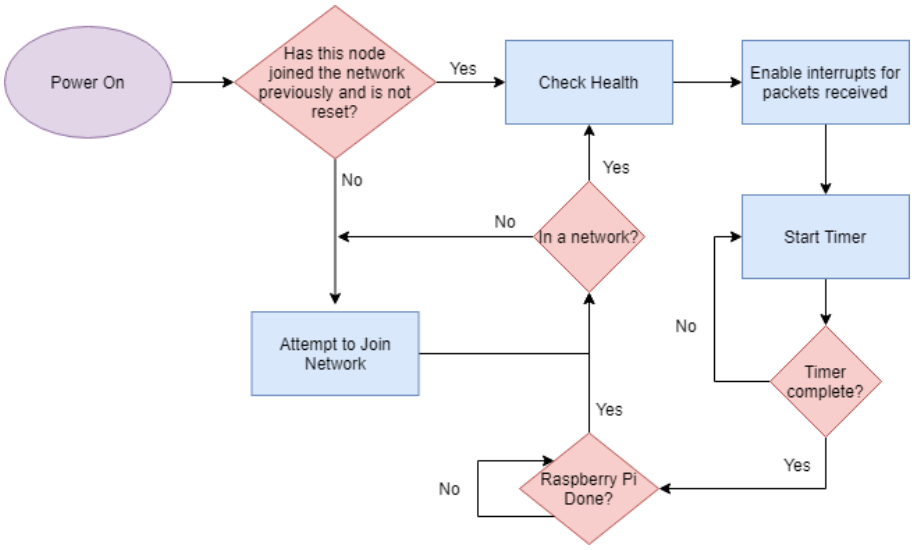


Figure 59: General software flow when power is applied to the system

The image above is the process the system takes when power is applied to the system. Once the SAMR35 is ready to begin running instructions, it begins this process. First it must check if it has joined a network already. If it has joined a network already, then it skips the join process. Skipping the join processes is critical to not get stuck in the edge case where it can be heard by the network but cannot receive messages. In this case, it can still send messages even though it is not “joined” to the network. The system sends a join request to the network. During this time the node takes a randomly generated UID and will broadcast this UID to the network. An acknowledgement is expected from at least one network node which will confirm the UID and allow the node to attempt to send a packet to the root node. If an acknowledgement is received and claims that a UID is invalid, the system will select a new UID and try again. This is rare and shouldn’t happen. Once a valid acknowledgement is received, the node waits to hear the acknowledgement from adjacent nodes and then transmits a packet to be forwarded to the root node. As mentioned in the last section, it is a stretch goal at this point that the node set up some kind of path memory so that the network can decide on some kind of load balancing mechanic. Once the root node acknowledges this new node, it will be considered “joined” to the network and can begin forwarding messages.

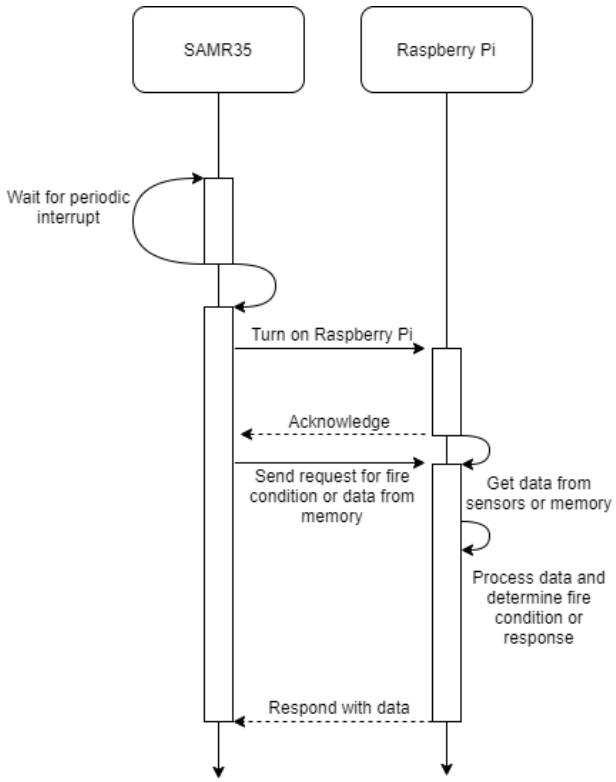


Figure 60: Raspberry Pi Flow

Once the system has been turned on and joined to a network, it begins a timer. This timer will be set to wait for the Raspberry Pi. Other timers may be created to monitor health of the system as well as sending network packets to the network. The most important timer, however, would be the Raspberry Pi timer. When the timer ends the network software will read a file written to by the sensor software. The sensor software after a timer will begin reading its sensors and determine if there is a fire. The Raspberry Pi saves data for the network software and the network software will decide whether or not to transmit the data. This may be control packets or requests for data. The data will be accumulated by the Network Software from the Raspberry Pi. The Sensor software determines if it is done processing or not and then will alert the Network software that it is finished. When it is finished, the network software will continue on.

Another important aspect of the system is the actions to be taken when a message is received. There are many actions that could be taken based on many different parts of the packet that is sent to the node. All in all, there are 4 major conditions to check before deciding what to do with a packet. The first action is checking the CRC. If the CRC is bad, the packet is rejected. A stretch goal may be to send back a negative acknowledgement so that the packet can be re-transmitted. Otherwise the packet is rejected, and no action is taken. The next conditions that matter are pertinent if the message's destination is the root, the current node, or if it is a join request. If it is neither of these things the packet is rejected. This is so the network is not clogged by forwarded packets that are unnecessary. From the perspective of the node, all other non-adjacent nodes are hidden. If it receives a message for one of those nodes, it is considered an invalid packet and ignores it. No

node can transmit to a node that is abstracted by one or more layer. Any node always knows of the root node. There is, however, a broadcast ID that a node can choose to use that all other nodes respond to.

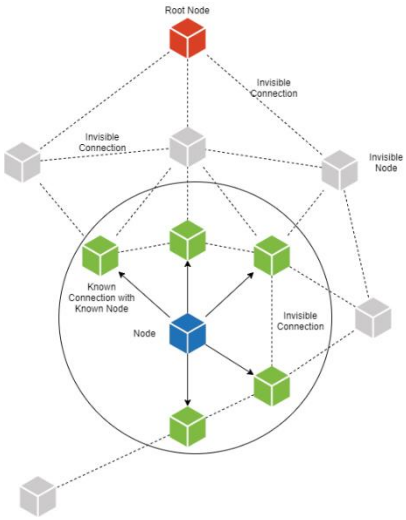


Figure 61: Known Connections Diagram – Mesh

If the packet is a packet that must be dealt with by the receiving node, then the node will determine what kind of reaction is necessary. Sometimes an acknowledgement may be necessary and at this time the node will broadcast that acknowledgement. Special consideration is taken for a fire message as this message requires the system to wake up its sensors and check for a fire in the local area. See the figure below for more information on the actions to take fore messages that are received by a node. Most messages require some kind of transmission to be made afterwards.

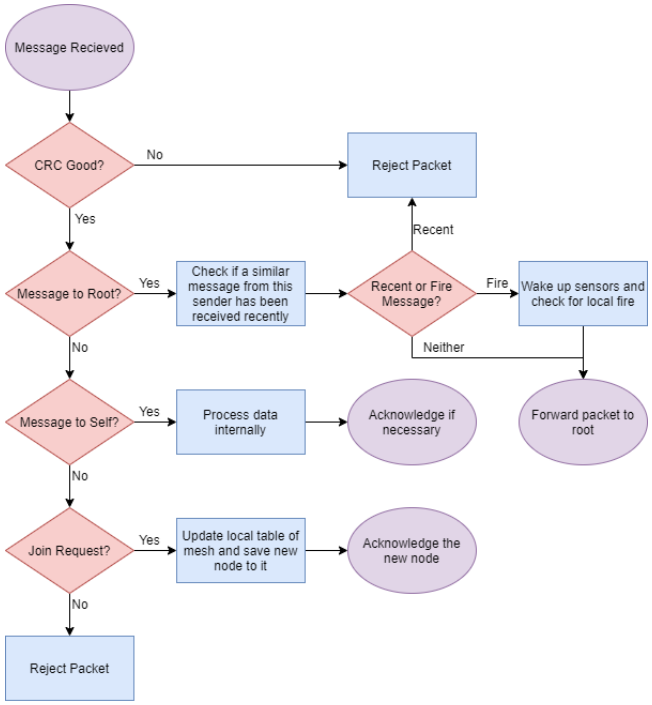


Figure 62: Actions taken on a Message Received event

The Raspberry Pi is an important part of the system as it determines if a fire exists or does not. From a “black box” perspective, the network controller will periodically ask “is there a fire to report?” and the raspberry pi responds with an answer. This is done in the final version by the network software reading a known file and the sensor software writing a confidence value to the same file. This kind of “request” structure is important as the network controller may have stored request packets in an internal buffer and there may be multiple tasks for the raspberry pi to complete when it is ready to receive requests. This methodology allows for multiple groups of data to be put in the network software’s send queue and sent, one at a time, into the network. Once the Raspberry Pi has serviced all the sensors, it informs the network software that it is done so the network software can continue. It is up to the sensor software to decide if it is finished or read sensors or do whatever it needs to do. The sensor software does not need to wait for the network software and can immediately finish whenever it decides.

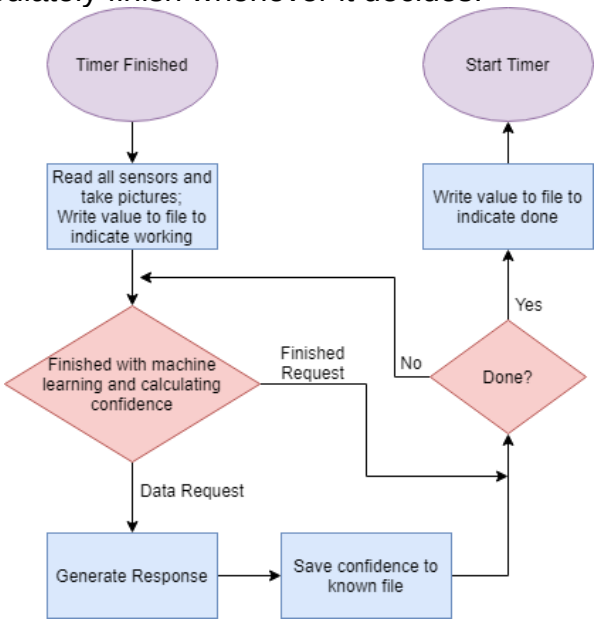


Figure 63: Raspberry Pi decision making

5.3.5. Non-Volatile Storage of Configuration & Packet Buffer Loss

The system stores some information in non-volatile memory to ensure that upon power loss all configuration items stay intact. Any machine learning, image sensing, and previous sensed data (with regards to detection confidence) must be saved to non-volatile memory prior to shut down conditions. The Network software, on the other hand, only saves its network map to non-volatile memory as a stretch goal. All other data is considered volatile and can be changed. This decision caries the implication that if the network is busy and power is unexpectedly removed from the network controller, all pending packets will be lost. These packets cannot be recovered. Ideally, however, the network can recover from this immediately since all nodes can forward all packets to the

root. In this case, a new route may be found by the network. If the network is not created with this in mind when installing the network, the node may not be able to transmit to any other nodes. Packets will be lost with no chance of recovery in this case so care should be taken when setting up a network.

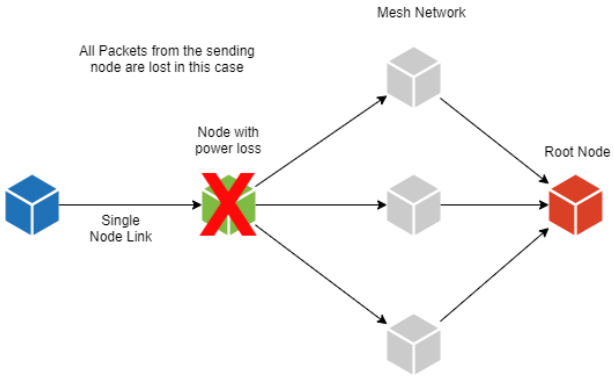


Figure 64: Lost Packets Diagram

The Raspberry Pi also stores important data like the models used to detect fires since they will only improve over time. Therefore, current conditions must be saved in such a way that when power is lost, we do not lose the current state or the previous conditions. When the Raspberry Pi turns on it will load the data into memory and then process that data.

5.3.6. Network Packet Types

To transmit and understand information effectively, the system utilizes opcodes to know which action to take on different packets. The different packets are defined in the table below. The major packet types are “Fire Packets” and “Join Request” packets. Fire Packets contain information for the root node of whether there is a fire and at which location that fire may reside. This packet is forwarded to the root by other nodes in the mesh, however whenever a node attempts to forward a valid Fire Packet, it also will wake up the sensors and see if a fire is in its local area. Join Request packets are for nodes in the local area.

Any node that hears a join request responds and lets the joining node know that it can hear it and that it is ready to receive messages. The other packets contain information to or from the root node that may be pertinent. Heartbeat Message packets are periodically sent out by nodes that are only read by nodes in the local area. The heartbeat may receive an acknowledgement so that the sender knows it is still in the network and can decide which nodes are ideal to send to, as a stretch goal. If multiple heartbeats are sent out without responses, then the sending node may have low confidence that the nodes in its internal connection list are still connected. “Node Messages” Can be sent from the root or another node and may contain control data such as a “I’ve heard your message” response or acknowledgements.

Table 11: Packet Types

Opcode	Name	Purpose
0xA1	Fire Packet	Alert that a fire has been detected. May contain data to describe the sensor readings.
0xB1	Generic Root Message	Generic message for the root node that may contain ASCII text as a payload.
0xB2	Binary Root Message	Message for the root node that contains binary data as a payload.
0xC1	Heartbeat Message	Packet contains nothing. This is meant to show that the node is alive.
0xC2	Join Request	Request to join the network. Allows the node to send and receive data from the network. Contains a randomly generated UID and possibly other data.
0xD1	Debug Message	Could contain anything. Software Defined.
0xE1	Node Message	Message to a node instead of to the root node. Follows the same structure as 0xB1 (Generic Message).

5.4. Computer Vision

Machine learning will be used to implement computer vision for our system to detect fire in forests. Machine learning is a topical subject that has appeared in recent years. In our project, it is useful to classify images as “fire” or “not fire”. This classification and identification of different features of fire makes our design case a decent candidate for machine learning. By implementing and training a machine learning algorithm correctly, the system should be able to identify, with confidence, a fire rather quickly.

Although there are many available resources and libraries for computer vision to detect fires, they are not accommodated to the processing power of Raspberry Pi. GPU is often used to implement these functions specially to train the model to a certain dataset as it

can be very large and may take large amount of processing. Our main task will be focusing on how to tackle the issues due to utilizing Raspberry Pi such as slower processing, memory limit, and limited power consumption.

5.4.1. Color Classification plus Optical Flow

Fire is a distinct object in a forest environment. It has contrasting color as well as distinct movements (flickering and spreading) compared to the rest. To utilize this information, we decided to implement color classification to isolate pixels with the colors of fire and detect motion using optical flow with the help of OpenCV.

