UNIVERSITY OF CENTRAL FLORIDA

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

Department of Electrical Engineering and Computer Engineering
Senior Design I Project Report, Group 4, Spring 2020

Noora Dawood – Electrical Engineering

Nicholas Hainline – Electrical Engineering

Jonathan Kessluk – Computer Engineering

Arisa Kitagishi – Computer Engineering

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 (Ouni, Ayoub, & Kamoun, 2019; Jurvélius, 2003). 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 (Jurvélius, 2003). 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 (Alkhatib, 2017; United Nations Enviroment Programme, 2020). 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.

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 forest fires. The devices will be portable so that in can be mounted on trees and will be able to 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 will include 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 will manage and maintain the network and allow for sending messages through the network to a “gateway”. The Fire Detection software will use sensor data to determine, through image processing and/or machine learning, if there is a fire. The two software sub-systems will communicate to know whether or not to send a message.

Control: To process and control the data, the system will include a microcontroller and a raspberry pi that work together to achieve the goals of the system. The microcontroller will handle the wireless communication and joining and maintaining the network. The Raspberry Pi will handle sensor data and determine if there is a need to send a message across the network.

Communication: A mesh network will be adopted for the monitoring system to allow the devices scattered in the forest to communicate dynamically and send data to be processed at the central hub.

Power Supply: The system will be powered by solar panels mounted to the top of the tree connected to the individual device. Since each device will draw modest current, the solar system will be 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 preliminary expected requirements and constraints as determined by the project specification.

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 hub
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
R9	Electrical	The system shall read voltages of the battery to determine health
R10	Software	The system shall report its own status/health to the hub.
R11	Software	The system shall store configuration and user defined data in non-volatile memory
R12	Mechanical	The system shall withstand fires up to 4 hours
R13	Mechanical	The system shall be able to withstand normal weather conditions
R14	Electrical	The system shall monitor environment with temperature and humidity sensors
R15	System	Average installation time should not exceed 30 minutes
R16	Mechanical	The system shall be able to withstand normal weather conditions
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. (15 x 15 x 15 cm)
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 (ASQ, 2020). 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 our "House of Quality".

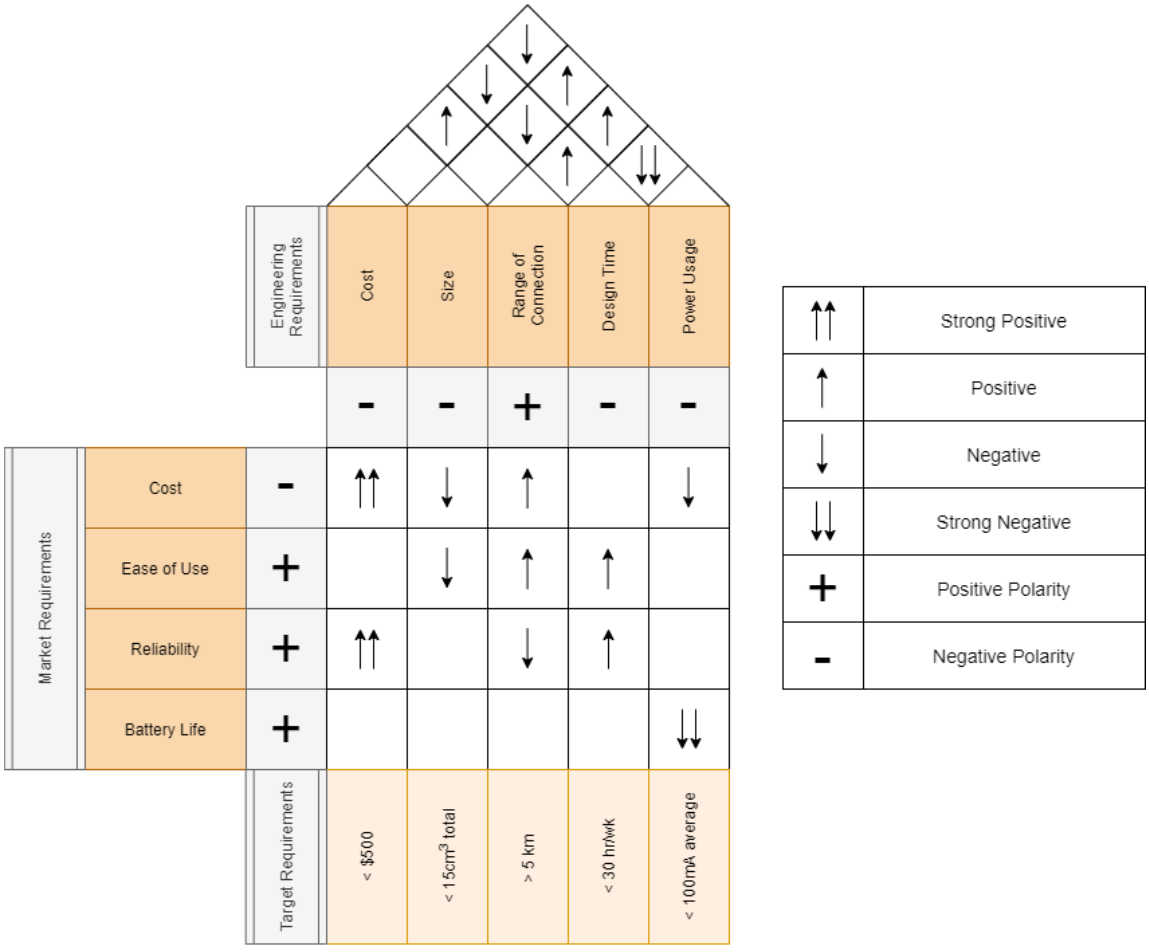


Figure 1: House of Quality

Our House of Quality has four major relationships displayed. These four relationships are the cornerstone to our design. The most obvious relationship is the connection between the Cost of the product and the cost to design the product. If it costs us 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. With lower power consumption we 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.

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 that we have when designing the project. Our project uses electricity meaning that there is important consideration that we should make in its design.

3.2.1. RoHS

Some materials are hazardous and harmful to the environment. To mitigate our effects on the environment, this project will be RoHS compliant to the best of our ability. This means that we will avoid components that contain lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ethers, and some phthalates (RoHS, 2005).

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. How good would a fire detection system be if every so often the systems caught fire. To alleviate this issue the state of the battery is constantly monitored at every start of the detection system to make sure that the voltage and current are within the specifications laid out by the manufacturer. To accomplish this task the MCU would read the voltage and current across a resistor and compare it to a set of known values to check if the battery is within its specifications. The state of the system will be converted from analog to digital and transmitted if any anomalous behavior is detected at which time the system can power down and wait for a technician to repair it.

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 (IPC-2221, 1998; Electronic Code of Federal Regulations, 1996).

4. Research and Background Information

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

4.1. Current Fire Detection Systems

The first step is 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 (Nörthemann, Bienge, Müller, & Moritz, 2013). 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 (Nörthemann, Bienge, Müller, & Moritz, 2013):

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 we aim to design (Arduino fire alarm system using temperature and smoke sensor with Android connectivity, n.d.).

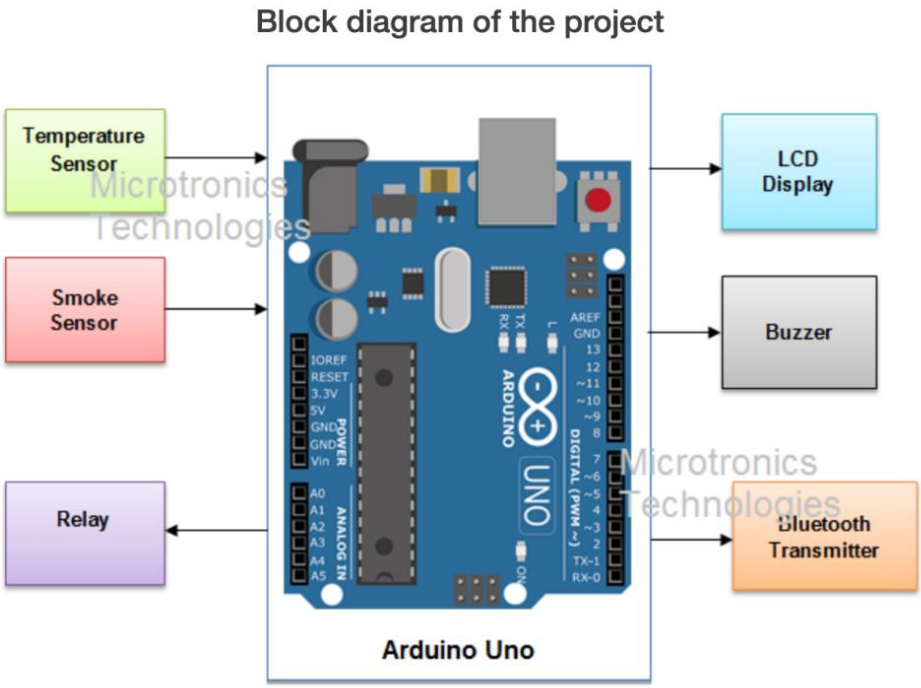


Figure 2: Arduino Uno being used in a similar project (Arduino fire alarm system using temperature and smoke sensor with Android connectivity, n.d.)