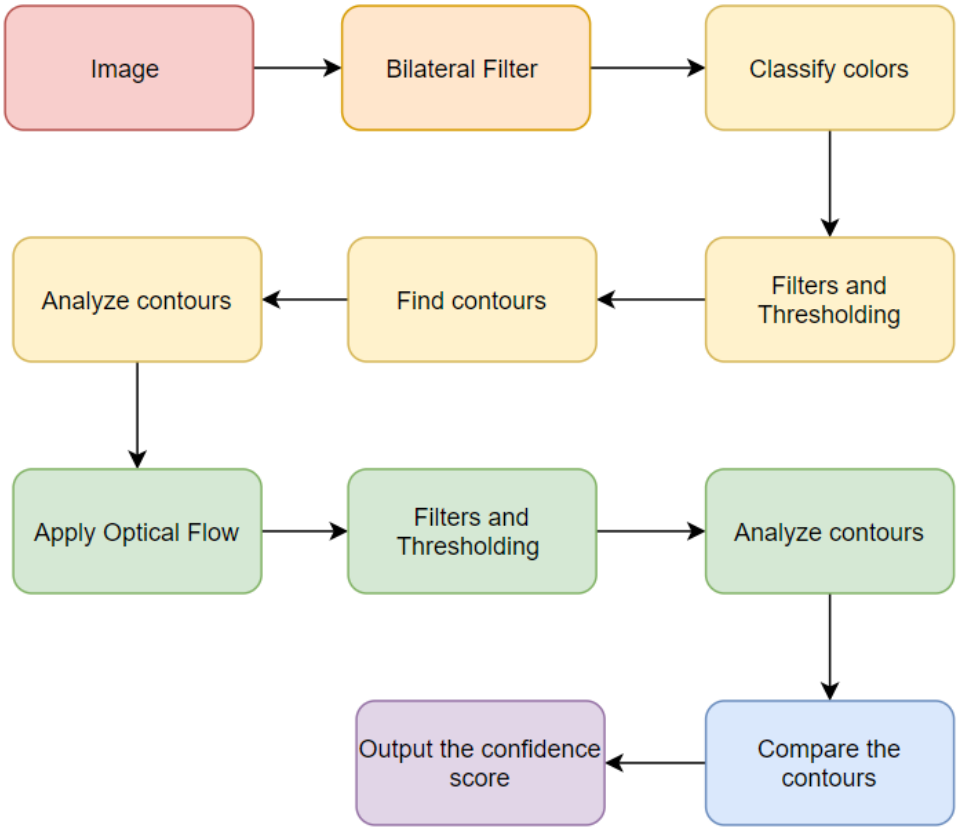


Figure 65: Overview design of color classification plus optical flow

At least two images are needed for the optical flow to calculate the motion. Thus, two images will be taken using the camera attached to the Raspberry Pi Zero. Then the bilateral filter is applied to these images to blur the image while keeping the edges sharp. By doing so, the colors of fire will appear more sharply and clearly in some cases. Then the color classification is applied. The range of colors will be defined beforehand. In our project, we defined minimum RGB values to be (100, 17, 15) and maximum RGB values to be (255, 255, 180). The pixels will be isolated based on this range of RGB values. The rest will be masked. To find the contours of the fire, the image needs to be blurred using Gaussian Blur and resized slightly. Then, it is converted to grayscale to apply thresholds to the image. The contours are found by applying the findContours method from OpenCV

to the threshold image. These contours are then sorted based on the area size. The small area sizes are ignored to avoid false positives and excess detection.

To utilize optical flow, the original images are used as frame 1 and 2. Then, they are converted into grayscale. The Gunner Farneback's algorithm is used for two-frame motion estimation based on polynomial expansion. The hue value indicates the direction of the motion while the value plane corresponds to the magnitude of the motion. To find the contours in the optical flow image, methods used in color classification are similarly used (Gaussian blur, thresholding, findContours method, sort and restrict by area size). If there is at least one contour found in color classification, 35 is added to the confidence score. Similarly, if there is at least one contour found in optical flow, 35 is added to the confidence score. The last 30 comes from the overlap of the top contours between the two methods. The overlap indicates that there is an object with colors similar to the fire while having dense movements. The total confidence score obtainable from the color classification plus optical flow method is 100 (35 + 35 + 30).

5.4.2. Machine Learning with Raspberry Pi Zero

To have computer vision in Raspberry Pi Zero, we must pick the right packages and neural network architecture to obtain best possible processing time and accuracy. Upon experimenting and testing different methods, we found out that TensorFlow 1.8.0 and Keras 2.1.5 work best with Raspberry Pi Zero using Python 2.7. Since Raspberry Pi Zero is not optimal to train a model, we utilized Google Colab to train our model and upload the trained model to the Raspberry Pi to predict the fire in an image taken by its camera in addition to color classification plus optical flow method.

Due to hardware restrictions and specific need to detect fire, training a model was much easier than making pre-trained models such as YOLO work in our system. Our neural network architecture is inspired by the Fire Detection Net [36].

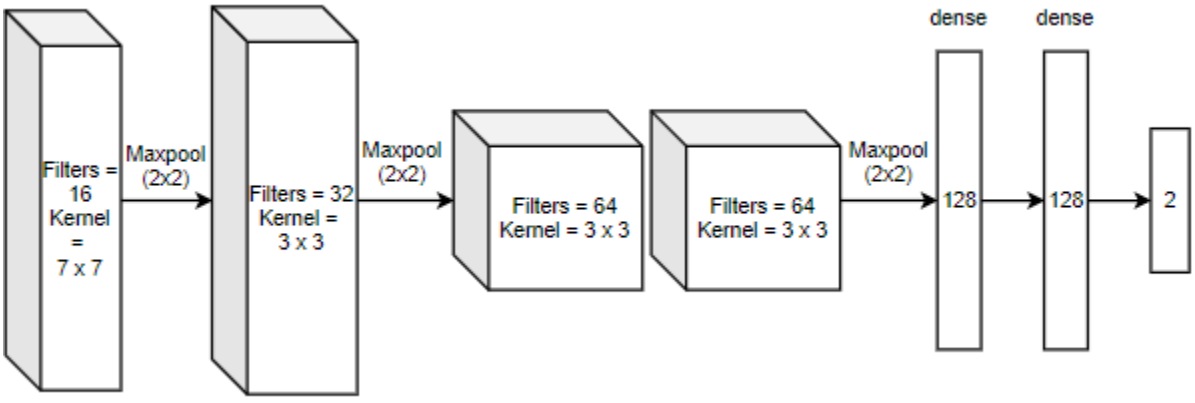


Figure 66: Fire Detection Net Architecture

Through several experimentations and testing, we decided to use out current neural network architecture described in the diagram below. Minimizing the parameters resulted in 10 to 20 seconds faster processing time, but the accuracy significantly dropped. Thus, we decided to focus on simpler designs. At the end, we were able to achieve one percent higher accuracy with the model trained with the network architecture below.

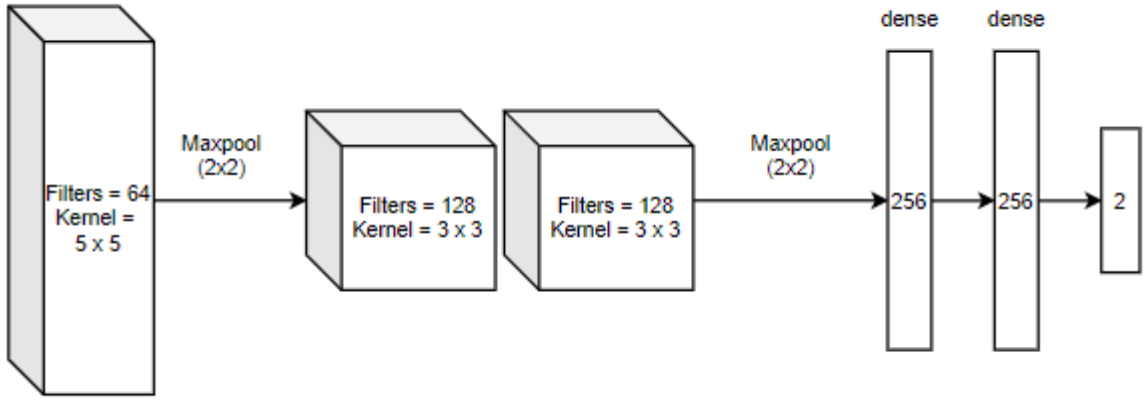


Figure 67: Current neural network architecture

The result of our training is shown in the graph below. It uses binary cross entropy for the loss, ReLu for activation function, batch size of 32, learning rate of 1e-5, and 200 epochs. The highest validation accuracy was 93.36%. The accuracy corresponds to the confidence score (total of 100 for machine learning).

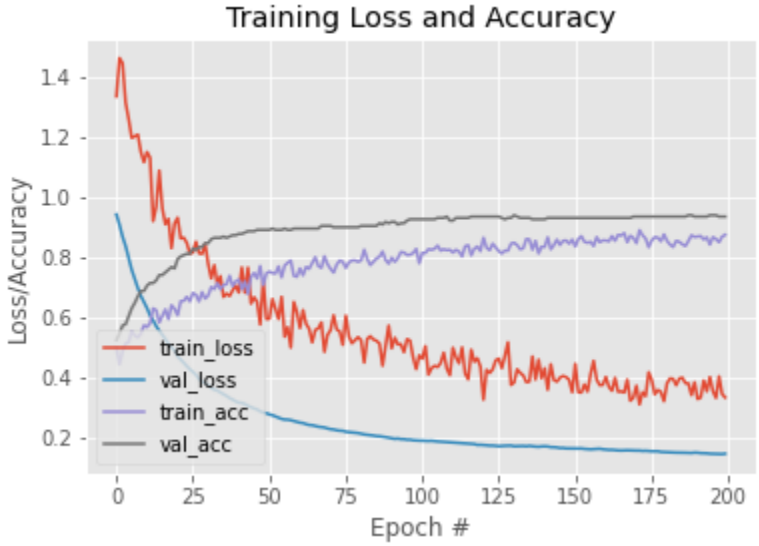


Figure 68: Results of Training

5.4.2.1 Dataset

Dataset is a crucial aspect in training the neural network. The accuracy depends heavily on the quantity and quality of the data. We did not find any free accessible forest fire specific dataset, so we decided to create our own dataset by acquiring online images relating to forest fires and forest (with no fire) to train and test our models with the help of Microsoft API [47]. The training went well for this specific dataset. However, we hypothesize that for best results, it is recommended to acquire these images by using the camera attached with the Raspberry Pi Zero so that the model is trained with appropriate environment and quality of images.

5.4.3. Final Design of Computer Vision

Now we must build the color classification, optical flow, and machine learning together to calculate the overall confidence score. There are total of 200 confidence score obtainable from the computer vision aspect. There are other confidence scores obtainable from the sensors. We take these values and take the average to obtain our final confidence score (maximum 100). If the confidence score is above 50, there is a fire detected. The threshold and weights of each sensors should be finetuned for best results. The overview of the combined software design is shown below.

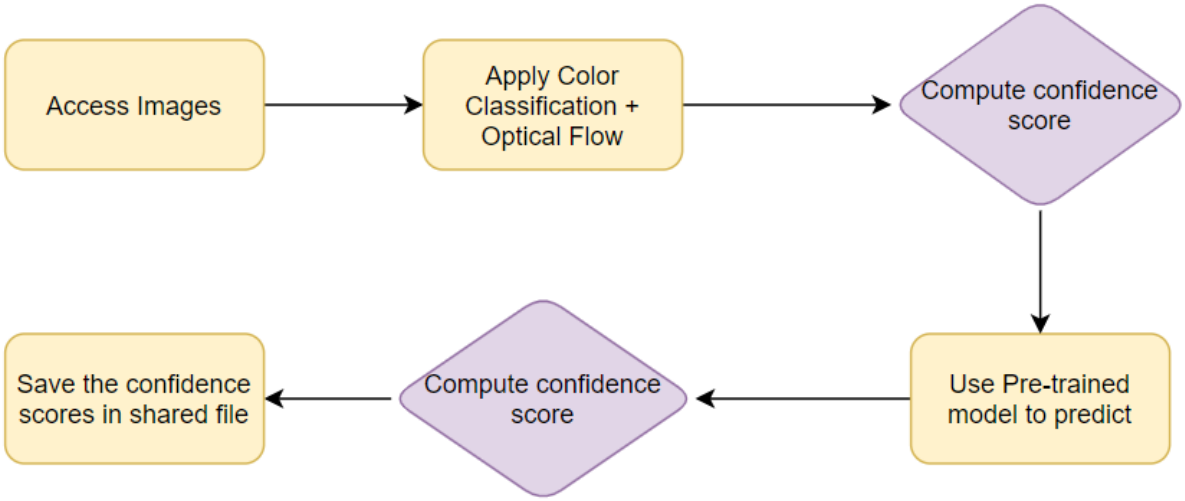


Figure 69: Overview design of computer vision

6. Testing and Prototyping

Testing the hardware and software was almost as important as the project itself. To ensure that everything was going to plan, each component and subsystem was tested separately first and then moved on to a more integrated testing system as time went on. This increased confidence that the final prototype was as functional as possible since all the bugs were worked out on a smaller scale before integration. The first section below deals with the advancement of our knowledge and experience with the subsystems that

prepared us for the final design. The sections after deal with testing a semi-final and final design.

6.1. From Nothing to Something

All projects must begin from somewhere. There are a few subsystems that must work together to complete the project: Power, Sensors, Network, Processing. This section deals with the process of prototyping in preparation for a final design.

6.1.1. Power Subsystem

Nothing in the system will function without the Power Subsystem. The power subsystem utilizes 3 different technologies to allow the full system to work. The first system that needed to be designed was the power regulation. We had identified through research that we will be using switching buck regulators for their efficiency. Energy lost to heat would not be ideal in a battery system. Tunable switching buck regulators were purchased to aid in simplifying the system and to account for any error in the power delivery. From this point some calculations were done to determine their efficiency and, using different loads, we will attempt to prove that the required power can be supplied by the regulator.

After the regulator design was investigated, it was important to work on the battery design. We simply need to prove that we can charge and discharge batteries. Our batteries and battery charging circuit were put together on a breadboard and we attempted to charge and discharge the batteries. Safety was very important as the batteries are an energy storage device. This proof of concept was able to prove that we can charge batteries and use them efficiently. A few charge and discharge cycles were observed to prove that the battery will not deteriorate quickly over the course of only a few discharges.

In the end, the system is meant to be powered from a solar panel. A solar panel was used to generate power from either the sun or a bright light. Use of the solar panel was investigated so we could become familiar with the characteristics of the device and how it affected our circuit. In the end, all three parts of this subsystem was combined with a simulated load such that we saw the power system could support everything. After testing this system, a layout was put together to put this system on a PCB and the design for the mechanical support structure for the solar panels was finished.

6.1.2. Sensor Subsystem

Testing the Sensor Subsystem took place in two parts: retrieving sensor data reliably and retrieving accurate data. To facilitate these tests, a Raspberry Pi was used so that the team could become familiar with the Raspberry Pi’s interface and operating system. The goal of the first portion of testing was to investigate the ease of use of the sensors. The sensor was hooked up on a breadboard, to power and all the supporting hardware was given to the device. Then the Raspberry Pi attempted to retrieve data from the device through I2C. At first the result’s values did not matter; just that results existed. All the sensors were tested to ensure their suitability for the project. It was at this point that a

sensor's complexity and difficulty was assessed and if it was deemed suitable it would go into calibration stage. This step was important to determine what sensors would be used in the final prototype.

Since the flame, smoke, and gas sensors were all modules and did not need test boards it was simple to test the sensors individually. The flame and smoke sensor both used an analog front end that converted analog signals to binary unitless values. Once the data bytes were converted to integers, an algorithm was developed to process the data and determine if a flame/smoke was detected after exposing the sensor to fire and none fire conditions. The gas sensor needed 48 hours of calibration before any data can be processed. Once it was properly calibrated, the data bytes were converted, and the compensation values were taken into account for calculation of the gas, temperature, and humidity measurement. The sensor was then exposed to fire and non-fire conditions to understand the gas measurements range are in the fire. The sensors went through several testing and calibration to tune the algorithm.

The camera will be attached to the Raspberry Pi and can be operated through commands or Python file. It will be able to record a video or take images depending on what the user needs. Testing both functions will be beneficial to ensure proper operation of the camera. For computer vision, it is important that the camera will be able to save multiple images to designated folder.

6.1.3. Network Subsystem

Testing the Network Subsystem consisted of a few parts to ensure that the Network software and hardware works all together. The first step was programing some development kits that use the SAMR34 as the processor. The development kit is the SAMR34 Xplained Pro Evaluation Kit. The SAMR34 has a built in Semtech SX1276 LoRa transceiver which gives the team a chance to get a feel for the software and hardware requirements since the plan is to use the SAMR35 microcontroller. Using two of these evaluation kits, software is written to send text to a screen when a button is pushed. This is important as serial communication is planned to be the method of communication between the Network Subsystem and the Processing subsystem. During testing, the SX1276 transceiver inside the SAMR34 was not functioning. This could be due to many things. To get it working, weeks were spent tweaking register values and attempting to get the system working. In the end, emails were sent to Semtech and Microchip to attempt to get it working. To get software working for the final prototype, two RYLR896 modules were bought to just get LoRa working in general with the SAMR34/35.

From there a program was written to allow a button push to turn on the built-in user LED on the other device. The LoRa protocol was used to complete this. This proof of concept step is important as it lays the foundation to sending data over the LoRa based network. The final piece of testing that went into the development of the system is to have user input. The user will type a string into a terminal that serially sends the data to the network controller. The network controller sent this string to the other device and the other device printed the data to another terminal. After this test, the software was ready to be

developed for the mesh network protocol since data can be transferred between the two evaluation kits. The evaluation kits will be used to develop the software until the time functioning PCBs for the final system are available for testing.

In the end, the RYLR896 modules worked with the SAMR34 Xplained PRO but when the final PCB was delivered, soldering issues were suspected with the SAMR35 and so these modules would not work with the PCB. To mitigate these issues RFM95W modules were purchased since they use the SX127x transceivers. Since they use SPI with the native module, the SPI bus was connected to the Raspberry Pi. This meant that the network software was now to be written for the Raspberry Pi to interface with the module.

The next part of testing for the Network Subsystem is the internal timers and GPIO pins. On a breadboard, the Network Subsystem connects to LEDs. This is to simulate the fire condition on the final system. The final round of testing included implementing the different packets and interpreting the LoRa binary data. This was successful and resulted in the implementation of the final design as described in this paper.

6.1.4. Processing Subsystem

Testing the Processing Subsystem is important as this is the subsystem that determines if a fire is in the area. There are three parts to developing the Processing Subsystem. To test the Processing Subsystem, first, a Raspberry Pi was set up with all the software necessary to perform computer vision algorithms and run Python code. Ensure that the camera is working using the Raspberry Pi and able to save the captured images to a folder accessible for machine learning algorithms. Each sensor outputs raw data that is then analyzed to create a data processing algorithm in python to determine probabilities of gas, smoke, and flame.

Since Raspberry Pi Zero cannot handle training the model using its processing power, we download a trained model into it and run the prediction using that model. To have the model trained, we used Google Colab, a free cloud service, in order to utilize its free GPU to train efficiently. It was ensured to install TensorFlow 1.8.0 and Keras 2.1.5 on the Google Colab notebook as well. Once a model is trained, it was run using the Raspberry Pi to test it. After testing different versions of Tensorflow, TensorFlow Lite, and Keras, it was determined that the best version to work with Raspberry Pi Zero is TensorFlow 1.8.0 and Keras 2.1.5. Once the model is ready to be tested, it was run along the other methods of computer vision which are color classification and optical flow to find the total confidence score of detecting fire.

However, the subsystem can significantly improve its accuracy by having more data for it to be trained. The best data would derive from the camera that the subsystem is utilizing along with consistent environment for it to capture images. By doing so, the neural network will be able to learn appropriately for its usage and result in higher accuracy compared to random online images that we currently use as the dataset. This is proven to work as our current model achieves 93% accuracy testing with our current dataset.

The second part of prototyping and testing the Processing Subsystem will be the interaction between the Processing Subsystem and the sensors. To go about testing this, the sensors were introduced into the system one at a time and it was ensured that the connections between the raspberry pi and the sensors works appropriately. For the gas sensor, a logic analyzer was used to see the signals sent to and from the Raspberry Pi and the sensor. This testing was meant to focus on the software interaction between the Raspberry Pi and the sensors since issues were noticed during integration.

Finally, the Sensor software was tested with the Network Subsystem. The Raspberry Pi attempts to write information to a file. This kind of testing will not focus on the software interacting correctly as in the final system, although it could simulate it. The goal of this testing is to verify that messages can be sent reliably between the two devices. The network software, now written for the Raspberry Pi and not the SAMR35 reads the file and takes some action based on the values in that file. These values are used later to determine if a fire is present and allows the network software to send a “fire found” packet through the network.

6.2. Step-by-Step Hardware Test Plan

Without the hardware, the software cannot do its job. It is imperative that the hardware operates on a reliable base for the software to be built upon so that the prototype functions in all conditions: day, night, harsh weather, or perfect, clear skies. The following sections discuss some of the step-by-step plans for testing the hardware components and why it was useful to do so.

6.2.1. Power

Stable power is the backbone to the entire circuit. Power is the only sure thing that a circuit must have working to perform its function. To test the power systems a step-by-step plan is introduced.

Step-by-step:

a. Set up all power supplies to the expected nominal voltage from our solar panels and allow for as much current draw that is necessary
b. Test all power converters and regulators separately and measure their outputs. Test them under the expected load of the system and make sure they perform.
c. Modify the load and map their efficiency to ensure proper operation.
d. Starve the converters and regulators of current and observe their effects on the simulated load. Make note of the minimum current the converters and regulators can maintain
e. Repeat the above steps for lower than nominal voltages and higher than nominal voltages. Do not exceed the recommended highest voltage of each converter or regulator.

f. Test different circuit protection techniques to help for overvoltage and overcurrent conditions.
g. Set up the charging circuit for the batteries and give it nominal conditions for charging and observe the effects.
h. Connect all the systems together, including power supply and load to get a fully working power system.
i. Shift the power supply to a solar panel and test it with a bright light source and/or sunlight.

6.2.2. Hardware Sensor Testing

Sensors will require hardware testing and software development discussed later in the paper. Hardware testing of the sensor included testing the physical capabilities of the sensor. The sensors use I2C, therefore it is important to ensure the sensors are visible in the I2C-detect when connected to the raspberry pi. Therefore, a testing procedure must be put in place.
Step-by-step:

a. Power sensors and connect SCL and SDL lines
b. Establish connection between raspberry pi and sensors through I2C communication to see if results can be read.
a. Create a circuit for the sensors and see if they respond to any external stimuli:
    a. Gas sensor will be exposed to nearby smoke and fire to test detection of organic gases, and measure temperature and humidity levels.
    b. Flame sensors will be exposed to nearby flames to detect flame flicker
    c. The camera will be exposed to nearby flames to record fire and non-fire data as explained in section 6.3.3 Computer Vision
    d. Smoke sensor will be exposed to nearby smoke to assess smoke is detected.
c. Hardware testing went through multiple trial and error runs with varying levels of gas, smoke, and flame exposure to not only obtain raw data but also to test the minimum and maximum capabilities of the sensor. Understanding the minimum and maximum capabilities will help determine the distance range between each device in the forest.
d. Then the software code was written to read the result and try to get meaningful data after converting data bytes to integers. The output was checked to see if fell within an expected and acceptable range.
e. The sensor reading was converted to meaningful “real world” values and ensure they are acceptable for real world scenarios (especially the current scenario the sensor is in).
        a. Gas, temperature, and humidity measurements in ohms, Celsius, and %
        b. Flame and smoke sensors to count the number of consecutive True changes detected.
f. Provide the data to machine learning engineer to use for algorithm development.

6.2.3. Controllers

Testing the controllers is important since these pieces of hardware control everything. Each system has its own set of requirements, however.

6.2.3.1. Raspberry Pi

The Raspberry Pi is a computer with a very small footprint. Since it runs a distribution of Linux, we should ensure that the Raspberry Pi can boot properly and can run software.

The test procedure is step-by-step as follows:

a. Boot into the Raspberry Pi operating system and interact with the terminal
b. Write some software to toggle GPIO pins, maybe to turn on and off an LED
c. Record power usage under idle and stressed conditions
d. Output SPI, I2C, and/or UART with the Raspberry Pi.
e. Using one of the voltage regulators in the Power subsystem, power the raspberry pi from that instead of the normal USB power (using pins 2 or 4 on the header pins). Repeat the following steps to ensure everything is working.

6.2.3.2. SAMR35

The SAMR35 is a full-on embedded microcontroller. As such, it does not use an operating system (unless one is uploaded onto it). For hardware testing, a simple plan can be put in place to test the different possibly required peripherals and ensure the chip is working correctly.

Step-by-step:

a. Program the chip to toggle a GPIO pin, possibly turning on and off an LED. Use delays based on timer interrupts if possible
b. Program the chip to output SPI, I2C, and/or UART
c. Record power usage under idle and stressed conditions
d. Using one of the voltage regulators in the Power subsystem, power the chip from that instead of a power supply
e. If using a Real Time Operating System, schedule two jobs to run concurrently and see how they interact. Using an oscilloscope see the delay between the two jobs if running concurrently.
f. Test RF capabilities if applicable/possible. Note: This step was attempted over weeks of development and did not result in success.

6.2.4. Radio Frequencies

Testing RF designs can be challenging. Testing this assumes that the SAMR35 has been tested and that some software has been written to interact with the LoRa peripherals. The following steps would have been done if the SAMR35 was successfully programmed to output LoRa.

Step-by-step:

a. Program the chip to send out data whether it be FSK or LoRa.
b. Watch a spectrum analyzer to see if that data is being transmitted in the air and if it is being transmitted properly
c. Take two devices and attempt simple communication, possibly light an LED on received data. Attempt to transmit larger packets as well like strings. Investigate streaming data.
d. Range test: With a working simple communication test, do some tests in different environments with range. Some ideas include: Line-of-Sight, in or around buildings/urban environments, wooded/forest environments. See how the range is affected. Measure results every 100-200m and expand until range is compromised.

In the end, the SAMR35 was not used to produce LoRa signals. For the final prototype separate SX127x modules were used. To test these modules, the following steps were used.

Step-by-step:

a. Program the Raspberry Pi to send SPI to the SX127x and get version information
b. Send a test transmission through LoRa to another Raspberry Pi
c. Print that data to a file
d. Implement basic packet structures
e. Send data that specifically turns on an LED

6.3. Step-by-Step Software Test Plan

Software is an important and critical piece of the prototype and must be done correctly to determine if there is a fire. Thus, proper testing of the software is important. The following sections outline step-by-step plans to testing the software components and why it might be useful to do so.

6.3.1. Connection Between the Hardware and Software

The hardware and software must work together to have a working prototype. To make sure this is the case, simple software was written to calibrate, initialize, test, and train the sensors and cameras to detect various characteristics of fire. This allowed the team to see if the implementation and ideas are feasible or if the implementation needs to be adjusted. The results from testing are then tuned to ensure that the sensors and camera were able to detect fire.

6.3.2. Software Development for Sensors from Hardware Testing

The sensors play an important role in the process of determining if there is a fire. Computer vision method is one option and will be discussed further in the paper; however, the other sensors can provide us with more confidence that there is, indeed, a fire. The sensors were used to monitor the environment and support computer vision in fire detection. The data from sensors can also provide indication on the type of fire. Sensor operation is depicted in the diagram of figure 74. Each sensor has its own data processing algorithm, which is discussed below:

Gas:
The data collection for the gas sensor is very straightforward. Since this sensor was calibrated for a long time, the measurements are stable and relatively accurate. The raw measurements for temperature, humidity, and gas are obtained and then compensated to improve accuracy. The final values are calculated in Celsius, percent, and ohms to interpret the data. Temperature and humidity readings from the can be used to monitor the environment and a warning can be sent when high temperatures exist in a dry environment. In fire conditions, the gas measurement rises significantly higher compared to non-fire conditions. When a gas measurement reaches a maximum value, a warning is sent to indicate high levels of volatile organic compounds present in the atmosphere.

Smoke:
The PIM438 module which includes the MAX30105 module receives a raw data from the IR and Visible light detector. The algorithm is designed to take the mean of incoming data and look for changes between the means by taking the difference between the recent mean and the mean taken X readings ago (delta). A change is detected when the delta value is greater than the threshold value. The mean size, delta size, and threshold can be tuned to increase data smoothing and sensitivity. A function was created to count the maximum number of consecutive True changes. If this value is greater than a set value, then flame has been detected.

Flame:
The sensing element in the ePY12241 provides an output current that is proportional to the rate of change of temperature of the material. The chip uses an analog front end to receive an analog signal, which is then filtered by a high pass filter. The signal then goes into a sigma delta ADC convertor. Then, the low pass filter removes large frequencies, and the data is then read by the MCU.

A data window is specified, and the RMS of the raw signal is taken to determine the signal strength of the combination of the frequencies in the bandpass of the filters used. The reading is then divided by the signal multiplier, which is the sample rate. From here the flame algorithm is similar to the smoke algorithm where the delta is calculated from the current value and the value X readings ago. A change is detected when the delta value is greater than the threshold value. The RMS data window, mean size, delta size, and threshold can be tuned to increase data smoothing and sensitivity. A function was created to count the maximum number of consecutive True changes. If this value is greater than a set value, then flame has been detected.

$$RMS = \frac{\sqrt{(d_1)^2 + (d_2)^2 + \cdots (d_n)^2}}{n}$$

dn is the raw data collected from the pyroelectric infrared detector.
n is the number of data collected in a data window (data window size).

Integration:
After exposing the three sensors to a fire burning in a grill, each sensor was able to successfully detect a fire. The gas sensor showed that when its exposed to a fire, the temperature increased greater than 60°C and the relative humidity was greater than 60%. However, this does not entirely indicate there is a fire, rather it can be used to determine potential forest fire condition. In other climates, the humidity can be lower. The gas measurement in ohms was significantly higher than in non-fire conditions.

The smoke sensor was able to detect more than 20 consecutive True changes which satisfied the requirement for smoke detection.

With a nearby fire, the most significant bit of the sensor is set which resulted in a flame detected. During a test using a lighter placed approximately 1m away in low light, the flame algorithm was able to detect flame with 10 consecutive true statements. This is expected since the lighter produces a smaller flame. The thresholds, channel and analog settings can be adjusted to increase or decrease the algorithm's sensitivity.

6.3.3. Computer Vision

The color classification and optical flow can be tested using the Raspberry Pi Zero. It is important for the camera was compatible Raspberry Pi Zero.

Step-by-step:

a. Install OpenCV and other basic packages to Raspberry Pi Zero
b. Enable camera and reboot
c. Test the camera to ensure it operates properly and saves the image through commands
d. Create a Python file to run the camera and save images to a folder
e. Create a Python file to run the code for color classification and optical flow by reading images saved in the folder
f. Finetune the parameters in filters, saturation, and so on
g. Check if the confidence score is properly saved in the designated txt file for later use

Here are several sample images from to ensure the code is working properly.

Figure 70: Sample results using color classification and optical flow

The figure above shows the original image on the left, the color classified image on the middle, and the image from optical flow on the right. The range of colors to be detected can be adjusted for better results.

Figure 71: Sample results of contouring

Detection of contours can also be adjusted by the area size, filters, and thresholding. Finetuning will most likely be necessary for specific environment.

For testing the machine learning, Google Colab and Raspberry Pi Zero are needed. The Raspberry Pi Zero cannot train a model due to its limited processing power. Thus, Google Colab becomes handy in training model and testing the model before running it in the Raspberry Pi. For both Google Colab and Raspberry Pi Zero, ensure that TensorFlow 1.8.0 and Keras 2.1.5 are installed for Python 2.7.

Step-by-step: Google Colab (can be ran with Python 3)

a. Ensure the correct version of TensorFlow and Keras are installed
b. Ensure access to the dataset
c. Train a model
d. Observe the training loss and accuracy
e. Adjust hyperparameters as needed
f. Test the trained model

Once the model is trained, it is ready to be used in Raspberry Pi Zero.

Step-by-step: Raspberry Pi Zero

a. Ensure the correct version of TensorFlow and Keras are installed
b. Ensure to have the correct trained model
c. Test the model
d. Observe the accuracy, it will correspond to the confidence score
e. Adjust hyperparameters as needed
f. Combine the code from color classification and optical flow

There should be two confidence scores resulting from the color classification plus optical flow and machine learning which are saved in shared file for later integration with the sensors.

6.3.4. Networking

The network hardware was tested in a step-by-step procedure. First, sending simple strings between the devices. Once strings were sent, the software could be developed. These tests describe the testing done after the switch to using the Raspberry Pi with LoRa modules instead of the SAMR35.