A major drawback of this kit is the high market value price despite the product using 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 (Bluetooth, 2020). 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 project would achieve a similar objective to the Arduino system; however, most importantly the cost will be significantly lower with wider range and similar fire detection technologies.

4.2. Background Research

After looking at systems that already exist to detect fires, we need to investigate 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) (Leens, 2009). SPI uses a protocol where a single device sends the communication to the slave devices, thus it uses the single-master communication protocol (Leens, 2009). In order for communication to occur, the master and slave must use SCLK frequency, CPOL, and CPHA (Leens, 2009). In the event when multiple slaves exist, the master will reconfigure itself each time to initiate the communication with each slave (Leens, 2009). 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 (Leens, 2009). 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.

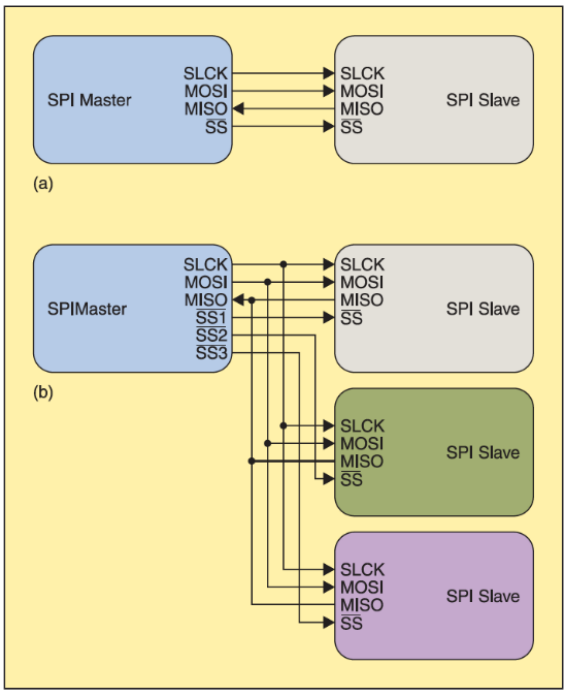


Figure 3: SPI Topology (Leens, 2009)

I2C or Inter-integrated circuit is known for requiring a 2-wire connection between the peripherals and the microcontroller (Leens, 2009). The two signals are called serial data (SDA) and serial clock (SCL) (Leens, 2009). 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 (Leens, 2009). The bus master is the IC that initiates the data transfer, while the remaining IC are considered bus slaves (Leens, 2009). 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 (Leens, 2009). There some variants of I2C that include a low speed mode at 1kb/s and fast mode + at 1Mb/s (Leens, 2009).

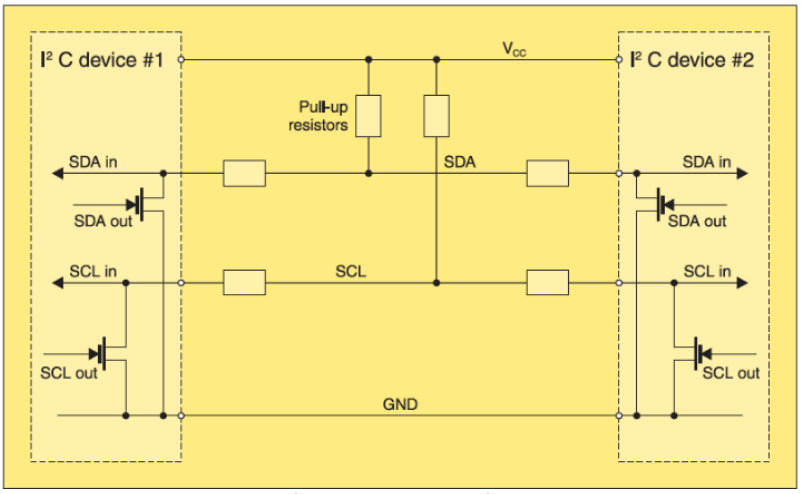


Figure 4: I2C Topology

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 (Leens, 2009). 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 (Leens, 2009).

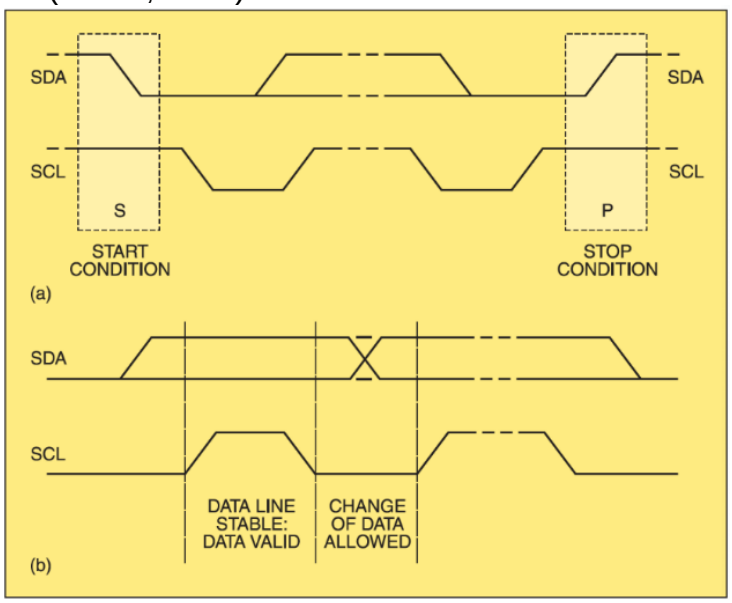


Figure 5: Timing Diagram of I2C (Leens, 2009)

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 (Leens, 2009). 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 (Leens, 2009).

After examining both protocols, I2C would be 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. The drawbacks will be considered during implementation but do not pose a risk for the project.

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 (Gaur, et al., 2019). 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 (Fonollosa, Solorzano, & Marco, 2018)

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%

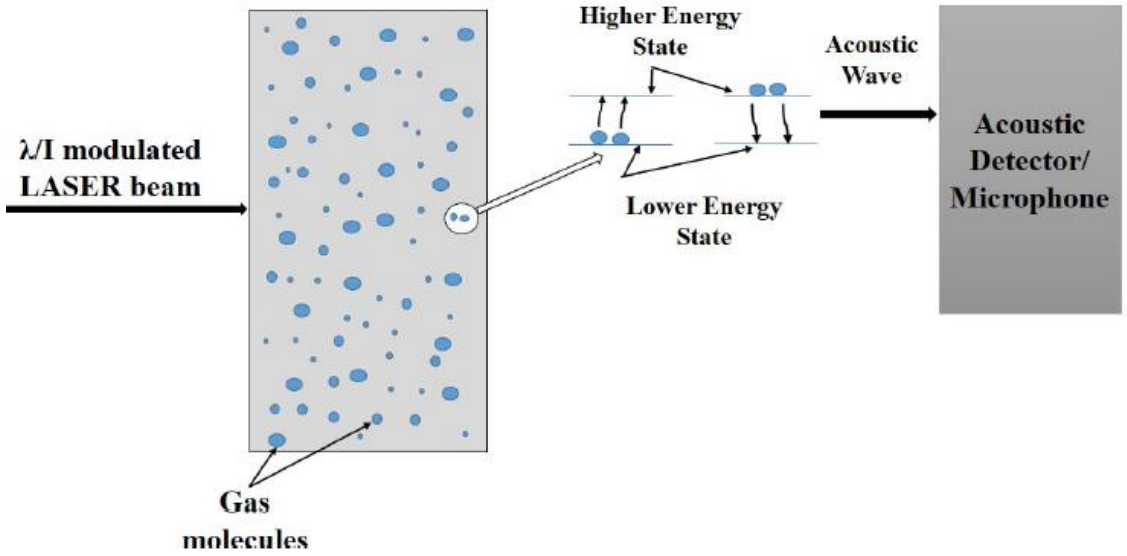
Sensors vary by the material used; existing materials in the market include semiconductor, catalytic bead, photoionization, infrared, and electrochemical. Additional gas sensor types include optical, acoustic, gas chromatograph, and calorimetric (Gaur, et al., 2019).

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_2O gasses (Gaur, et al., 2019). 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 (Gaur, et al., 2019).

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 (Gaur, et al., 2019). However, despite this issue, zeolites have been used instead of metal oxides to compensate for this issue (Gaur, et al., 2019). Moreover, gas sensors that use polymers have shown to enhance sensitivity (Gaur, et al., 2019).

Based on spectroscopy laws gas sensors that use optical methods are more stable, sensitive, possess better selectivity, and have a low response time (Gaur, et al., 2019). However, optical gas sensors come with the disadvantage of higher costs (Gaur, et al., 2019).

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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019)

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 (Gaur, et al., 2019). These gas sensors use metal oxide or n-LTPS MOS Schottky diode on a glass substrate (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019).