Step-by-step:

a. Turn on LoRa modules with Raspberry Pi and configure them to send/receive LoRa frames.
b. Set up receiver to wait for RX interrupt and print string that fired that interrupt to the terminal
c. Write to the transmit FIFO with SPI on the transmitter.
d. Read back the FIFO to confirm that it is saved
e. Set the transceiver to “TX” mode
f. Watch the terminal

From this point on, the device is ready to run the Network Software.

6.4. Prototype Construction

The sensor subsystem, power system, and RF subsystem were individually designed on KiCad and then integrated into a single schematic using hierarchical pages. The PCB layout was then routed. Using any PCB manufacturer, the specifications of the PCB used to purchase it are a 2-layer FR-4 PCB with 0.8mm thickness and 6/6mil minimum track/spacing. The minimum Hole size was .3mm and lead free HASL was chosen for the surface finish. 1oz copper was used as it was cheaper. To solder the surface mount components on the top layer, a stencil was also purchased.

6.4.1. Equipment

The follow equipment was used to complete testing and building the prototype:

A soldering iron kit with lead free solder, hand-held multimeter, tweezers, plyers, logic analyzer, breadboard, general components (such as resistors, capacitors, inductors), jumper wires, and wire strippers.

6.4.2. Bill of Materials

The following table is a detailed bill of materials of the parts and components used in the project.

Table 12: BOM

Part Name	General Description	System	Part Number	Qty	Unit Cost	Total Cost	Supplier
Solar Panel	Solar panel	Power	Generic	2	$4.99	$9.98	Ebay
AC-DC 5V 3A	power supply	Power	AL5V3ABK	1	$8.99	$8.99	Amazon
MSOP to Dip SMT Adapters	adapter	Power	IPC0078-ND	1	$6.29	$6.29	Digikey
Analog Devices LT3652EMSE#PBF	Li-Ion charging IC	Power	LT3652EMSE#PBF-ND	2	$7.61	$15.22	Digikey
BK-18650-PC4	Double 18650 Cell holder	Power	BK-18650-PC4-ND	1	$5.73	$5.73	Digikey
EEU-FC1V391S	Capacitor	Power	P10300-ND	1	$0.28	$0.28	Digikey
EMK212BJ106KG-T	Capacitor	Power	587-1295-1-ND	2	$0.20	$0.40	Digikey
ECA-1EHG101	Capacitor	Power	P5540-ND	1	$0.29	$0.29	Digikey
08053C105KAT2A	Capacitor	Power	478-5030-1-ND	1	$0.26	$0.26	Digikey
MBRS130LT3G	Schottky diode	Power	MBRS130LT3GOSCT-ND	3	$0.47	$1.41	Digikey
LQM18FN100M00D	Inductor	Power	490-4025-1-ND	1	$0.15	$0.15	Digikey
ERJ-3EKF7873V	Resistor	Power	P787KHCT-ND	1	$0.10	$0.10	Digikey

MCU0805-100K-CFCT-ND	Resistor	Power	MCU0805-100K-CFCT-ND	2	$0.39	$0.78	Digikey
A102249CT-ND	Resistor	Power	A102249CT-ND	1	$0.19	$0.19	Digikey
619kOhm	Resistor	Power	RMCF0805FT619K	1	$0.10	$0.10	Digikey
412kOhm resistor	Resistor	Power	541-412KHCT-ND	0	$0.10	$0.00	Digikey
412kOhm resistor	Resistor	Power	RC0603FR-07412KL	1	$0.10	$0.10	Digikey
CRA2512-FZ-R050ELF	Resistor	Power	CRA2512-FZ-R050ELF	2	$0.60	$1.20	Digikey
CRA2512-FZ-R100ELFCT-ND	Resistor	Power	CRA2512-FZ-R100ELFCT-ND	2	$0.56	$1.12	Digikey
910-PA0006	16 SOIC to DIP adapter	Sensor	PA0006	1	$4.09	$4.09	Mouser
Pyreos EPY12241	Flame sensor	Sensor	EPY12241-B1	1	$41.54	$41.54	Mouser
Microchip Technology 579-RE46C190S16F	Smoke Sensor	Sensor	RE46C190S16F	1	$1.30	$1.30	Mouser
Sensiron SVM30-J	Gas sensor	Sensor	403-SVM30-J	1	$20.80	$20.80	Mouser
100uF	Capacitor	general	GRM31CR61A107ME05L	1	$0.78	$0.78	Mouser
10uF	Capacitor	general	C0805C106M9PAC	2	$0.15	$0.30	Mouser
1uF	Capacitor	general	8850122070 78	3	$0.10	$0.30	Mouser
4.7uF	Capacitor	general	0805X475M250CT	1	$0.10	$0.10	Mouser
33uF	Capacitor	general	C2012X5R0J336M125AC	1	$0.87	$0.87	Mouser
.1uF	Capacitor	general	0805ZD104KAT2A	14	$0.10	$1.40	Mouser

33pF	Capacitor	general	VJ0805A330GXXCW1BC	1	$0.29	$0.29	Mouser
4.7pF	Capacitor	general	VJ0805A4R7BXXCW1BC	1	$0.10	$0.10	Mouser
17pF	Capacitor	general	0603N170J500CT	0	$0.10	$0.00	Mouser
17pF	Capacitor	general	GRM0335C1H180GA01D	2	$0.10	$0.20	Mouser
4.7nF	Capacitor	general	8850122080007	1	$0.14	$0.14	Mouser
3.9pF	Capacitor	general	0805N3R9B500CT	2	$0.13	$0.26	Mouser
22pF	Capacitor	general	VJ0805A220GXXCW1BC	1	$0.29	$0.29	Mouser
18pF	Capacitor	general	0805N180J500CT	1	$0.10	$0.10	Mouser
3.3pF	Capacitor	general	RF18N3R3B250CT	3	$0.28	$0.84	Mouser
8.2pF	Capacitor	general	08053A8R2CAT2A	1	$0.23	$0.23	Mouser
5.6pF	Capacitor	general	06033A5R6BAT2A	2	$0.76	$1.52	Mouser
1nF	Capacitor	general	8850122070033	1	$0.10	$0.10	Mouser
47pF	Capacitor	general	VJ0805A470GXXCW1BC	1	$0.29	$0.29	Mouser
2.7pF	Capacitor	general	VJ0805A2R7BXXPW1BC	1	$0.25	$0.25	Mouser
Diode	Schottky	Power	STPS2L40AFN	2	$0.36	$0.72	Mouser
Transistor	Transistor for Relay	Power	2N7002NXAKR	3	$0.10	$0.30	Mouser
Yellow	LED	Sensor	LTL2R3KYD-EM	3	$0.10	$0.30	Mouser
Green	LED	Sensor	LTL2R3KGD-EM	2	$0.10	$0.20	Mouser
IR Emitter + Photodiode	IR Emitter + Photodiode	General	SEN-00241	2	$1.95	$3.90	Mouser

SAMR35	Network Controller	Communication	ATSAMR35J18BT-I/7JX	0	$5.64	$0.00	Mouser
SAMR35	network Controller	Communication	ATSAMR35J18B-I/7JX	3	$5.64	$16.92	Mouser
RF Switch	RF Switch for Band select	Communication	SKY13373-460LF	2	$1.20	$2.40	Mouser
Connector	Connector for Flame Sensor	Sensor	61301021821	2	$0.74	$1.48	Mouser
Connector	Prog. Conn. for Smoke Sensor	Sensor	A3B-12PA-2DSA(51)	2	$0.57	$1.14	Mouser
Connector	Connector for Pi Zero	Communication	61304021821	2	$1.79	$3.58	Mouser
Connector	Programming Conn. For SAMR35	Communication	FTSH-105-01-F-DV	2	$1.47	$2.94	Mouser
Connector	Wire connector for gas sensor	Sensor	DF51K-4DS-2C(800)	2	$0.23	$0.46	Mouser
Connector	Connector for Gas Sensor	Sensor	455-1751-1-ND	2	$0.79	$1.58	Digikey
SMA Connector	Connector to an Antenna	Communication	142-0701-201	2	$2.79	$5.58	Mouser
10uH	Inductor	General	RLB0912-100KL	1	$0.31	$0.31	Mouser
Ferrite bead	Inductor	General	MMZ2012D121BT000	3	$0.11	$0.33	Mouser
2.2nH	Inductor	General	L06032R2CGSTR	1	$0.61	$0.61	Mouser
33nH	Inductor	General	AIMC-0805-33NJ-T	2	$0.14	$0.28	Mouser
11nH	Inductor	General	LQP03HQ11NH02D	5	$0.21	$1.05	Mouser

10nH	Inductor	General	AIMC-0805-10NJ-T	0	$0.16	$0.00	Mouser
10nH	Inductor	general	CE201210-10NJ	5	$0.11	$0.55	Mouser
100R	Resistor	General	CRCW1206100RFKEAC	1	$0.10	$0.10	Mouser
0R	Resistor	General	CRCW12060000Z0EAC	10	$0.10	$1.00	Mouser
330R	Resistor	General	CRCW1206330RFKEAC	2	$0.10	$0.20	Mouser
1k	Resistor	General	RR1220P-102-D	10	$0.10	$1.00	Mouser
100k	Resistor	General	RR1220P-104-D	3	$0.10	$0.30	Mouser
39R	Resistor	General	RN73R1JTTD39R0D25	0	$0.35	$0.00	Mouser
39R	Resistor	General	RT1206FRE0739RL	2	$0.19	$0.38	Mouser
32.768kHz	Oscillator	General	COM-00540	2	$1.50	$3.00	Mouser
32MHz	Oscillator	General	1664-1300-1-ND	2	$2.91	$5.82	Digikey
Button for the Join and Reset	Button	Sensor	COM-11997	3	$0.98	$2.94	Mouser
MR5FT50L0	Resistor	Power Testing	MR5FT50L0CT-ND	1	$1.54	$1.54	Digikey
EEU-FC1V391S	Capacitor	Power Testing	P10300-ND	1	$0.28	$0.28	Digikey
RSMF1JTR100	Resistor	Power Testing	RSMF1JTR100CT-ND	1	$0.37	$0.37	Digikey
MFR-25FBF52-787K	Resistor	Power Testing	787KXBK-ND	1	$0.10	$0.10	Digikey
RSF2JT100K	Resistor	Power Testing	RSF2JT100KCT-ND	2	$0.29	$0.58	Digikey
MFR-25FBF52-280K	Resistor	Power Testing	280KXBK-ND	1	$0.10	$0.10	Digikey

MFR-25FBF52-412K	Resistor	Power Testing	412KXBK-ND	1	$0.10	$0.10	Digikey
MFR-25FBF52-619K	Resistor	Power Testing	619KXBK-ND	1	$0.10	$0.10	Digikey
ECQ-E2105KF	Capacitor	Power Testing	EF2105-ND	1	$0.58	$0.58	Digikey
1N5822-TP	Schottky diode	Power Testing	1N5822-TPMSCT-ND	3	$0.45	$1.35	Digikey
Lora Module SX1276	Communication Module	Communication	REYAX RYLR896	2	$19.50	$39.00	Amazon
JST PH Cable Connector	Cable connector for gas sensor	Sensor	26AWG	1	$7.99	$7.99	Amazon
Air quality sensor	new gas sensor	Sensor	SEK-SVM30	1	$21.28	$21.28	Digikey
Pimoroni smoke sensor	Smoke sensor BO	sensor	PIM438	2	$16.30	$32.60	Mouser
Samsung 18650's	4, 18640 Cells	Power	B08C3XGWG8	1	$28.68	$28.68	Amazon
m2 Standoffs (optional	Litorange 320PCS M2 Male Female Nylon Hex	Sensor	31161816	1	$13.99	$13.99	Amazon
Raspberry pi camera	LoveRPi 5MP Camera Module	camera	B07KF7GWJL	1	$9.99	$9.99	Amazon
Adafruit BME680	New gas sensor	sensor	BME680	1	$22	$22.00	Mouser
LM2596	DC-DC converter	power	LM2596	1	$11.99	$11.99	Amazon
128 microsd card	SD card	Communication	MB-ME128HA	1	$19.00	$19.00	Amazon
DFRobot Accessories Gravity4Pin IIC/I2C	Gas sensor connector	Sensor	FIT0513	1	$6.00	$6.00	Mouser

UART SNSR Cables							

6.5. Testing Environment

There were three conditions to test the prototype in. The first one is direct line of sight testing which means it must have large empty spaces half a mile to a mile long for the best-case scenario. The following environments are potential testing environments.

Remote at home testing:
Prior to selecting the final components used for the final prototype, the main components for each subsystem was purchased to test before determining its suitability. This included each peer purchasing a raspberry pi, SD card, DC power source as well as the components each peer was responsible for in the project. This process also included downloading any relevant software applications and becoming familiar with the chosen programming language, python.

During at home testing, the peer attempted to test their components in a controlled environment. The sensors were tested using a lighter to test for flame and gas. Smoke signals will be tested by burning wood and placing the sensors near the fire. This will also be necessary for calibrating certain sensors. The camera will be tested in front of fire and non-fire conditions to save images that will be used for computer vision. This dataset will be used to train a model that can attempt to identify a fire. The LoRa module was tested by attempting to establish communication with at least two devices to see if data can be sent and received. The solar panel system was tested at home to observe and understand its ideal positioning for maximum sunlight absorption.

On campus testing - UCF Arboretum:
The University of Central Florida has an arboretum that acts as a creative learning environment. The arboretum includes a 5-acre Cypress dome, an oak hammock of 3-acres, and 15 acres of sand pine and Florida scrub connected to the original Arboretum by the saw palmetto community and the longleaf pine flatwoods. Currently the entire area of the arboretum includes 82 acres [48]. The UCF arboretum has the landscape and environment for potential forest fires. Thus, it would be ideal to create a controlled fire in this space and determine if the sensors are able to detect fires. Moreover, testing in this environment will allow for experimentation with various mechanical designs and understand which design is best suited for this project. In addition, this will allow the engineers to understand where is the best placement of the devices on the tress: how close to the earth can the sensors be placed in order for it to be close enough to detect the gas, fire, and smoke without interrupting the natural environment and wildlife. Lastly, the engineers wanted to test the range of the devices using the LoRa module at 10m,

50m, 100m, and 150m apart to observe if the communication and data transmission is still maintained. Moreover, the engineers wanted to investigate how close the sensors should be for effective fire detection.

Testing in this area will require permission from the UCF college of engineering department and the UCF facilities and safety department. In the event the project is ahead of schedule, the engineers were open to the possibility of testing the F.I.R.E device in a controlled fire that is routinely done by UCF Facilities and Safety team as a prevention mechanism for forest fires. The UCF arboretum would have been the ideal testing environment since it is used by students from other colleges for educational purposes.

Due to COVID-19 and social distancing requirements, the UCF campus was closed and the engineering working in this project could not access the arboretum to test the prototype. Moreover, the project faced delays to due to lack of equipment and resources, since the UCF Senior Design labs were closed and not accessible. As a result, this project was tested outdoors near a barbeque grill where a controlled fire was made to test the prototype.

Figure 72: Controlled fire at the UCF Arboretum [48]

7. System Integration

7.1. System Design

An overall glance at the system shows the solar array hooked up to an ‘all in one’ LT3652 IC which will output 16.6 volts and a max of 2 amps with the input from the array. That

was then feed into a set of Li-ion batteries. That battery was then, with help from excess power from the solar panel, inputted into two buck converters to make 5 volt and 3.3-volt rails. These two rails handled power to the entire system as some components require specific voltages. The 5-volt rail ran the LoRa module and the Raspberry Pi and this rail pulled the most power out of the system due to how much power a Raspberry Pi requires. The 3.3-volt rail handled powering all the sensors and pulled less power.

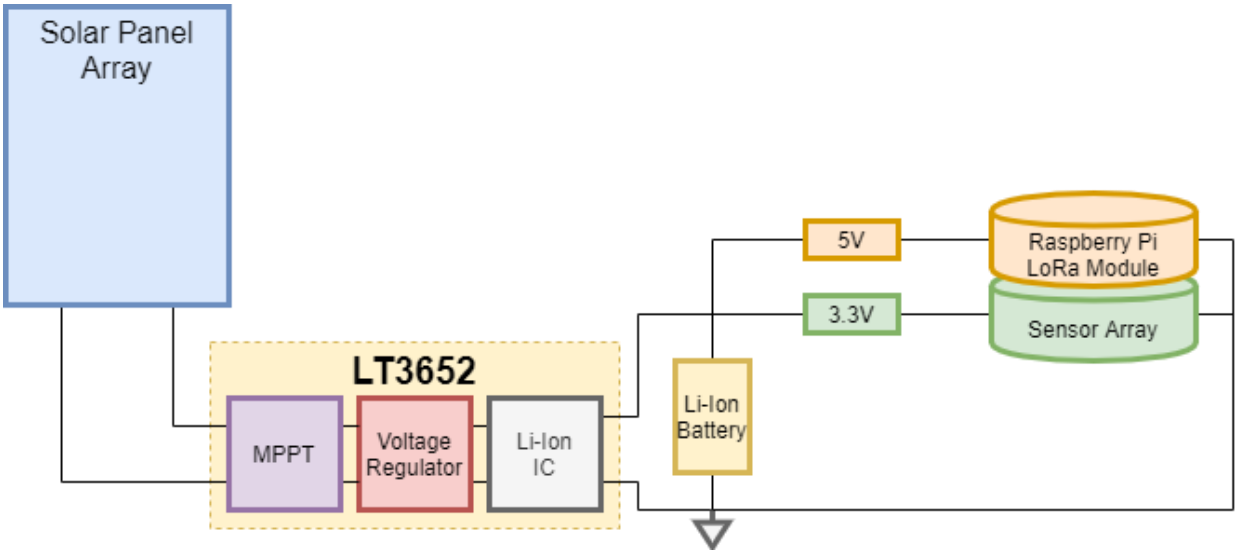


Figure 73: High Level System View

7.1.1. Sub-System Connections

Each system must have all the connections necessary to communicate properly between each other but inside each system there are smaller components that need their own source of power or need to be connected to the same node as another system for grounding or the correct resistive purposes. To make sure everything is wired correctly the sub-systems were designed using KiCAD's hierarchical sheet system. Each component was individually designed and linked together using this hierarchy structure, so the overall design did not get to cluttered or large to view and edit. Doing the designs like this also helped with making sure everything was wired correctly in the final assembly of the system.

7.2. System Operation

The system operates in a cyclic fashion turning on to cycle through all the sensors to check if there is a fire under certain conditions. This is depicted in the diagram below. It then processes that information using the computer vison on the Raspberry Pi and state-space to store and check if the other sensors are within their constraints. If everything is determined to be fine, then the machine will go back into a sleep mode and wait for its timer to turn itself back on and run through the same process. If the system runs through all its checks and determines that there is a fire the LoRa module will then be booted up

and that data will be transmitted in a mesh network until it gets to an operator who is monitoring the overall system. That operator can then check what determined that fire and decide with human intervention if it is a false alarm or if they need to respond by the measures deemed necessary.

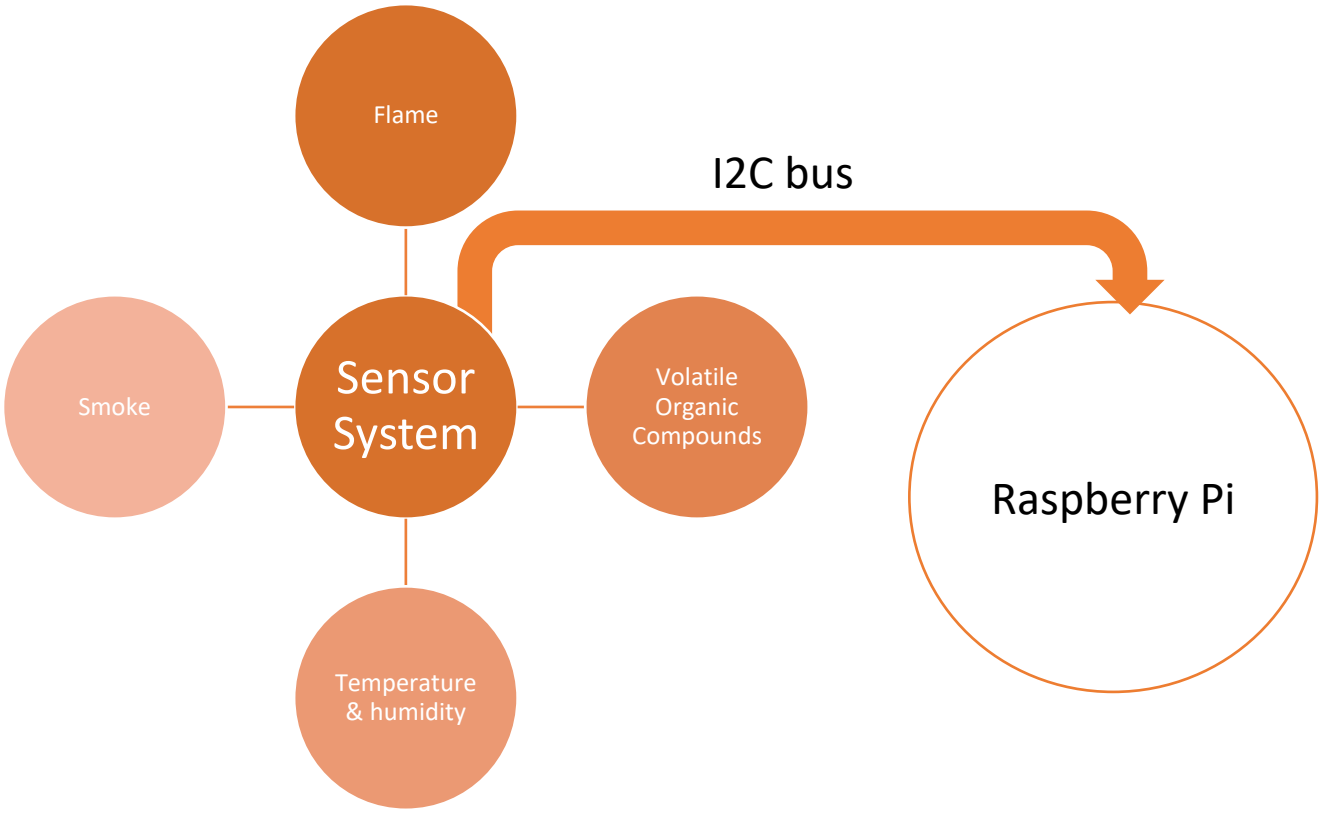


Figure 74: Sensors data sent to processor

7.2.1. Power system

For the initial setup of the power system all that must be done is connecting the solar panel array to the board. Next is to connect the two 18650 li-ion cells to the board by sliding them into the holder making sure to pay attention to the polarity of the cells. After that the system will automatically turn on using the relay. It is really that simple just make sure to pay attention to the solar panel and cell polarity.

7.2.2. RF system

The RF system operates on the Raspberry Pi. Running the python script starts up the software. There is no interaction needed after that point.

To ensure that the mesh network can be joined correctly, connect an SX127x module to the Raspberry Pi through the following connections (each pin is the BCM pin, not board pins):

MOSI to 10	**MISO to 9**	**SCK to 11**	**NSS to 8**	**DIO0 to 4**
DIO1 to 17	**DIO2 to 18**	**DIO3 to 27**	**RST to 22**	**LED to 13** **(Optional)**

The software is no ready to use on a sensing node. The root node needs a similar setup but different software. The software files are provided. The root node need only respond correctly to join requests for the scope of the prototype. See the included software written in Python as an example.

7.2.3. Sensor system

The following instructions explain how to connect the sensors to the raspberry pi.
Prior to running any script, it is important to make sure that the I2C bus is enabled in the raspberry pi (rasp-config), and that the i2c-tools package is installed in the raspberry pi.

Once this is complete the user can proceed to connect each sensor individually first.

Flame:
The flame sensor requires 5 connections to the raspberry pi explained in the table below:

Flame sensor pin	Raspberry Pi pin
Vsupply	3.3V
GND	GND
CS	3.3V
SCL	SCL
SDA	SDA

Once the connections are made, the user can run the i2cdetect -y 1 command on terminal/command window and must see the slave address, 0x65. Then the user can run the startflame.py python script to initialize the sensor, then the flame.py python script to begin reading data and detecting flame.

Smoke:
The smoke sensor requires 4 connections to the raspberry pi:

Smoke sensor	Raspberry Pi pin
Vsupply	3.3V
GND	GND
SCL	SCL
SDA	SDA
INT	GPCLCK0

The INT pin is an optional connection.

Once the connections are made, the user can run the i2cdetect -y 1 command on terminal/command window and must see the slave address, 0x57. Then, the user can run the smoke.py script to initialize and begin reading data from the smoke sensor.

Gas:
The gas sensor requires 4 connections to the raspberry pi:

Smoke sensor	Raspberry Pi pin
Vsupply	3.3V
GND	GND
SCL	SCL
SDA	SDA

Once the connections are made, the user can run the i2cdetect -y 1 command on terminal/command window and must see the slave address, 0x77. Then, the user can run the gas.py script to initialize and begin reading data from the smoke sensor

Ensure that the data read from all the sensors are saved in a shared file. This is for the system to integrate the data with the computer vision system to output an overall confidence score in detecting fire.

7.2.4. Computer Vision System

The computer vision system uses Python 2.7 with Keras 2.1.5, TensorFlow 1.8.0, and latest OpenCV compatible with Python 2.7 for Raspberry Pi while Python 3 with Keras 2.1.5 and TensorFlow 1.8.0 is used for Google Colab deep learning. Installations in Raspberry Pi Zero may take several hours which is normal. The correct versions are crucial in making the computer vision system work.

Camera should be connected to Raspberry Pi and operable through commands and Python file.

As long as the version matches, the trained model should be operable in Raspberry Pi Zero but not for real-time use. It is best for Raspberry Pi to access images saved in a folder as it has limited processing power.

The testing of the computer vision system is explained in the section 6.3.3.

8. Administrative Content

The following section is a discussion of the administrative content that comes from a project. Included will be the division of work, milestones and timelines, information about our sponsor work for Siemens and our cost.

8.1. Division of Labor

The project was divided into multi sub-parts for each team member to work on and specialize. Each subject is not mutually exclusive however, as the team is expected to help the others in their designs and research. **Table 13** goes into what each team member was assigned to accomplish as well as a list that breaks down a little more about what that team member is doing specifically. **Table 14** provides a detailed description.

Table 13: Division of Labor

	Area	Focus
Noora	Sensors	Hardware & Software
Nicholas	Power & Mechanical	Hardware
Jonathan	Control & RF	Hardware & Software
Arisa	Data Processing	Software

Table 14: Division of Labor Breakdown

Engineering student	System components	Description
Noora	Flame Sensors Smoke Sensors. Gas sensors Temperature/Humidity sensors Data processing	Noora's focus was on designing the printed circuit board which will include selecting appropriate sensors components and ensuring these sensors are able to detect fire elements such as flame, smoke, and volatile organic compounds, as well as communicate with the raspberry pi. Noora worked closely with Arisa to send values used in the confidence interval after data processing.
Nicholas	Solar Panel Power Battery Charging Protection. Power Regulation	Nicholas was responsible for designing printed circuit board that will be used for supplying power to the entire system. The system will be powered by a solar panel system. Nicholas chose appropriate solar panels that will efficiently supply enough power to the system; this included selecting the right type, model, and size of panels. Subsystems will need a 3.3V and 5V supply, thus Nicholas designed the PCB with appropriate regulators and rails to ensure the components received ample and stable power supply.

Jonathan	Microcontroller Design Network Software RF Design	Jonathan was working on the communication between the devices. This was done through radio frequency using the LoRa RF module. Jonathan designed the system to ensure communication between the devices is maintained and data from sensors can be sent to the hub using RF waves. Jonathan tested the range of the system for which data can be sent and design the communication network of the system.
Arisa	Raspberry Pi Camera, Sensor Data Processing Software Machine Learning' Raspberry Pi	Arisa was responsible for designing the machine learning algorithm that took data from the raspberry pi camera and the sensors. By recording and analyzing the processed data from the sensors, Arisa trained the machine to recognize fire and non-fire conditions using an algorithm. This algorithm will determine forest fire conditions and recognize all characteristics of fire such as flame, smoke, and VOC gasses and visual aspects from the camera. The signal will be sent to the microcontroller to trigger the fire warning.

8.2. Project Milestones

The following two tables provide timeline of the project in Senior Design 1 and 2. Table 9 was timeline for the spring 2020 semester. During this semester, the background research on the project's need, standards, requirements, hardware components, mechanisms of detection, testing environment, sponsor requirements, logistics, feasibility is examined through the senior design 1 report. Each peer's strengths and weaknesses were identified to be understand how each individual can contribute to the project. Moreover, areas that can be challenged and improved were also identified. Once this was recognized, assigning the project tasks and requirement to each peer became a natural and organic process. By the end of the spring semester, the group and the sponsor had a reasonable understanding of the project's scope so that the prototyping stage can begin.

Table 15: Spring 2020 Milestones

Week	Milestone (Tasks)	Start Date	Deadline
1 to 2	Brainstorm ideas	January 06, 2020	January 17, 2020
3 to 4	Choose a project and discuss basic design and roles	January 20, 2020	January 31, 2020
4	Finish Divide and Conquer V1		January 31, 2020
5	Discuss the details of the project	February 03, 2020	February 07, 2020
5 to 6	Update Divide and Conquer V2 Finish proposal for sponsor	February 03, 2020	February 14, 2020
6 to 9	Research and fine-tune design	February 17, 2020	March 06,2020
9	SPRING BREAK – COVID 19 closure	March 09, 2020	March 13, 2020
10	60-page Draft		March 20, 2020
10 to 12	Finalize design Finish technical documentation	March 16, 2020	April 03, 2020
12	100-page Report		April 03, 2020
12 to 15	Organize all documentations	April 06, 2020	April 17, 2020
15	Submit Final Documentation		April 21, 2020

Table 10 below shows the timeline for the summer 2020 semester. Summer was an accelerated semester with 4 weeks less than the other semesters. This leaves less opportunity for errors. As a result, after completing a robust report, the summer semester marked the beginning of the projects prototyping, testing, and implementation stage. The aim was to complete fulfilling the project's intended purpose by July 20th. This final product, as well as electrical and CAD designs were handed off to the sponsor. Moreover, the senior design 1 paper will be modified to reflect the project's realistic achievements.

Due to COVID-19, several delays occurred in the project as a result of social distancing and restricted access to the UCF laboratories. Thus, this schedule includes the unexpected delays that occurred during the summer 2020 semester.

Table 16: Summer 2020 Milestones

Week	Milestone (Tasks)	Start Date	Deadline
1 to 3	Acquire initial components and conduct initial testing.	May 11, 2020	May 29, 2020
4	Critical Design Review Presentation preparation and video editing	June 01, 2020	June 05, 2020
5	Complete KiCad PCB Schematics and submit order. Order additional parts and replacement parts	June 08, 2020	June 12, 2020
6	Hardware and software developments	June 15, 2020	June 19, 2020
7	Midterm video recording and editing	June 22, 2020	June 26, 2020
8	Midterm demo presentation and meeting	June 29, 2020	July 3, 2020
9	Soldering and integration	July 6, 2020	July 10, 2020
10	Finalize product	July 13, 2020	July 17, 2020
11	Finalize documentation and handover to sponsor	July 20, 2020	July 31, 2020

8.3. Sponsor Information

8.3.1. Siemens Foundation

Siemens Foundation was founded in 1998 as a non-profit organization in the United States [49]. The foundation has invested more than $122 million in the United States to foster an inclusive and innovative culture through a variety of professional developments programs for the Siemens workforce, STEM outreach activities for youth, and scholarships for future students [49].