Smoke detection methods that use the photoelectric principle are primarily used for smoldering conditions and is effective in doing so; response times are quick (Gaur, et al., 2019). In this method, the ionization smoke sensor measures smoke relative to the ionization levels in the air (Gaur, et al., 2019). A potential difference is applied through a chamber and the output current is measured as a result (Gaur, et al., 2019). Moreover, photoelectric method dictates that the concentration of smoke in the air will proportionally increase the light scattering capacity (Gaur, et al., 2019). 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 (Gaur, et al., 2019). When the smoke concentrations are high, smoke particles decrease the number of alpha particles in the gate terminal which then reduces the current (Gaur, et al., 2019). Other photoelectrical methods investigated the range of transmission for wood smoke using a white polychromatic LED, an optical fiber, pyrex glass window, and photodiodes (Gaur, et al., 2019). This could be implemented in a forest environment. The figure below provides a visual of how photoelectrical smoke sensors works.

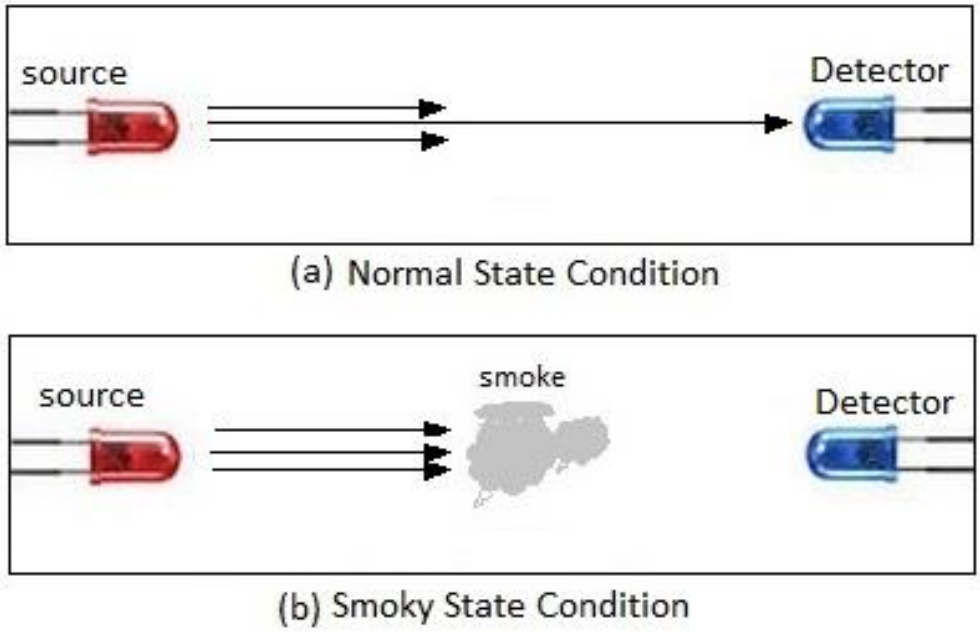


Figure 7: Visual representation of photoelectric smoke detection (Gaur, et al., 2019)

Visual techniques mostly use cameras which can detect both flame and smoke (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019).

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 (Gaur, et al., 2019). Non-visual flame sensors use ultra-violet, visible, and infrared rays (Gaur, et al., 2019). This is because flames emit a radiation whose intensity is determined by the flame temperature and the type of fuel burning (Gaur, et al., 2019). An ultra-violet sensor is used to measure the brightness since UV sensors are not impacted by interreferences from other radiations such as infrared (Gaur, et al., 2019). Additionally, infrared and visible light sensors are used to measure flame. However, IR and visible light sensors are more effective than ultra-violet sensors (Gaur, et al., 2019). UV sensors tend give out more false positive alerts due UV sensors emitting sparks of UV spectra that essential interferes with the signal (Gaur, et al., 2019). 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 (Gaur, et al., 2019).

Visual techniques for detecting flame can be difficult because standard heat, smoke flame, and gas sensors is the delay in receiving a response (Gaur, et al., 2019). This is because the particles must reach the sensors in order for the sensor to trigger a response signal (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019). 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 (Gaur, et al., 2019). This requires the use of color spaces; YCbCr color space showed to be the most effective for flame detection (Gaur, et al., 2019). Other color spaces that can also been used are RGB, CIE L∗a∗b∗, YUV, or HIS (Gaur, et al., 2019).

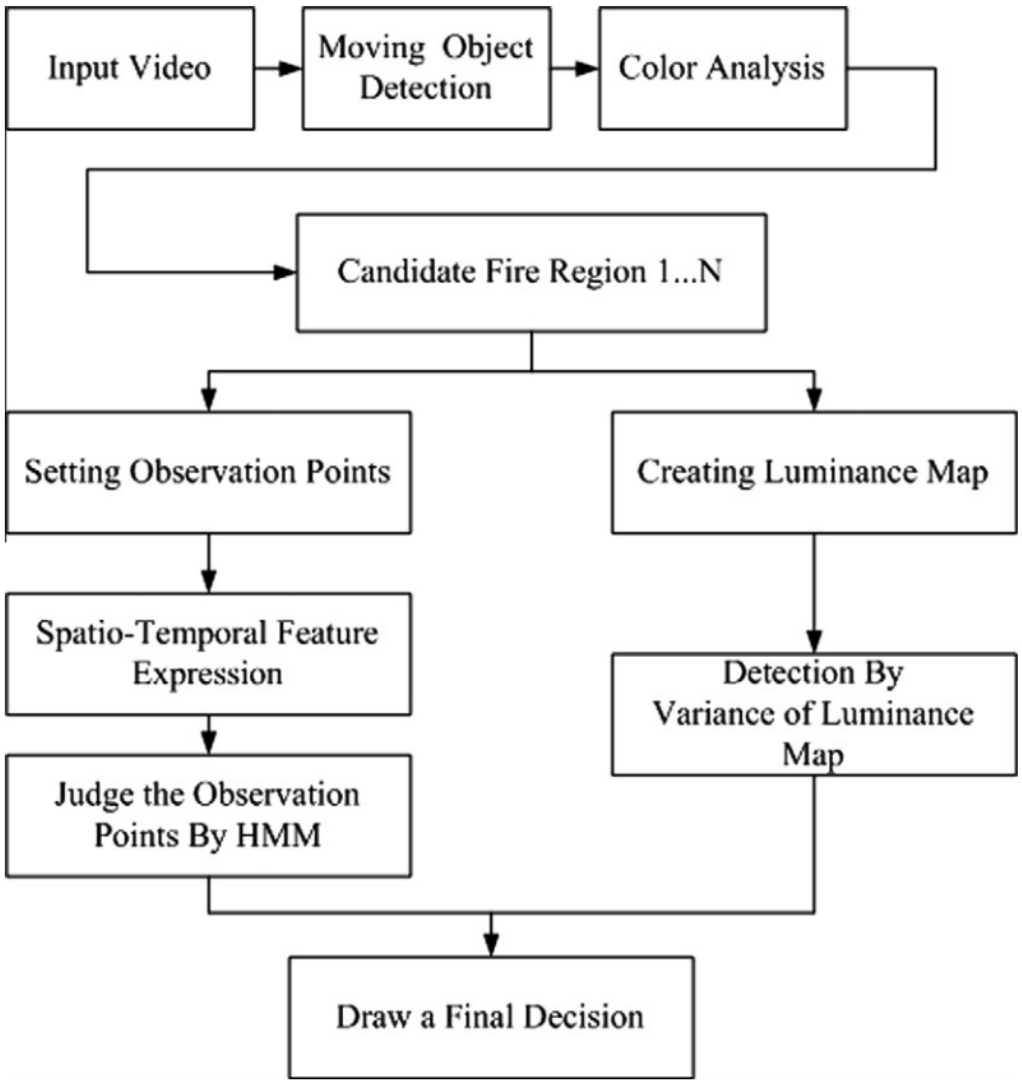


Figure 8: Hidden Markov model used to detect flame flicker (Liqiang Wang, 2011)

Color information is not enough to provide accurate results (Gaur, et al., 2019). Movement of fire has also been examined for fire detection techniques by using background subtraction method, temporal differencing, and optical flow analysis (Gaur, et al., 2019). 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 (Gaur, et al., 2019). Moreover, a moving camera can be used to observe moving flame pixels without using background subtraction (Gaur, et al., 2019). 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 (Gaur, et al., 2019). The reliability of the results was increased using a "convolution operation" (Gaur, et al., 2019).