The most notable program is Siemens STEM day which initially was an event dedicated to engaging K-12 students in a variety of hands-on activities through experiments and problem-solving exercises. Currently the program has expanded past a one-time event to a portal that provides employees access to over 150 STEM activities allowing Siemens volunteers to facilitate STEM day activities any time of the year in addition to STEM day [49]. These activities range from easy to difficult and revolve around themes popular in the industry. The STEM kits target students of all ages, however, there is currently a demand for activities that target older students to emphasize various applications of scientific knowledge in real life, especially in disciplines that are needed in the US. Facilitating these activities is important when considering the demand for STEM professionals and closing the opportunity gap for the youth.

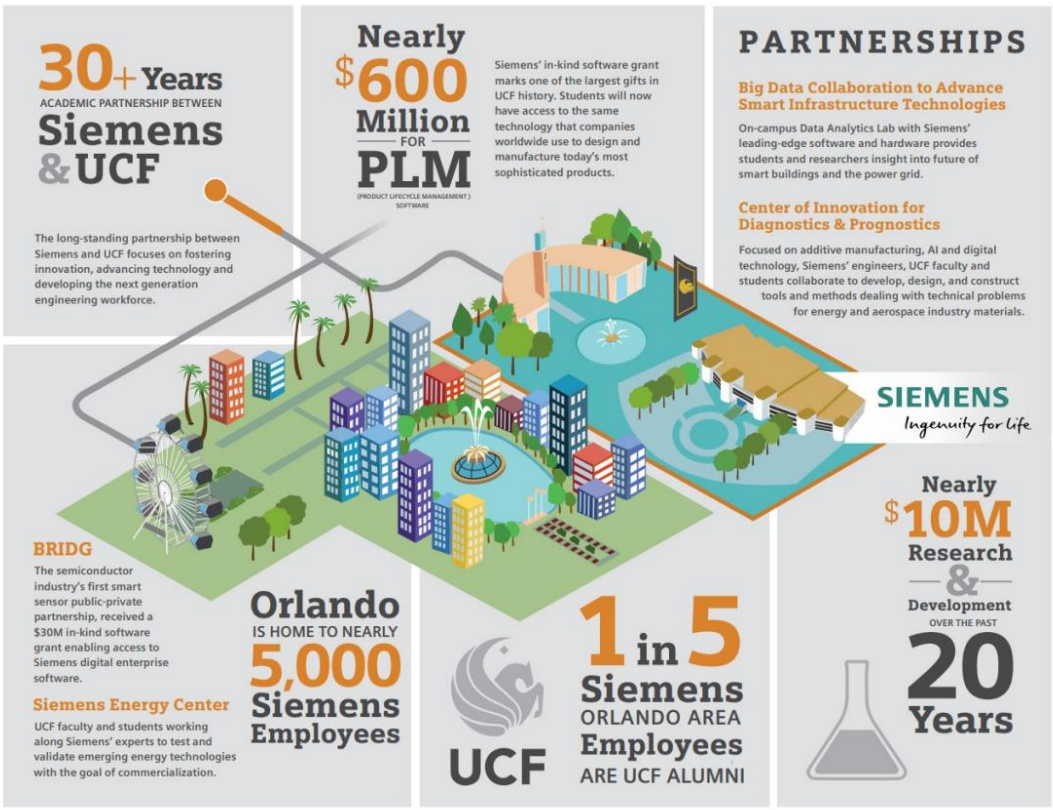


Figure 75: 30+ Years of Academic Partnership Between Siemens & UCF to foster the goals of Siemens Foundation [49]

8.3.2. A Product for Siemens STEM Initiative

Ultimately, the F.I.R.E device will not only serve its purpose of forest fire and detection and monitoring, but also will be meticulously designed keeping in mind that the product will serve as an introduction to electrical engineering kit. Through this kit, students will become exposed to sensor technology, programming and communication through mesh network, and an optional hands-on experience soldering parts to a printed circuit board. The importance of engaging the youth in STEM related activities has gained traction due to the decline in the overall number of students pursuing STEM fields. Thus, exposing STEM opportunities to young students, especially to students from marginalized groups, is important in encouraging and fostering a culture of innovation, research, and diversity. SIEMENS' STEM initiative is founded on these values. Thus, this product will be designed to be used in SIEMENS STEM Day activities to expose students to fundamental concepts of electrical engineering and importance of environmental consciousness.

The product is aimed to be utilized as an advanced activity for students ideally between the 9 – 12th grade that are in the early stages of exploring and deciding career options to pursue after completing high school. This project will help introduce and educate students on a leading environmental issue, forest fires, while also demonstrating how electrical engineering concepts can be used to solve a growing environmental concern. Moreover,

students will also learn about the fire and gas sensors that are used for SIEMENS gas turbines and how their function compares with the sensors designed in the kit. Overall, students will gain an understanding of how the system was designed, and how it can be implemented. This learning kit will also be a unique exposure to engineering project management and execution.

The objectives of the activity are detailed below:

1. Understanding forest fires, their growing intensity, and how fire emissions are shaping climate change.
2. Solving this issue by providing proactive solutions to mitigate the risks.
3. Understanding the technology used to tackle the issue:
    a. Flame detection (visual and non-visual techniques)
    b. Gas detection
    c. Smoke detection through photoelectric sensors
4. [Optional] Soldering basic components to a printed circuit board.
5. Straightforward programming exercise understanding how values are read and communicated in a network.
6. Testing the device and witnessing how it can react to a fire.

At the completion of the project, the final product will be delivered to SIEMENS' STEM initiative group with a detailed lesson activity guide for Siemens employers to use for STEM day activities. In addition, the printed circuit board schematic and design, as well as any CAD design, will also be provided so that additional boards can be produced for enhanced learning activity that incorporates soldering components to the printed circuit board.

8.3.3. Connection to the Siemens industry

A significant portion of the project's requirement and the sponsorship from Siemens is not only supporting our aspirations of designing this system but also emphasizing how the product connects to the Siemens industry in terms of the similarities in the technology and strategies used, as well as the potential opportunity for Siemens to utilize this product in their industry.

Siemens AG headquarter is in Munich, Germany [49]. It is a multinational conglomerate and considered to be one of the largest industrial manufacturing companies in Europe. The main industries it is involved in are: Energy, Healthcare, and Infrastructure. The Siemens offices in Orlando, FL are primarily focused on power generation, energy efficient buildings and infrastructure, wind energy, and healthcare [49]. Its proximity to the University of Central Florida has enabled a partnership allowing for $10 million in investment for research projects at the university such the Digital Grid Innovation Laboratory, Center of Innovation for Diagnostics & Prognostics, and the Siemens Energy Center [49].

8.3.3.1. Gas Turbine

Siemens' gas turbine manufacturing and commissioning is one of the dominating businesses in Orlando, FL. Siemens gas turbines range from 4 – 593 MW and are used for a variety of applications including power generation for utilities, independent power producers, oil and gas as well as industrial users such as chemicals, pulp and paper, food and beverage, sugar, automotive, metal working, mining, cement, wood processing, and

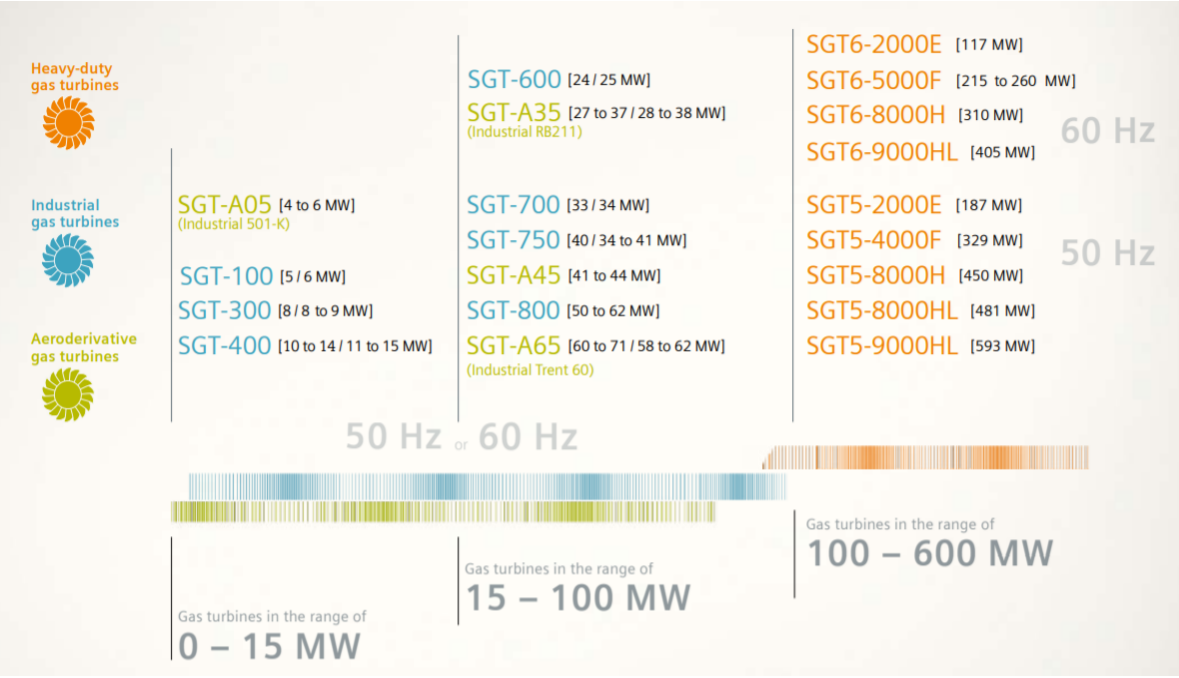


Figure 76: Overview of Siemens gas turbines [55]

textiles. Siemens gas turbines fall into one of three categories: heavy-duty, industrial, or aeroderivative [50, 51].

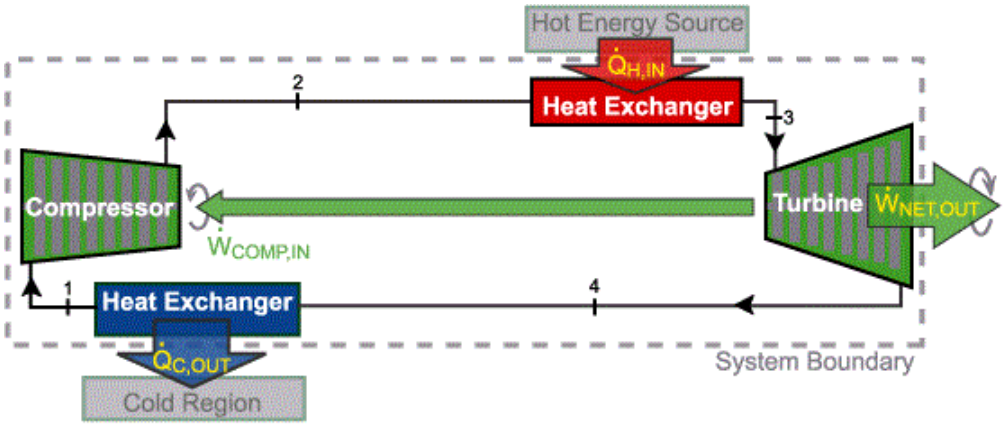


Figure 77: Typical gas turbine cycle as stated in [52], the figure shows where a fire and gas sensor would be needed.

The primary components of the gas turbine using the Brayton cycle is a compressor, combustion chamber, gas turbine, and generator as depicted below. Siemens gas turbine control system includes a variety of instruments used to measure the gas turbines temperature, pressure, speed, and vibration. The main interest for this project will be the temperature sensing of the system for fire and smoke detection. Current temperature sensing for the gas turbines includes a gas thermocouple and the infrared temperature sensor [52, 53].

A thermocouple is composed of two dissimilar metals connected together creating a junction through welding. [52] One end of the connection is taken for reference and other end of the junction is used for measurement [52]. Temperature measurement is possible when there is temperature difference between the two junctions; this causes an electric current to flow in the circuit [52]. By understanding voltage-temperature relationships of metal combination, the temperature can be measured [52]. There are many types of thermocouples; however, type K thermocouple is commonly used in gas turbines. Siemens SGT-A05 KC uses the Measured Gas Temperature (MGT) thermocouple to extend the in-service life of the turbine and it is also used in 180 other engines Pictured below is the MGT thermocouple [54]. The SGT-A05 KB/KH also uses the TOT Thermocouple or the TIT thermocouple to improve overall accuracy in temperature monitoring [54].

Figure 78: Thermocouple used in Siemens SGT [54]

Infrared temperature sensors are a good option to use to minimize the contact between the sensor and the object it is measuring, which for gas turbines is the blade. Infrared sensors function by “focusing the object’s infrared energy onto photodetectors” [52].
This provides an electrical output signal that is proportional the infrared energy received. The infrared energy emits varying levels of infrared energy to the object according the temperature which allows for an accurate description of the object’s temperature [52]. Siemens SGT-750 uses infrared cameras to measure the temperature of the blade surface [52]. Temperature is recorded each rotation and is used for the cooling system [52]. Below is an image of the infrared temperature sensor used in the SGT-750.

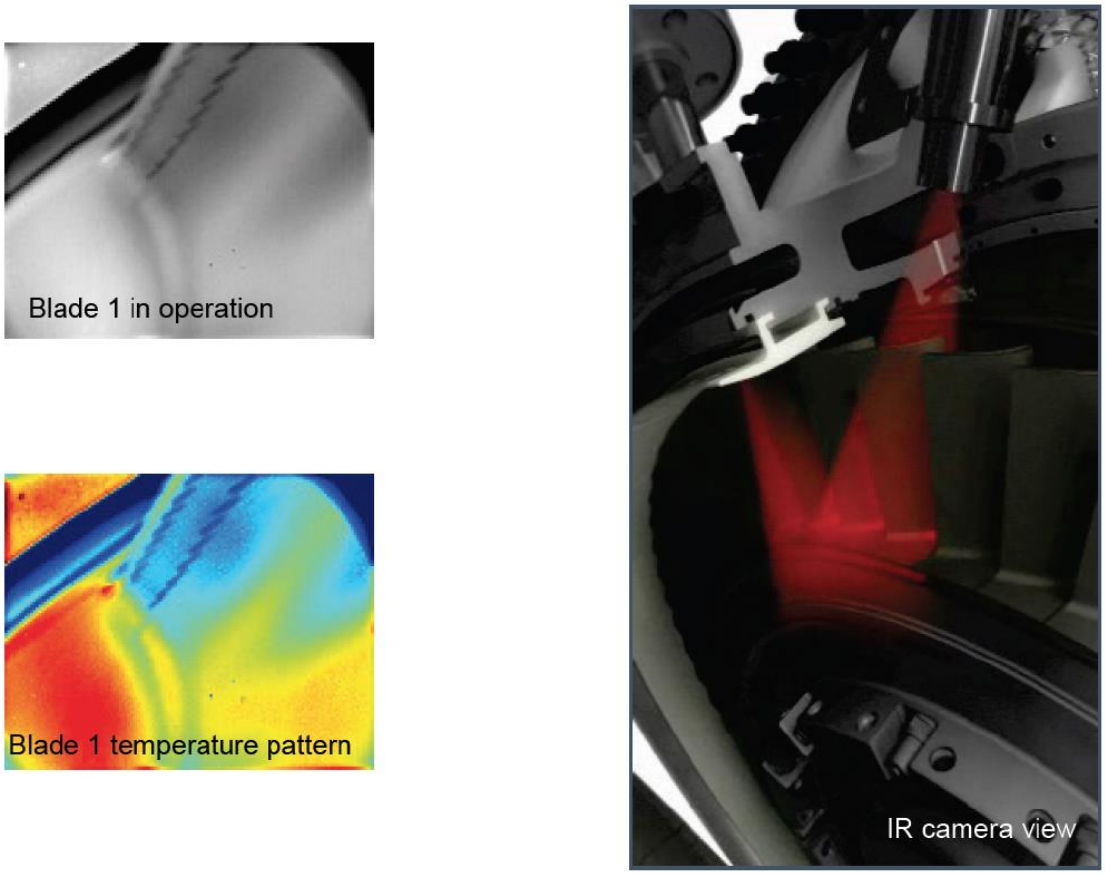


Figure 79: Infrared temperature sensor used in the SGT-750 [52].