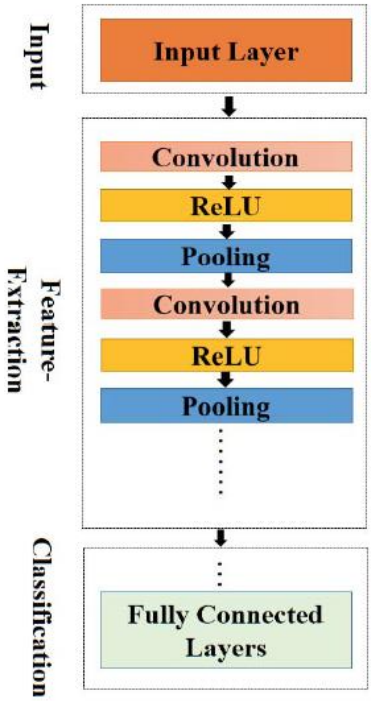
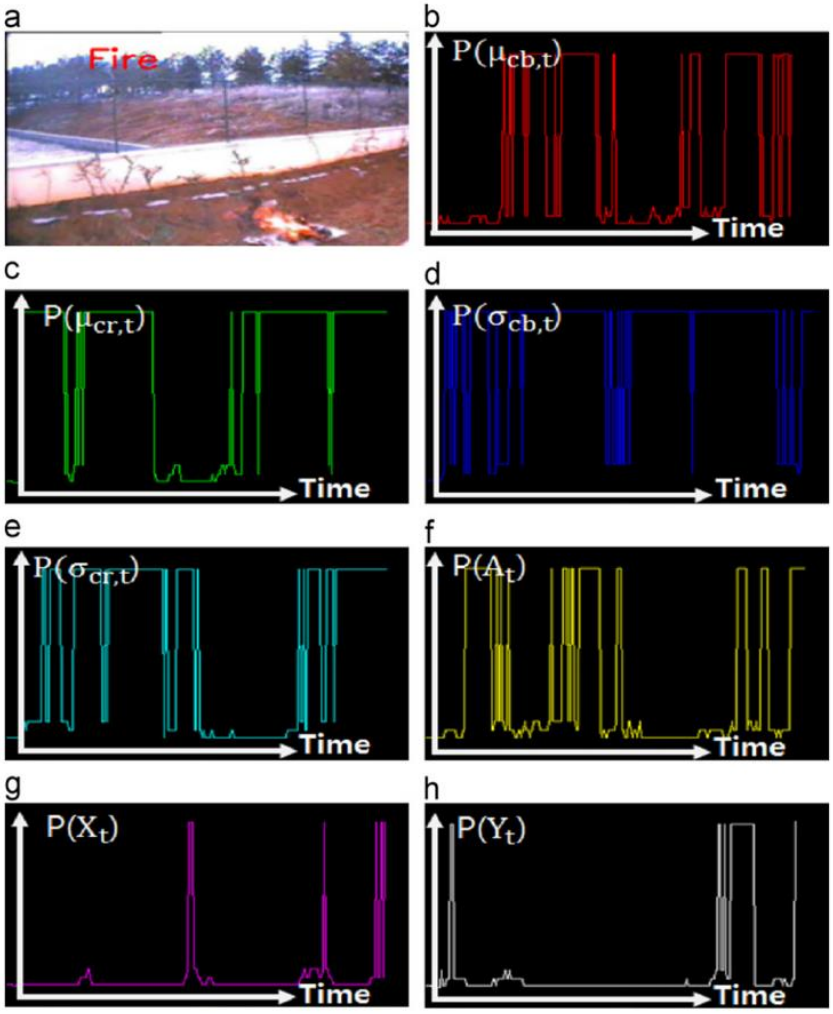


Figure 9: Convolution neural networks approach layers (Gaur, et al., 2019)

The second approach of designing the fire detection algorithm utilizes a learning-based approach (Gaur, et al., 2019). 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 is 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 each prototype will not be fire resistant. This is done to minimize the cost and manufacturing process 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 (ScienceDirect). 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 (ScienceDirect).

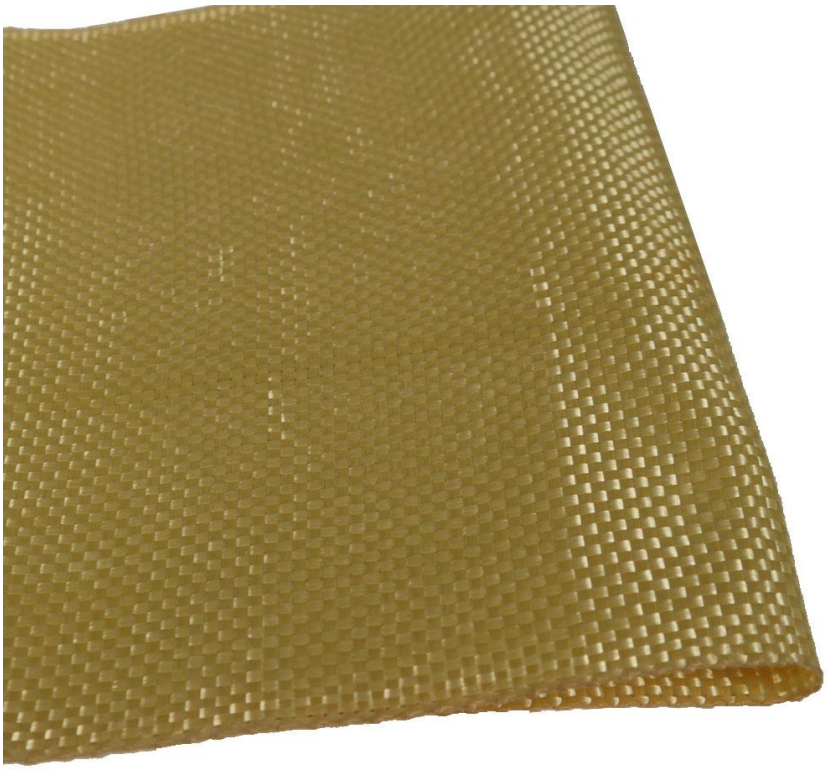


Figure 11: Synthetic Fiber Kevlar (Jamestown Distributors)

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 we would want a plastic compound that does neither. 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 has to get through first to burn the plastic.

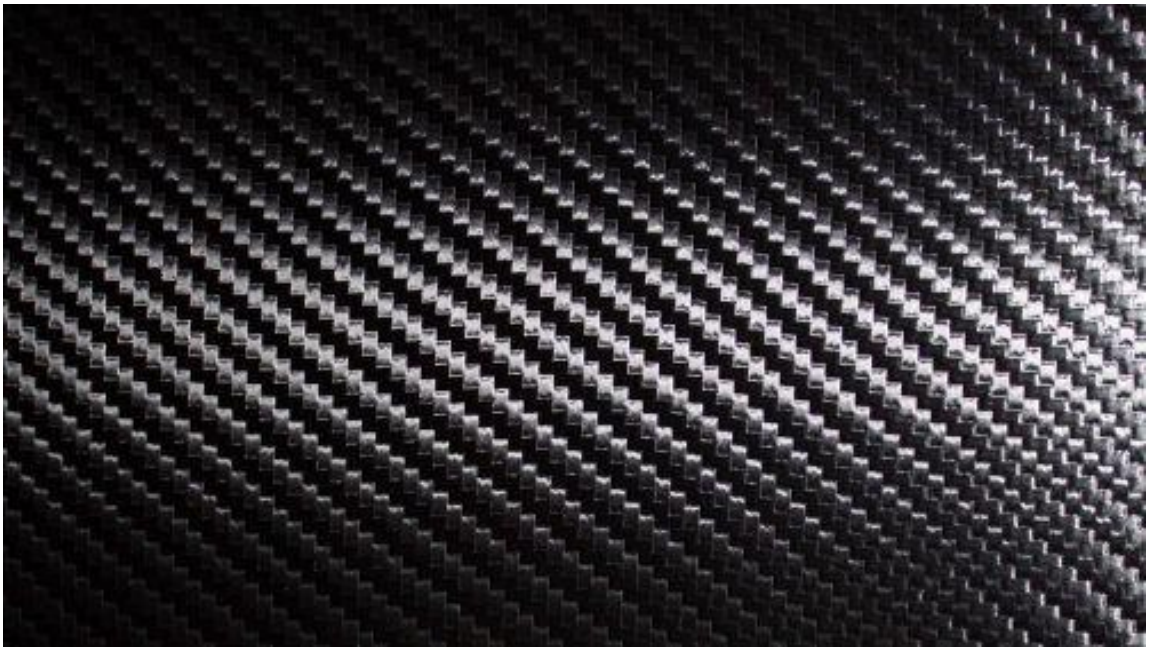


Figure 12: Carbon Fiber Weave (WallpaperAccess)

4.2.4. Battery Charging and Battery Chemistries

The battery for the project must be able to last throughout the night and when the solar radiation is low on average for the winter months it must be able to 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 couldn't 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 would have to be used. For this project the best option to go with would be to choose a battery within budget that is the most power dense and the chemical makeup of said battery allows for the most charge cycles.

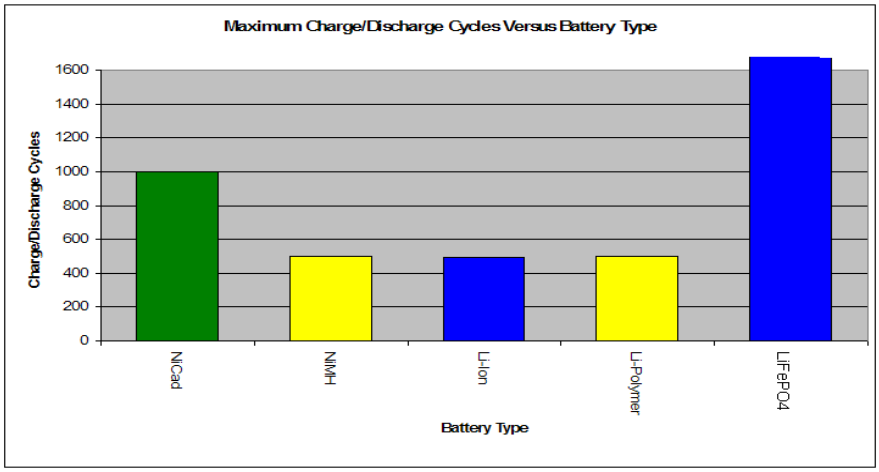


Figure 13: Maximum Charge/Discharge Cycles Versus Battery Type (Tan, 2019)

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 would make 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 seemed like the best option for the prototype system.