Project F.I.R.E utilizes similar techniques used in the Siemens gas turbine for fire and smoke detection. Siemens uses a thermocouple which is a typical choice for a higher scale, range, and accuracy for heavy industrial applications. In our project, a temperature sensor IC will be utilized since it will help drive the cost and size down for forest fire applications. The IR sensor is comparable to the flame detection technique F.I.R.E as it involves detecting hidden infrared rays to measure thermal heat of the gas turbine blades.

8.3.3.2. Digitalization/Internet of Things

A growing field in the industry is the digitalization of many products results in a demand for the Internet of Things (IoT). Siemens offers IoT services ranging from Consulting, Solution Design, and Solution Development and Implementation which all includes Change Management and Cyber Security [55]. There are five phases that Siemens uses for successful IoT implementation detailed below in the diagram [55].

IoT as discussed previously in this paper has the potential to digitalize many industries

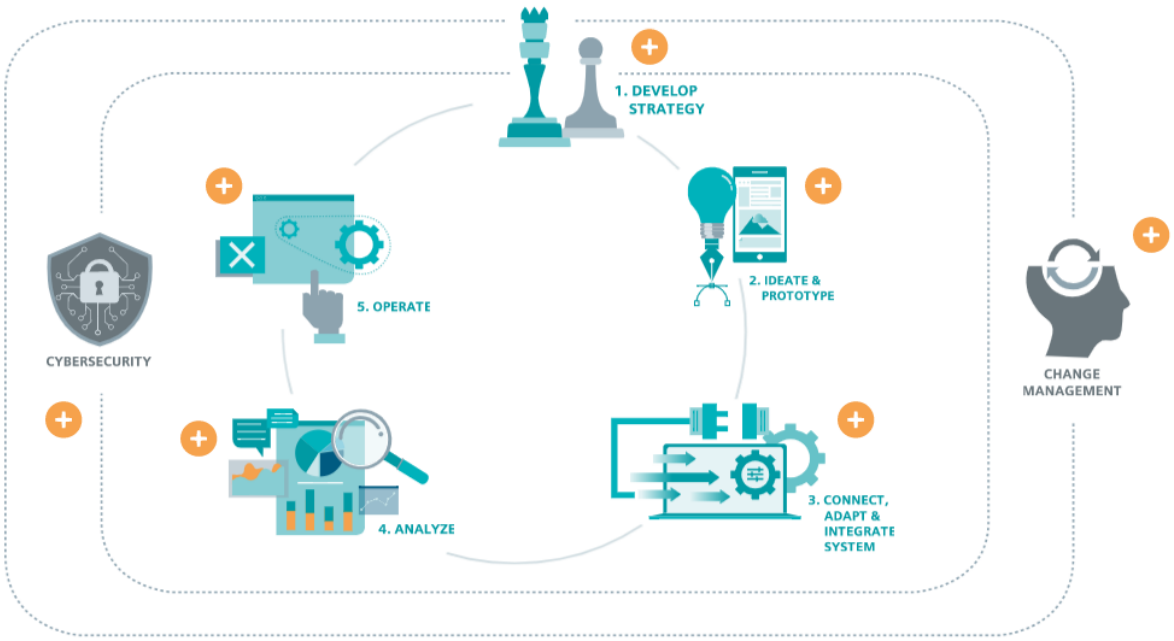


Figure 80: IoT integration cycle developed by Siemens

including manufacturing, energy utilities, healthcare, transportation and building technologies, which are the industries Siemens is mostly tied to. Before the users can benefit from the insights of IoT, data must be collected and sent through a gateway data communication [55]. The data is then transferred and stored where it can be used to conduct data analytics and conduct machine learning algorithms [55]. From here, it can be used to provide insight for efficiency and create better business models [55].

Siemens has been heavily involved in IoT as the possibilities of improving business and performance for the industries it is involved in are endless. For example, Siemens was involved in an air quality monitoring system in the city of Nuremberg. Nuremberg city officials were concerned about the air pollution as a result of increased traffic which made it difficult for the city to maintain recommended levels of nitrogen dioxide set by the World Health Organization [55]. Siemens set up an IoT system that allowed it to collect data such as air pollution levels, weather, and traffic patterns from sensors placed around the city [55]. This data is then used to forecast the city's air quality for the next 5 days [55]. With this data, the city is able to take appropriate measures to reduce air pollution levels. The Siemens City Air Management and the City performance Tool is also able to conduction simulations and make long term predictions factoring various parameters such

as environmental legislature and new technology; they are now able to make predictions until the year 2030 with remarkable accuracy [55].

Another case were Siemens was able to utilize IoT was in the case of the Sello shopping mall in Finland [55]. The shopping mall wanted to increase its energy efficiency since it accommodates more than 24 million shopper every year. Siemens engineers turned the mall into a “virtual power plant” and it was able to operate as a load for the Finnish demand response markets [55]. 2 MW batteries were installed with a solar panel system that included a microgrid with smart building automation and cloud analytics [55]. The process took a few years using an iterative approach and followed the five phases depicted in the diagram [55]. Sensors were installed in the building management system that measured weather data, energy consumption, energy price, weather forecast data and the amount of energy stored in the battery [55]. By using smart analytics, an algorithm was designed to determine whether energy should be drawn from the solar panels, the 2-MW battery (stored energy), or the national energy provider when electricity rates are low [55]. This implementation helped reduce carbon emissions and saved the business €643,000 ($690,00) [55].

8.3.3.2.1. Siemens IoT implementation phases in F.I.R.E.

Interestingly, this project will adopt similar phases during its life cycle, which is an important connection this project has to Siemens’ current IoT practices. In the initial phase, this project underwent strategy development where the best method of fire detections was investigated. This included identifying mechanisms and principles of detection that are used in the industry. The challenges were also explored, such as range and scalability during this phase. Most importantly was also determining how this project provides a value not only for us, but also Siemens and how this project aligns with the ambitions of the Siemens Foundation and the Siemens’s industry goals.

Once the idea was established, the technical implementation next stage is followed. As mentioned, Siemens is our customer and they are at the center of our focus. Their requirements are to create a solar powered forest fire detection and monitoring system that will also be used as STEM kit to educate the youth on electrical engineering concepts, and also how the technology and implementation relates to the industry. Another crucial requirement is heeding their budget requirement of approximately $500. Furthermore, the university (UCF) is also our customer because they are expecting a senior design project that fulfils the criteria set by Accreditation Board of Engineering and Technology. Lastly, this product has potential to be used in the industry, therefore government of countries experiencing forest fires as well as the authorities that protect reservations that are likely to experience forest fires are our unheard audience; we are not able to interact with them directly, however we have built our assumptions based on their experiences and the technologies they have used for forest fire detection and past research.

By integrating these three audiences’ concerns, demands, and needs as well as our own skills set and experience, we are able to identify a reasonably sound solution and initiate the first prototype. The protype will be used to gather as much data possible; in this

project's case once the sensors have been selected and are fully functional, the sensors will begin accumulating temperature, humidity, pressure, gas concentration levels, smoke and flame conditions. This historic data will be useful for mathematical and statistical methods to determine an algorithm than assess various parameters and identify similar patterns in the data set. Machine learning will be used to train the model and improve prediction outcomes.

The third stage involved connecting, adapting, and integrating systems. The main components in this process include the sensors, communication networks, cloud infrastructure and IoT platforms and applications. In this process the data gathered from the sensors can be sent to other devices and the main hub which will house all the database. The communication protocol becomes vital as it determines range, latency, data volume, and transmission frequency. The F.I.R.E system uses RF communication from the LoRa module which accounts for each of these factors. The database has yet to be established for this project however the two options will mostly like be either premise-based or cloud-based. Communication is vital however it is also important that the data from the various sources are in a uniform language in order for it to be processed to a device or cloud. Once the machine learning algorithms are able to model and predict the data, the outcome will be presented in a visually clear manner for the user to understand.

The fourth stage used in Siemens IoT implementation which will be followed in project F.I.R.E is analyzing the data. As mentioned, the data needs to be easy to read and understand so that appropriate action can be taken with the information provided. In this stage it is important to differentiate between correlation and causality. Correlation is a statistical measure to observe the relationship between two variables; the relationship can be random without grounds for a direct cause. As a result, correlation can produce noise in the data which can lead to less accurate predictions and outcomes. Causality is a relationship that describes the cause-effect connection. Therefore, during this stage is important to interpret the data logically to avoid misrepresentation and to continually train the system to improve and optimize models to avoid false-positive outcomes of a fire.

The final stage is operation. Once the system is operating successfully, it will be important to maintain it regularly to avoid malfunctions. With respect to the F.I.R.E project, this device will be handed off to Siemens to use for future STEM events potentially manufacture more STEM kits in the future. To ensure proper maintenance and use is observed, a guide will be provided with the step-by-step procedure of operating the system and testing it under various conditions. This guide will be a combination of written material and video tutorials to ensure it can be properly understood and avoid vague rhetoric. It will be targeted towards Siemens engineers who will be conducting the activities and will be responsible for maintaining the system's operational standards.

8.3.3.3. Siemens Gamesa: Wind Turbines

In 2016, Siemens announced it would merge it wind businesses with Gamesa with a 59%-41% split between the two shareholders [56]. Siemens Gamesa is one of the leading manufacturers and suppliers in the world for wind turbines. Siemens Gamesa have installed wind turbine technology in over 90 countries with base capacities exceeding 99GW [56]. Siemens Gamesa's businesses is primarily focused in onshore and offshore wind turbines and service maintenance. They are situated globally and also have an office in Orlando, FL.

Like gas turbines, wind turbines need to be maintained and protected to ensure optimal performance. Gas turbines are more likely to catch fire because the nature of the fuel is highly flammable [57]. With wind turbines, although it is not powered by a flammable source, the wind turbine system still needs to be designed with a fire detection system

Figure 81: Nacelle of a wind turbine where the AFFS is installed

since it is designed with various mechanical and electrical components where a potential malfunction could start a fire [57]. Most wind farms in isolated areas and the possibility of a turbine being struck by lightning is also a concern. Earlier in February 2020 there was a turbine rotor that caught fire in a wind farm in northeastern Brazil; the turbine was a 2MW G97 Siemens Gamesa turbine [58]. Similarly, a G80 2MW wind turbine caught fire in Japan in 2017 [59]. The issue with fires in wind turbines is they become difficult to save the turbine once it catches fire, especially if the source of the fire is in the nacelle as shown in the figure [57]. Repair costs are very high and put technicians who must conduct the offshore repairs at risk of injury or death [57]. Most wind turbines include fire-protection products which include circuit breakers, semiconductor protection fuses, differential current monitoring devices, measuring instrumented for power monitoring, residual-current devices, and busbars [57]. Graduated protection is also an additional measure

taken to avoid turbine failures; this includes disconnecting defective systems from the grid earlier on to avoid a fire from igniting [57].

In 2014, Siemens Building Technologies Division announced it developed automatic fire-extinguishing system for off-shore turbines and the new system would be installed at Riffgat project in the German North Sea [60]. The Active Fire Fighting System (AFFS) works by detecting fires by reading sensor signals from the Advanced Signal Analysis (ASA) fire detectors to alert the system of a fire in a nacelle or tower [60]. The system then activates nitrogen gas to extinguish the fires, operating on principles of oxygen displacement, using the Sinorix gas fire extinguishing system [60]. The turbine is shut down until the fire is extinguished. An advantage to this system is that it does not produce false alarms and low maintenance and resistant [60]. The added extinguishing feature prevents the fire from spreading nearby and reduces the need for fire helicopters [57]. Moreover, the operators can remotely access the system and identify the source of the fire from the control station which will allow turbines to resume activity as soon as possible. For added safety two AFFS systems are installed in a turbine: in the nacelle and in the tower, both operate independently in the event of a power failure or network outage. Currently, the AFFS system is in operation in 30 wind turbines [60]. Siemens was recognized as the first company to test and approve a fire detection and extinguishing system for wind turbine equipment; it has been certified by VdS Schadenverhütung GmbH and approved by Germanischer Lloyd [60].

Figure 82: ASA fire detectors by Siemens

Figure 83:Sinorix fire extinguisher used by Siemens

Siemens AFFS fire detection and prevention system holds many similarities to our device. The system uses a similar technique of installing sensors that read and process data of the current conditions and an algorithm is then used to identify probably cases of a fire. One distinguishing feature that the AFFS device has is that it is paired with an extinguishing feature for swift prevention of the fire spreading [60]. This feature was a potential feature we had also considered but it was ruled out on the basis that the

extinguishing gasses could harm the wildlife, animals, and the forest environment. Thus, it was decided that extinguishing the fire was outside the scope of this project and could perhaps be further researched using drones. However, this difference is mainly attributed to the fact that the intended purpose of the AFFS system is for wind turbines that typically located in remote areas. This simply establishes the importance of recognizing the planned purpose of the product and how it is integrated during the design and prototyping process.

8.4. Estimated Cost

The table below, **Table 17,** provides the estimated list of costs associated with the project. A major target of the project is delivering a system that is cost effective while maintaining product performance. Based on preliminary research and experience, an estimated cost breakdown was prepared. The data included in the table is a rough cost estimate on all items that were purchased to build the prototype. Initially, the supposed was supposed to be composed of 3 to 4 devices that will communicate data with each other. However, due to cost constraints and limited resources and equipment from the COVID-19 lockdown, only one prototype could be built. Thus, the cost below illustrates the total cost of designing and implementing one prototype.

The table acts as a guide to see the general cost for the system. Cost is determined by the distributor price when purchasing a single item, not in bulk. As we progressed further into the project, potential areas to cut cost became apparent through careful research, design, and testing.

Table 17: Estimated Cost

Item	Estimated Cost ($)
Solar Panel System	100
Sensors*	
Gas sensors + Temperature + Humidity sensor	22
Flame sensor	41
Particle sensors (smoke detection)	16
Raspberry pi camera	10
Communication system (RF and controllers)	68
Electronics*	
General components (resistor, capacitors, inductors, connectors, jumper wires)	30
PCB manufacturing	75

Shipping costs	40
Backup parts	70
Total Cost**	≈ $472

Appendix A: Sponsor Branding Approval

Dawood, Noora Ali (ext) (SE G SO EN EO INT)

From: Lemme, Cameron (GP EPC SO EN ORL PTEC PEC)
Sent: Friday, April 3, 2020 12:22 PM
To: Dawood, Noora Ali (ext) (GP EPC SO EN EO)
Subject: RE: STEM Senior Design project with UCF EE/CpE students
Attachments: full_color_logo.png

Noora,

You may use the Siemens & the STEM@SIEMENS logo for the purpose of the UCF final report. Please refer to the following website and be sure to follow all formatting guidelines for the logo. The goal is to maintain the proper ratio and color scheme of the logo. Additionally, please ensure that it is clear that this is a Siemens / STEM@Siemens sponsored project and that this is not a Siemens Publication.

https://brandville.siemens.com/en/hub?returnTo=/en/homepage

Thanks,
Cameron

From: Dawood, Noora Ali (ext) (GP EPC SO EN EO) <noora.ali_dawood.ext@siemens.com>
Sent: Friday, April 3, 2020 3:06 PM
To: Lemme, Cameron (GP EPC SO EN ORL PTEC PEC) <cameron.lemme@siemens.com>
Subject: RE: STEM Senior Design project with UCF EE/CpE students

Hi Cameron,

We are in the process of completing our final report for UCF. We would like use Siemens logo and branding in our completed report.

Please let me know if this will be allowed. UCF requires written consent for using the sponsor's logo in our report.

Best,
Noora

Appendix B: References

[1] S. Ouni, Z. T. Ayoub and F. Kamoun, "Auto-organization approach with adaptive frame periods for IEEE 802.15.4/zigbee forest fire detection system.," 16 Jan. 2019. [Online]. Available: https://doi.org/10.1007/s11276-018-01936-x. [Accessed 30 Jan. 2020].

[2] M. Jurvélius, "FOREST FIRES AND INTERNATIONAL ACTION," 2003. [Online]. Available: http://www.fao.org/3/XII/0820-B3.htm. [Accessed 30 Jan. 2020].

[3] A. A. Alkhatib, "Forest Fire Monitoring," 20 Dec. 2017. [Online]. Available: https://www.intechopen.com/books/forest-fire/forest-fire-monitoring. [Accessed 30 Jan. 2020].

[4] United Nations Enviroment Programme, "Governments, smart data and wildfires: where are we at?," 3 Jan. 2020. [Online]. Available: https://www.unenvironment.org/news-and-stories/story/governments-smart-data-and-wildfires-where-are-we. [Accessed 30 Jan. 2020].

[5] ASQ, "House of Quality Tutorial - How to Fill Out a House of Quality | ASQ," 2020. [Online]. Available: https://asq.org/quality-resources/house-of-quality.

[6] RoHS, "RoHS Guide," 2005. [Online]. Available: https://rohsguide.com/rohs-faq.htm. [Accessed 4 Mar. 2020].

[7] IPC-2221, "Generic Standard on Printed Board Design," Feb. 1998. [Online]. Available: http://www.ipc.org/TOC/IPC-2221.pdf. [Accessed 5 March 2020].

[8] Electronic Code of Federal Regulations, "e-CFR," 28 May 1996. [Online]. Available: https://www.ecfr.gov/cgi-bin/text-idx?SID=7248d37fdd25d0947f5611197fd5c6c8&mc=true&node=se47.5.101_1113&rgn=div8. [Accessed 17 March 2020].

[9] K. Nörthemann, J.-E. Bienge, J. Müller and W. Moritz, "Early forest fire detection using low-energy hydrogen sensors," 1 Nov. 2013. [Online]. Available: https://pdfs.semanticscholar.org/b807/d5144095c15f7805fd272cb71a8a023a9516.pdf. [Accessed 30 Jan. 2020].

[10] "Arduino fire alarm system using temperature and smoke sensor with Android connectivity," Microelectronics Technologies, [Online]. Available: https://www.projectsof8051.com/arduino-fire-alarm-system-using-temperature-and-smoke-sensor-with-android-connectivity/. [Accessed 4 Mar. 2020].

[11] Bluetooth, "Understanding Bluetooth Range," 2020. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/range/. [Accessed 30 Jan. 2020].

[12] F. Leens, "An introduction to I2C and SPI protocols," Feb. 2009. [Online]. Available: https://ieeexplore-ieee-org.ezproxy.net.ucf.edu/document/4762946. [Accessed 30 Jan. 2020].

[13] A. Gaur, A. Singh, A. Kumar, K. S. Kulkarni, S. Lala, K. Kapoor, V. Srivastava, A. Kumar and S. C. Mukhopadhyay, "Fire Sensing Technologies: A Review," 1 May

2019. [Online]. Available: https://ieeexplore.ieee.org/document/8625538. [Accessed 19 Mar. 2020].

[14] J. Fonollosa, A. Solorzano and S. Marco, "Chemical Sensor Systems and Associated Algorithms for Fire Detection: A Review," 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/2/553. [Accessed 30 Jan. 2020].

[15] M. Y. J. D. Y. Z. Liqiang Wang, "Hybrid fire detection using hidden Markov model and luminance map," *Computers and Electrical Engineering,* vol. 37, pp. 905-915, 2011.

[16] ScienceDirect, "Kevlar," [Online]. Available: https://www.sciencedirect.com/topics/engineering/kevlar. [Accessed 1 April 2020].

[17] Jamestown Distributors, "Kevlar Cloth - Plain Weave," [Online]. Available: https://www.jamestowndistributors.com/userportal/show_product.do?pid=4022. [Accessed 18 March 2020].

[18] WallpaperAccess, "Carbon Fiber," [Online]. Available: https://wallpaperaccess.com/carbon-fiber. [Accessed 18 March 2020].

[19] J. Tan, "How to Choose Battery for Your Emergency Lighting Wisely?," 29 Sept. 2019. [Online]. Available: www.sanforce-tech.com/how-to-choose-suitable-battery-emergency-lighting-wisely/.

[20] F. Leng, C. M. Tan and M. Pecht, "Effect of Temperature on the Aging rate of Li Ion Battery Operating above Room Temperature," 6 Aug 2015. [Online]. Available: https://www.nature.com/articles/srep12967. [Accessed 16 April 2020].

[21] J. Donovan, "Selecting Antennas for Embedded Designs," Convergence Promotions LLC, 08 11 2012. [Online]. Available: https://www.digikey.com/en/articles/selecting-antennas-for-embedded-designs.

[22] A. Designer, "Embedded RF Design: Ceramic Chip Antennas vs. PCB Trace Antennas," Altium Designer, 16 2 2018. [Online]. Available: https://resources.altium.com/p/embedded-rf-design-ceramic-chip-antennas-vs-pcb-trace-antennas.

[23] J. Redmon, "YOLO: Real-Time Object Detection," arXiv, 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed 27 Feb. 2020].

[24] A. Kathuria, "How to implement a YOLO (v3) object detector from scratch in PyTorch: Part 1," PaperspaceBlog, 16 Apr. 2018. [Online]. Available: https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/. [Accessed 27 Feb. 2020].

[25] M. Sandler, "MobileNet," 12 Nov. 2019. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet/README.md. [Accessed 28 Feb. 2020].

[26] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le and H. Adam, "Searching for MobileNetV3," 6 May 2019. [Online]. Available: https://arxiv.org/abs/1905.02244. [Accessed 28 Feb. 2020].

[27] Keras, "Keras: The Python Deep Learning library," [Online]. Available: https://keras.io/. [Accessed 29 Feb. 2020].

[28] PyTorch, "PyTorch: From Reasearch to Production," [Online]. Available: https://pytorch.org/.

[29] n. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, M. Irving, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenburg, D. Man, R. Monga, S. Moore, D. Murry, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Vi, O. Vinyals, P. Warden, M. Wattenburg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow," 2015. [Online]. Available: https://www.tensorflow.org/lite.

[30] OpenCV, "Open Source Computer Vision," 18 April 2020. [Online]. Available: https://docs.opencv.org/3.4/df/d6c/group__ximgproc__superpixel.html. [Accessed 18 April 2020].

[31] N. True, "Computer Vision Based Fire Detection," University of California, La Jolla, CA.

[32] C. Yu, Z. Mei and X. Zhang, "A Real-time Video Fire Flame and Smoke Detection Algorithm," 14 Aug. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705813013222. [Accessed 28 Feb. 2020].

[33] "Optical Flow," [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html.

[34] A. J. Dunnings and T. P. Breckon, "EXPERIMENTALLY DEFINED CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE VARIANTS FOR NON-TEMPORAL REAL-TIME FIRE DETECTION," [Online]. Available: https://breckon.org/toby/publications/papers/dunnings18fire.pdf. [Accessed 18 April 2020].

[35] A. Deshmukh, T. Breckon and A. Dunnings, "fire detection cnn," 19 Dec 2019. [Online]. Available: https://github.com/tobybreckon/fire-detection-cnn. [Accessed 18 April 2020].

[36] A. Rosebrock, "Fire and smoke detection with Keras and Deep Learning," 18 November 2019. [Online]. Available: https://www.pyimagesearch.com/2019/11/18/fire-and-smoke-detection-with-keras-and-deep-learning/.

[37] M. R. M. F. R. Errynando Surya Sasmita, "Integrating Forest Fire Detection with Wireless Sensor Network Based on Long Range Radio," in *The 2018 International Conference on Control, Electronics, Renewable Energy and Communications*, Bandung, 2018.

[38] Y. E. Aslan, I. Korpeoglu and Ö. Ulusoy, "A framework for use of wireless sensor networks in forest fire detection and monitoring," Nov. 2012. [Online]. Available: https://doi.org/10.1016/j.compenvurbsys.2012.03.002. [Accessed 30 Jan. 2020].

[39] S. Bouckaert, E. D. Poorter, P. D. Mil, I. Moerman and P. Demeester, "Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies," 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5425861. [Accessed 30 Jan. 2020].

[40] IEEE, "Integrating Forest Fire Detection with Wireless Sensor Network Based on Long Range Radio," 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8711991. [Accessed 30 Jan. 2020].

[41] M. Rouse, "internet of things (IoT)," TechTarget, 2 2020. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[42] Semtech, "AN1200.22 LRa Modulation Basics," Semtech, Camarillo, CA, 2015.

[43] S. Ghoslya, "LoRa: Symbol Generation," [Online]. Available: https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html.

[44] Bosch, "Low Power Gas, Pressure, Temperature and Humidity Sensor," [Online]. Available: https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf. [Accessed 1 April 2020].

[45] PYREOS, "AN136 Application Note: Understanding pyroelectric infrared detectors," 2020. [Online]. Available: https://pyreos.com/wp-content/uploads/2020/01/AN136-Understanding-pyroelectric-infrared-detectors.pdf.

[46] PYREOS, "ezPyroTM SMD I2C Pyroelectric Infrared Sensor," 22 June 2018. [Online]. Available: https://forum.pycom.io/assets/uploads/files/1585523830287-sensor_datasheet-min-3.pdf. [Accessed 20 May 2020].

[47] A. Rosebrock, "How to (quickly) build a deep learning image dataset," 9 April 2018. [Online]. Available: https://www.pyimagesearch.com/2018/04/09/how-to-quickly-build-a-deep-learning-image-dataset/.

[48] J. M. LEE, "Fighting Fire with Fire: How Controlled Burns Keep Us Safe," 14 January 2020. [Online]. Available: https://www.ucf.edu/news/fighting-fire-with-fire/. [Accessed 10 April 2020].

[49] Siemens STEM Day, "Siemens Stem Day," [Online]. Available: http://www.siemensstemday.com/. [Accessed 18 April 2020].

[50] Siemens, "Siemens Gas Turbines," 2019. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:10f4860b140b2456f05d32629d8d758dc00bcc30/gas-turbines-siemens-interactive.pdf. [Accessed 18 April 2020].

[51] G. K. C. S. a. N. T. Nancy H Ulerich, "CONDITION BASED MONITORING OF GAS TURBINE COMBUSTION COMPONENTS," Siemens Energy, Inc., Jenetek Sensors, Inc., K Science, GP LLC., Orlando, Waltham, San Antonio, 2013.

[52] C. Isiadinso, "TEMPERATURE, PRESSURE, & SPEED SENSING SYSTEMS OF A GAS TURBINE FIRST STAGE ROTOR BLADE," *Sensor Systems,* p. 2015, 24 November 2015.

[53] M. H. RAITHATHA, "SIEMENS-UV OPTICAL FLAME DETECTION," College of Engineering University of California, Berkley, 2013.

[54] Siemens AG, "Siemens Assets," 2019. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:d9283b58-4f74-4f05-a8fb-0ccf0439ee82/version:1557162462/sgt-a05-service-solutions-2019.pdf. [Accessed 18 April 2020].

[55] Siemens, "Siemens IOT Assest," 1 April 2019. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:131ac2f9-5e8b-4968-ba2f-734eefccdb50/version:1556633115/turning-iot-into-reality-whitepaper-by-siemens-iot-services-fina.pdf. [Accessed 18 April 2020].

[56] Siemens Gamesa, "Siemens Games Renewable Energy," [Online]. Available: https://www.siemensgamesa.com/en-int/about-us. [Accessed 18 April 2020].

[57] M. Froese, "Fire prevention and protection for wind turbines offshore and on," 24 June 2016. [Online]. Available: https://www.windpowerengineering.com/fire-prevention-protection-wind-turbines-offshore/. [Accessed 18 April 2020].

[58] A. Spatuzza, "Recharge News," 3 February 2020. [Online]. Available: https://www.rechargenews.com/wind/siemens-gamesa-investigates-after-wind-turbine-rotor-crash-in-brazil/2-1-749505. [Accessed 18 April 2020].

[59] M. Foster, "Gamesa turbine catches fire in Japan," 22 August 2017. [Online]. Available: https://www.windpowermonthly.com/article/1442624/gamesa-turbine-catches-fire-japan. [Accessed 18 April 2020].

[60] K. Garus, "Siemens' fire detection and extinguishing system is certified," 23 July 2013. [Online]. Available: https://www.offshorewindindustry.com/news/siemens-fire-detection-and-extinguishing. [Accessed 18 April 2020].

[61] D. Zima, "Lora Best Design Practices EMC Compliance," in *Lora Workshop, UCF*, Orlando, 2020.

[62] T. X.-P. Zhao, S. Acherman and G. Wei, "Dust and Smoke Detection for Multi-Channel Imagers," Oct. 2010. [Online]. Available: https://www.researchgate.net/publication/47380702_Dust_and_Smoke_Detection_for_Multi-Channel_Imagers. [Accessed 19 March 2020].

[63] X. Xu, "Everitt's blog," github, 10 Aug. 2018. [Online]. Available: https://everitt257.github.io/post/2018/08/10/object_detection.html. [Accessed 27 Feb. 2020].

[64] Z. A. S. Syed, "Frequency, Range and type of Wireless Communication," in *NSA, University of Oslo*, 2016.

[65] S. S. Roy, "Real-Time Object Detection on Raspberry Pi Using OpenCV DNN," 23 Oct. 2018. [Online]. Available: https://heartbeat.fritz.ai/real-time-object-detection-on-raspberry-pi-using-opencv-dnn-98827255fa60. [Accessed 28 Feb. 2020].

[66] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Cornell University, Ithica, New York, 2016.

[67] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Cornell University, Ithaca, New York, 2018.

[68] H. W. Ott, "Henry Ott Consultants," 14 Feb. 2001. [Online]. Available: http://www.hottconsultants.com/techtips/freq-wavelength.html. [Accessed 8 March 2020].

[69] J. Noci, "Antenna Design Overview — Copter documentation," Ardupilot.org, 2020. [Online]. Available: https://ardupilot.org/copter/docs/common-antenna-design.html. [Accessed 10 March 2020].

[70] A. Mordvintsev and A. K., "OpenCV-Python Tutorials," 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html. [Accessed 29 Feb. 2020].

[71] H. Friis, A Note on a Simple Transmission Formula, IRE Proc.: 254–256, 1946.

[72] L. E. Frenzel, "Welcome To Antennas 101," Electronic Design, 13 Aug. 2008. [Online]. Available: https://www.electronicdesign.com/technologies/passives/article/21769333/welcome-to-antennas-101. [Accessed 12 March 2020].

[73] E. Edje, "Tutorial to set up TensorFlow Object Detection API on the Raspberry Pi," 19 Oct. 2019. [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/blob/master/README.md. [Accessed 28 Feb. 2020].

[74] W. De-Chang, X. Cui, E. Park and C. Jin, "Adaptive flame detection using randomness testing and robust features," Oct. 2013. [Online]. Available: https://www.researchgate.net/publication/257410367_Adaptive_flame_detection_using_randomness_testing_and_robust_features. [Accessed 19 Mar. 2020].

[75] RF Wireless World, "Smoke Detector basics | Smoke Detector types," [Online]. Available: https://www.rfwireless-world.com/Articles/smoke-detector-basics-and-smoke-detector-types.html. [Accessed 19 Mar. 2020].

[76] IEEE Std 145-1993(R2004), IEEE Standard Definitions of Terms for Antennas., New York, NY: The Institute of Electrical and Electronics Engineers, 1993.

[77] Legal Information Institute, "47 CFR § 15.209 - Radiated emission limits; general requirements.," 2020. [Online]. Available: https://www.law.cornell.edu/cfr/text/47/15.209. [Accessed 10 March 2020].

[78] "This Plastic’s on Fire! 4 Types of Flame Retardant Plastic Additives," Craftech Industries, [Online]. Available: https://www.craftechind.com/this-plastics-on-fire-4-types-of-flame-retardant-plastic-additives/. [Accessed 1 April 2020].

[79] P. Smith, "Siemens develops automatic offshore fire-fighting system," 17 February 2014. [Online]. Available: https://www.windpowermonthly.com/article/1281184/siemens-develops-automatic-offshore-fire-fighting-system. [Accessed 18 April 2020].

[80] Melexis, "MLX90640 32x24 IR array," 2012. [Online]. Available: https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90640. [Accessed 1 April 2020].