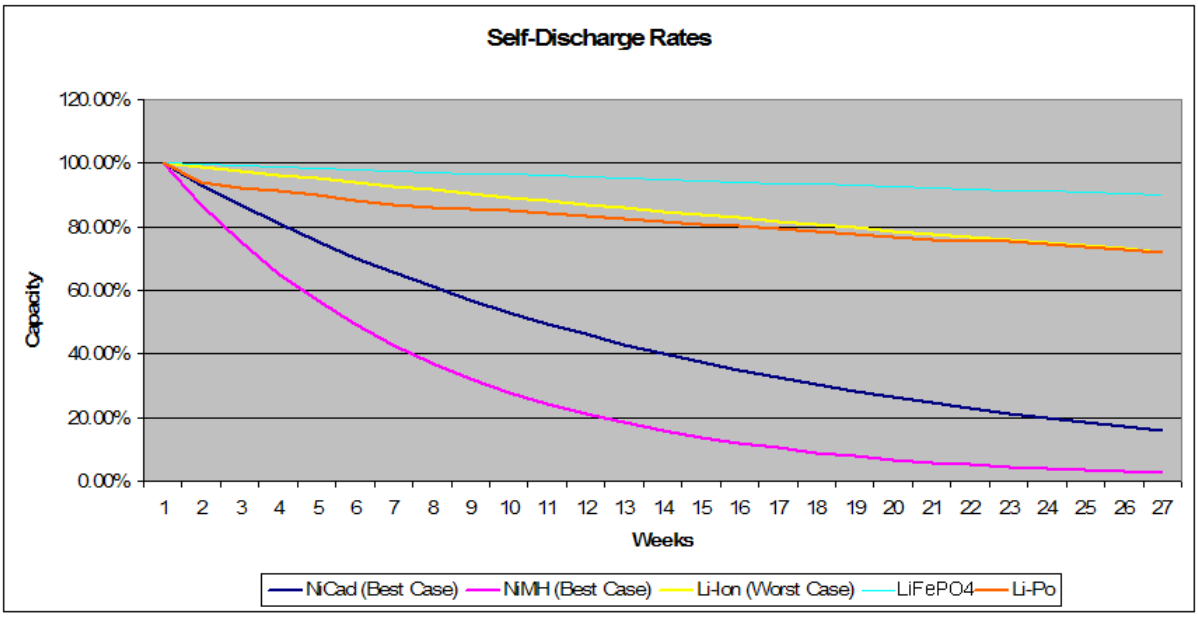


Figure 14: Self-Discharge Rates of Batteries (Tan, 2019)

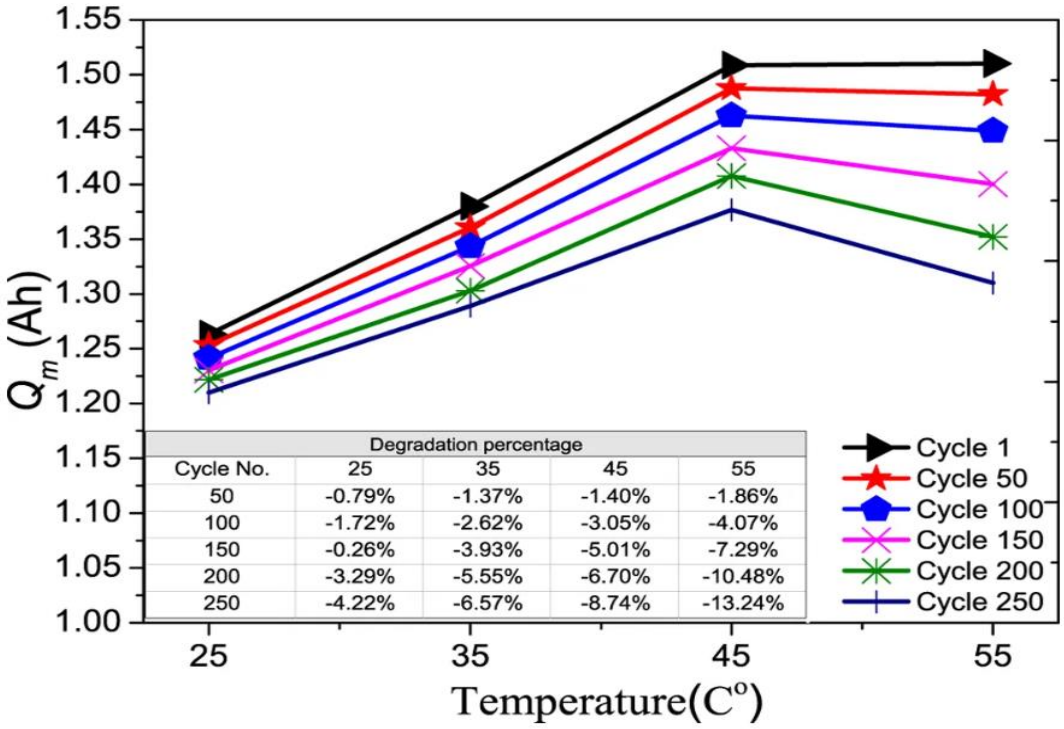
To further cause issues in choosing a battery for the project the self-discharge rate was a factor to deal with. Self-discharge is when a battery will slowly lose charge over time when not in use. Since the solar panels were going to charge the batteries every day the discharge rate wasn't a massive impact but the battery that was chosen needed to be at least able to handle a few days 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.

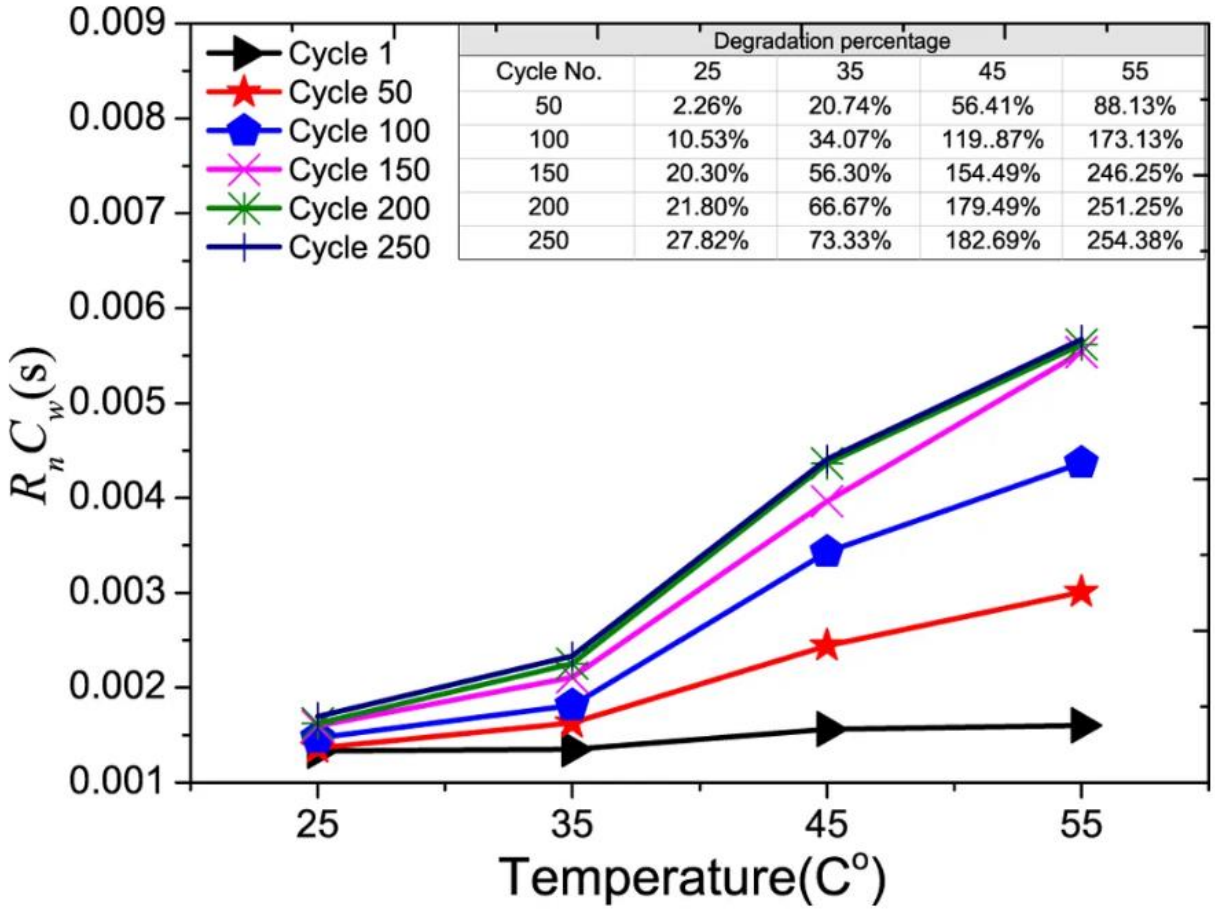
The figure bellow is a graph that highlights what happens to the battery maximum charge storage capacity if the temperature is increased like in summertime conditions. As shown the max storage of the battery is not affected until the natural battery charge cycle lifespan starts to end. The high temperatures 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.



Degradation percentage				
Cycle No.	25	35	45	55
50	-0.79%	-1.37%	-1.40%	-1.86%
100	-1.72%	-2.62%	-3.05%	-4.07%
150	-0.26%	-3.93%	-5.01%	-7.29%
200	-3.29%	-5.55%	-6.70%	-10.48%
250	-4.22%	-6.57%	-8.74%	-13.24%

Figure 15: Temperature Vs Charge (Leng, Tan, & Pecht, 2015)

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) (Leng, Tan, & Pecht, 2015)

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 on multiple decisions. The major decision this battery was chosen for this project was cost and availability. Li-ion is the most used battery in technology right now.
This means that near and size and configuration for the battery can be 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 the battery is depleted or how the batteries come with safety built in to prevent over charging, or overvolting, the battery which is good for preventing the system from causing a fire. 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 doesn't 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 is a solar array that is 12 volts nominally and is hooked up to a voltage regulator that converts that 12 volts into 5 volts with a mix current draw of 2 amps. This is then hooked up to a lithium battery charging IC. The battery and solar panels are then hooked to a buck-boost converted that maintains 3.3-volt and 5-volt rails for the Pi and sensors for the system. This power system must be able to utilize the solar cells and battery at the same time as to not put undue stress on the battery and cause constant charging to occur.

To ensure that the system operates at the proper voltages, we must employ some kind of DC to DC power converter. In the case of our solar panel, we must ensure that at all times it outputs an acceptable voltage for charging to our charging IC, but at the same time allows for the proper voltages to be fed into the system during normal usage. This will require step down converters.

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 don't 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 8V. If we use a

linear regulator to step down to 5V or 3.3V, 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, we must make sure that we are 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 we will implement 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 we are charging the battery and running the circuit off of the solar panel, we 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 we 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, we don't want the device to have too much affect 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, we can 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 our design, we do 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, we determined the needs of the system to be almost 29Wh per day. For 5 hours of maximum power output with an MPPT we need 5.76W 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 wouldn’t 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 two applied. The needed array and battery size were doubled to account for unforeseen issues. This didn’t affect the cost to greatly as finding solar panels in the sizes needed was difficult and had a price gouge on them for being so small. Cheaper panels that were larger and more powerful were cheaper so the system being doubled was the best option.

4.2.7.1. Azimuth and Elevation/Tilt Angle

To properly arrange a solar array to absorb the most solar radiation as possible it needs to have the proper azimuth and inclination angle for the location it is 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.

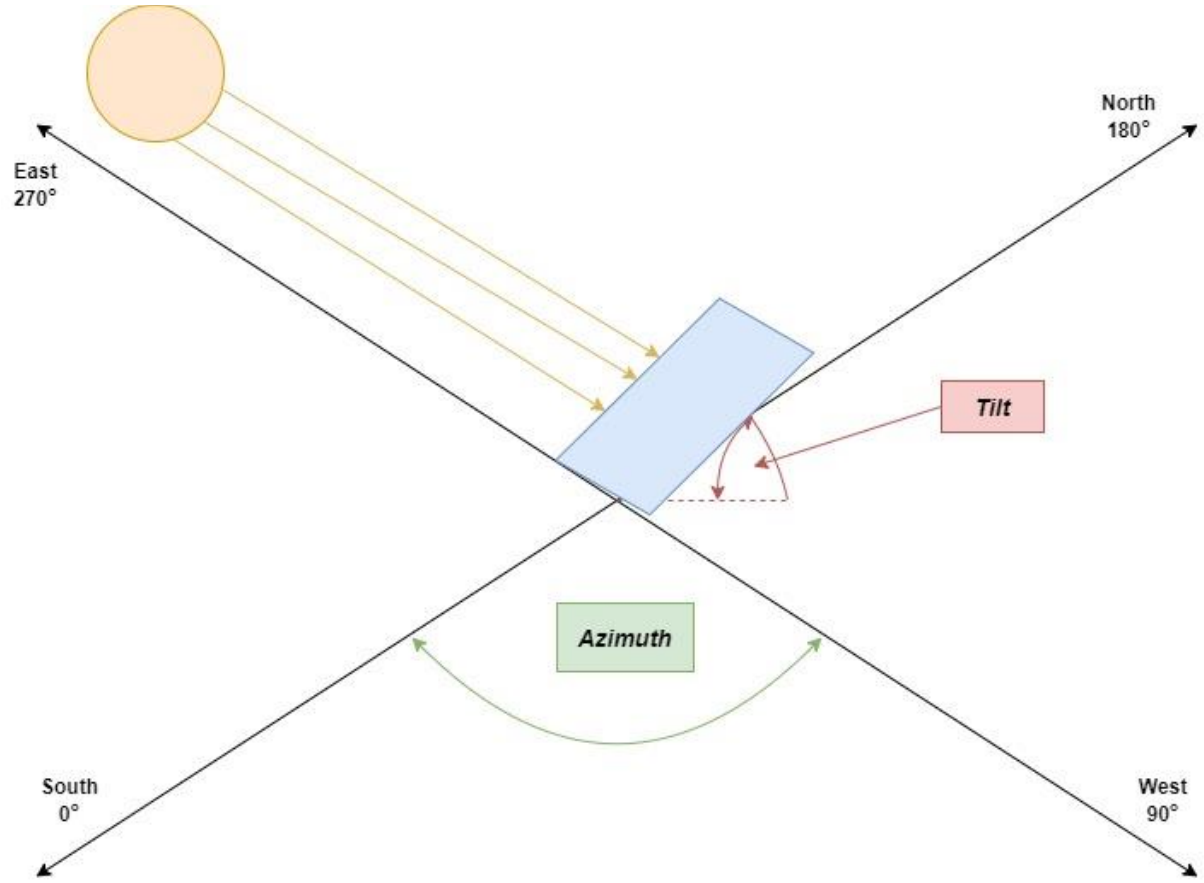


Figure 17: Azimuth and Elevation

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 be able 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. Ideally, we use the IEEE 802.1 Wi-Fi scheme. This however has limited range. These numbers are averages. Actual distances depend on a variety of variables.The following numbers were compared to get the maximum range out of our device.

Table 5: Comparison Between Technology Ranges and Frequencies

Technology	Frequency	Range
Bluetooth	2.45 GHz	30 Feet
Wifi	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

Furthermore, we can use the Free-Space Path Loss equation to determine the attenuation of radio energy between two antennas.

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

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

Assuming trying different ranges, we can generate a graph that shows the best option for longer ranges and frequencies to minimize attenuation.

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.

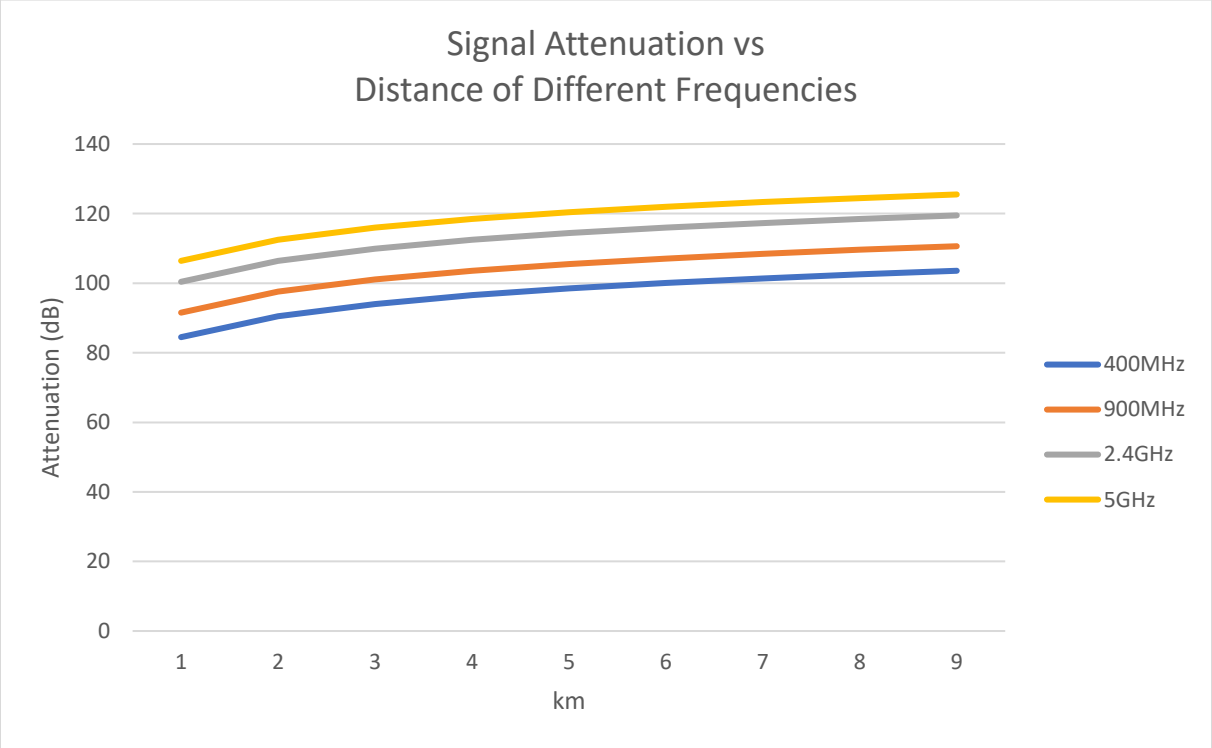


Figure 18: Signal Attenuation Vs Distance of Different Frequencies

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[45] 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.[46] 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 basically free as 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 [46]. 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.

4.3. 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, we need controllers that can support wireless communication and also process the sensor data. 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 following section about Radio Frequency communication Technologies, we settled on the LoRa modulation standard 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 SAM R35 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.

Microcontroller Comparison Table
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 integration	Required Peripherals
MSP430	66KB	512KB	-/-/2.36/.00045	Yes	No	Yes
ATMEGA4808	48KB	6KB	-/-/11/.0001	Yes	No	Yes
ESP32	520KB	4-16MB	240/100/30/.005	Yes	Yes	Yes
SAMR3x	32KB	256KB	95/16/4.5/.0008	Yes	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: 100mA. 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 when idling: 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 is seeming like the best modulation technique as it is simple, has many tutorials and examples, and has a variety of resources to pull from. Furthermore, the SAMR35 microcontroller already has the LoRa modulation scheme built into the chip.

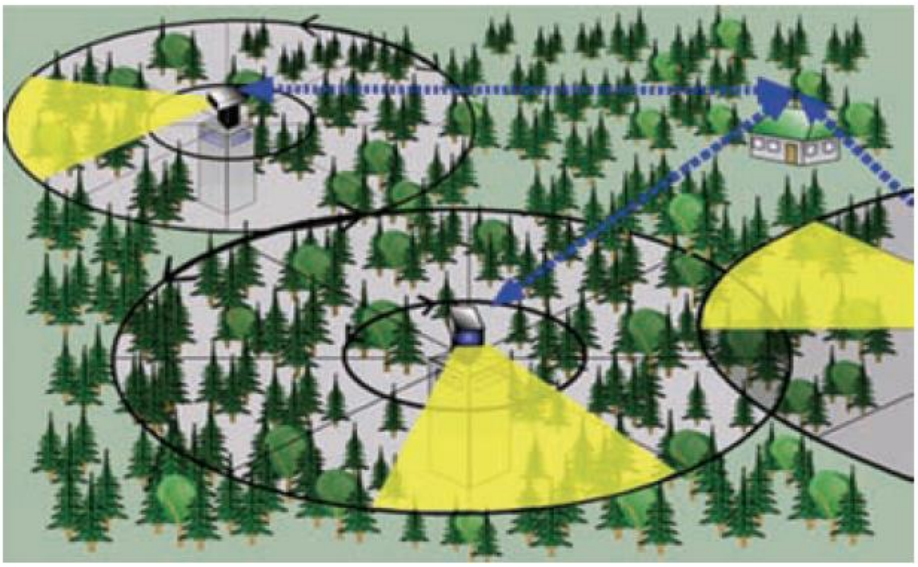
4.3.3. Fire Detection Technologies

The solution we have 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 will detect fire using three main approaches: flame detection, gas detection, and smoke detection.

Figure 19: Hydrogen sensor mounted to a tree during an experiment done in Humboldt University in Berlin, Germany. (Nörthemann, Bienge, Müller, & Moritz, 2013)

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) (Errynando Surya Sasmita, 2018). 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 (Fonollosa, Solorzano, & Marco, 2018). 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 (Fonollosa, Solorzano, & Marco, 2018).

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 (Alkhatib, 2017). 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 (Alkhatib, 2017). 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 (Alkhatib, 2017). Such optical systems are



usually integrated with local maps.

Figure 20: FireWatch adopts a similar concept to our method of scattering sensors in a forest, except their system uses cameras (Alkhatib, 2017)

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 (Aslan, Korpeoglu, & Ulusoy, 2012). A wireless sensor network mainly includes numerous sensor devices that typically use low power, low processing memory, and low bandwidths (Bouckaert, Poorter, Mil, Moerman, & Demeester, 2009) 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” (Aslan, Korpeoglu, & Ulusoy, 2012; Bouckaert, Poorter, Mil, Moerman, & Demeester, 2009). 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 (IEEE, 2018).

4.3.3.1. Potential sensor components.

Based on the research, the following components were selected as preferable sensors to be used in the system based on electrical characteristics (supply voltage), cost, I2C compatibility, and principle of detection. Sensor selection will not be finalized until potential sensors have been tested to understand their capabilities and performances. 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 Converter	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 temperatur	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.

e sensor combo module					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 sensors for $30	Alcohol, LPG, Methane, Carbon Monoxide, Hydrogen. Gas concentrations 200 to 10000ppm.
Gravity: Analog Gas Sensor (MQ2)	20℃-50℃	Analog output		$6.90 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 $20.82	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	Up to 80℃	SPI or I2C		1 for $22.50	Temperature, humidity, barometric pressure, and VOC gas. Must be calibrated. Detect gasses & alcohols such as Ethanol, Alcohol and Carbon. Must be calibrated

and Gas Sensor					Humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy.

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 and Timer Mode	-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 photoelectric-type smoke detector

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	3V or 5V micro controller		1 for $39.95	8x8 array of IR thermal sensors. 64 individual infrared temperature readings over I2C. Detect a human from a

	to 80°C	or computer.			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°

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 & Solidworks
Fusion 360 Student Edition was used to CAD and render the four preliminary mechanical designs for this project. 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 allows us to design parts that might need to be 3D printed or machined so that we can complete this task ourselves or have it created for us.

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. This will let us send the data to a manufacturing house and they will manufacture our PCBs for us. 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. This means we can get the most up to date data about the electrical components we will purchase.

4.3.5.2 Administrative Tools

Contained in this section are some of the administrative tools used for this project. Administrative tools are defined by us to be 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 our 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 we can always communicate if necessary. Once a group is created, we can talk, and chat and it does not seem as if there are any limits on file or image uploads as other chat applications may have. Furthermore, most of our 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 we chose to do meetings online. It provides conference calls, video calls, screen sharing, and file sharing to help us organize and meet up in an efficient way. It is also accessible through our phones and computers which allows us to communicate easily at any time.
The screen sharing feature lets us communicate and share our parts effectively during meetings. It also provides messaging and filing system to keep our works if needed. It can access our calendar by logging in using our knights' email. Since it is accessible by using our knights' email, we all have an account already made. Furthermore, it can utilize other Outlook features such as Word. We can also add other applications such as Trello as an add on.

Trello
Trello allows for task planning and scheduling so that everyone knows what project component is due at what time. This also allows each group member to schedule individual parts of the project so we can all see the big picture and stay organized. The many integrations of Trello allow us to do almost anything we want. Currently, we are using the Calendar integration so that it formats all our due dates into a calendar so that everything can be found quickly and easily. There is no confusion on when a deliverable is due.

In Trello, this is accomplished by creating a “Board” and then assigning “Cards” to that board. Each card can be assigned a due date and many different tags such as “In progress” or “Completed”. This allows us to see the status of each task and who is assigned as well as to store data from different meetings. We usually store meeting notes in Trello since all of our weekly meetings get scheduled there.

Microsoft OneDrive
OneDrive allows for use to easily share files together. The ease of one drive, compared to other tools like Google Drive is that OneDrive will sync files from the cloud to our computers directly. This means we can edit files and upload/download files directly from the file explorer. No need for a browser or external tool. The expansiveness of OneDrive also allows us to connect it to smart phones so that we can view documents on the fly. OneDrive has allowed us to just work on files and share files easily without an apparent “middleman” like Google.

Microsoft Word
Since we are using OneDrive, it became clear that we should also be using Microsoft office tools to work on our documentation. Word features a very comprehensive (but expansive) collaboration element so that we can all work on a document at the same time, but still have the power that Word provides normally offline. All the standard formatting tools exist, but in addition to that we can write comments and open up a collaborative chat with users currently editing a document. This means that we can quickly and easily discuss document changes and formatting changes without the need to go through other applications. It makes everything just that much easier to edit and work on.

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’s 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 we will use it for programming, debugging, and writing code for the SAMR35. Atmel Studio 7 features a programmer which is beneficial as the compiler, code editor, debugger, and programmer are all in one software package.

Python

Python is one of the popular programming languages known for its ease of use. It has the simpler syntax and format compared to other languages such as C or Java. It is the main language used in the recent computer vision applications and offers abundant libraries for us to implement them in our system. 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 us in debugging and constructing our code. It is easy to learn which will save us time and let us improve our system further.

C++

C and C++ are the standard languages used in embedded programming. As such, C++ will 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 will be easier to maintain the software we write, 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.

Atom

Atom is simply a text editor that provides syntax highlighting for different software. Some of the software and text documents will be 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 will become invaluable. Since it is a free and open source program, it will have many features that we will find useful. Putty supports many different communication protocols other than serial communication like Raw, Telnet, Rlogin, and SSH. Since we are using a Raspberry Pi, it may be useful to use the serial console through the UART pins of the Raspberry Pi or if we are using any Raspberry Pi that is not

the Raspberry Pi Zero for testing and debugging, we could use SSH to communicate with it via Ethernet, assuming we are using windows computers and may not have the ability to SSH.

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 world 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 (Rouse, 2020). This connection together allows for a bit of power 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 (Semtech, 2015) 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 (Semtech, 2015).

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 (Semtech, 2015). 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” (Ghoslya, n.d.).

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

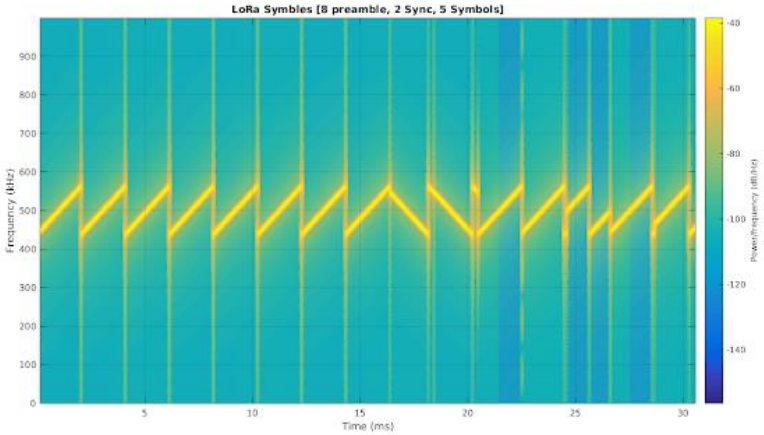


Figure 21: Spectrogram of LoRa physical layer (Ghoslya, n.d.)

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, will act autonomous but users still must interact with the system for the goal of the system to be successful. These uses are shown in the following sections.

5.1.1. Uses Case Diagram

Figure 19 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.

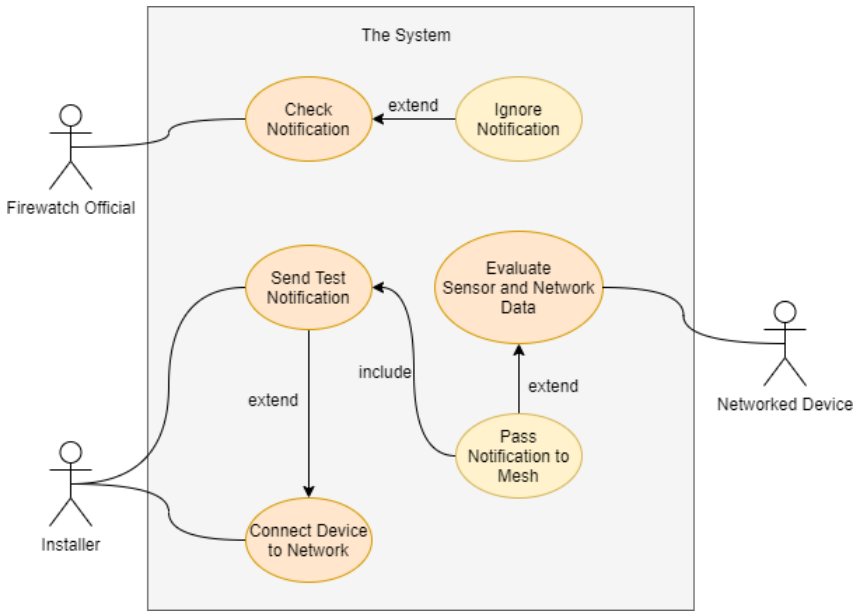


Figure 22: Use Case Diagram

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 will connect a device to the network and then may send a test notification throughout the system if he chooses. 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 will be present within the system. The hardware must work autonomously with very few failures at all times (day and night) to align with our design goals and motivations.

5.2.1. Hardware Block Diagram

The following diagram shows the hardware design sub-systems. There are 4 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 3 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.

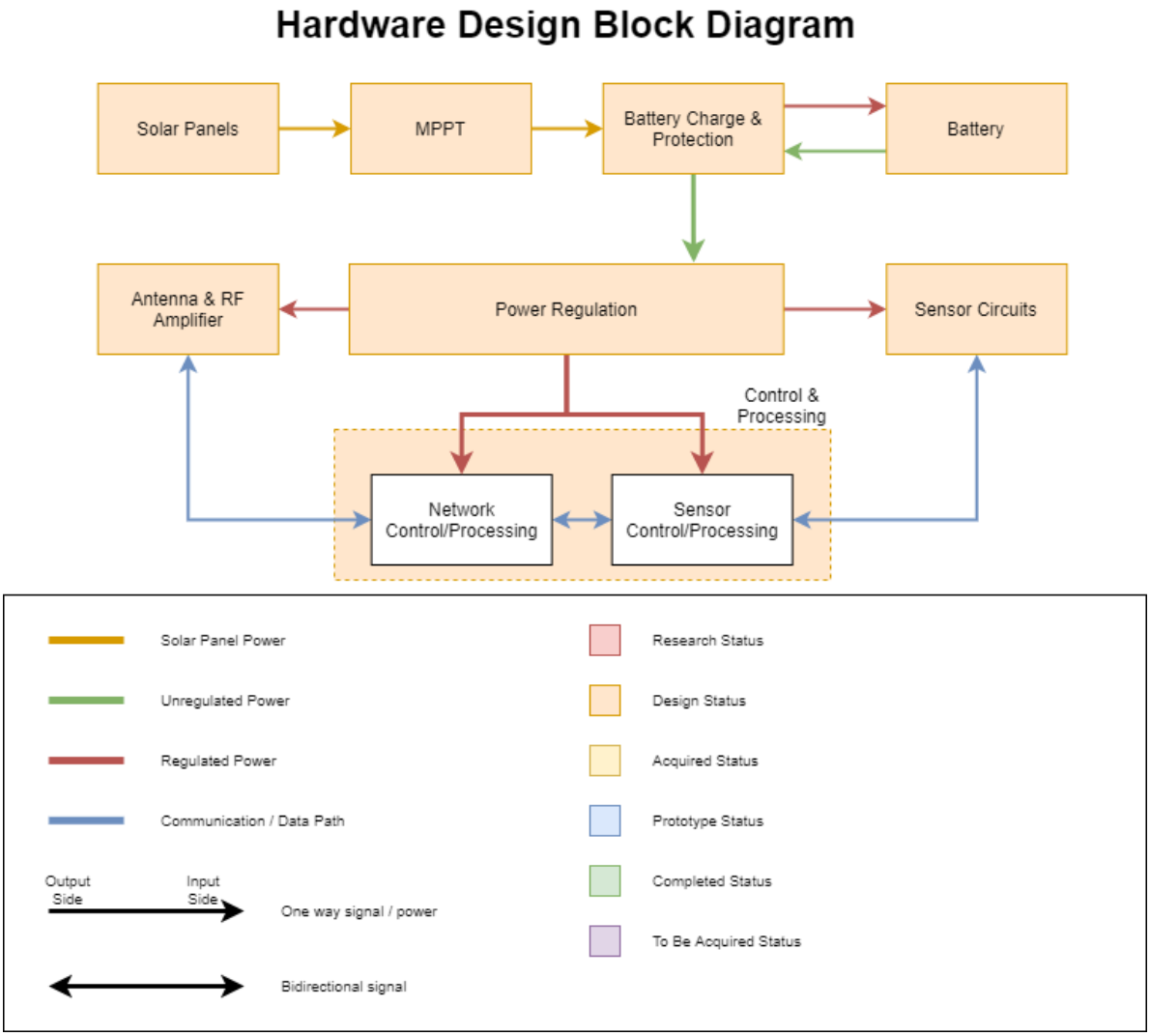


Figure 23: Hardware Design Block Diagram

5.2.2. Microcontroller and Processing Device

The system will make use of two processing devices/controllers. The SAMR35 and the Raspberry Pi Zero. These two devices will allow for much simpler interfacing and separation of responsibilities for the system. This will also allow our power consumption to be a minimum during down time but the simplicity of programming for our up time. The SAMR35's responsibilities include the Network control and processing. It will handle communications within the LoRa network and the connection to the network. The Raspberry Pi Zero will facilitate the sensor readings and the formation of packets. The

root node must contain a SAMR35 to process root node packets, but once it retrieves the data out of the SAMR35, any device can be used (such as another embedded system or a standalone computer). The SAMR35 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 complient with IEEE802.15.4g. Since this module is built in to the SAMR35, it does not have to be a standalone device.

5.2.3. Hardware Schematics

The following sections are descriptions and diagrams of the preliminary 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

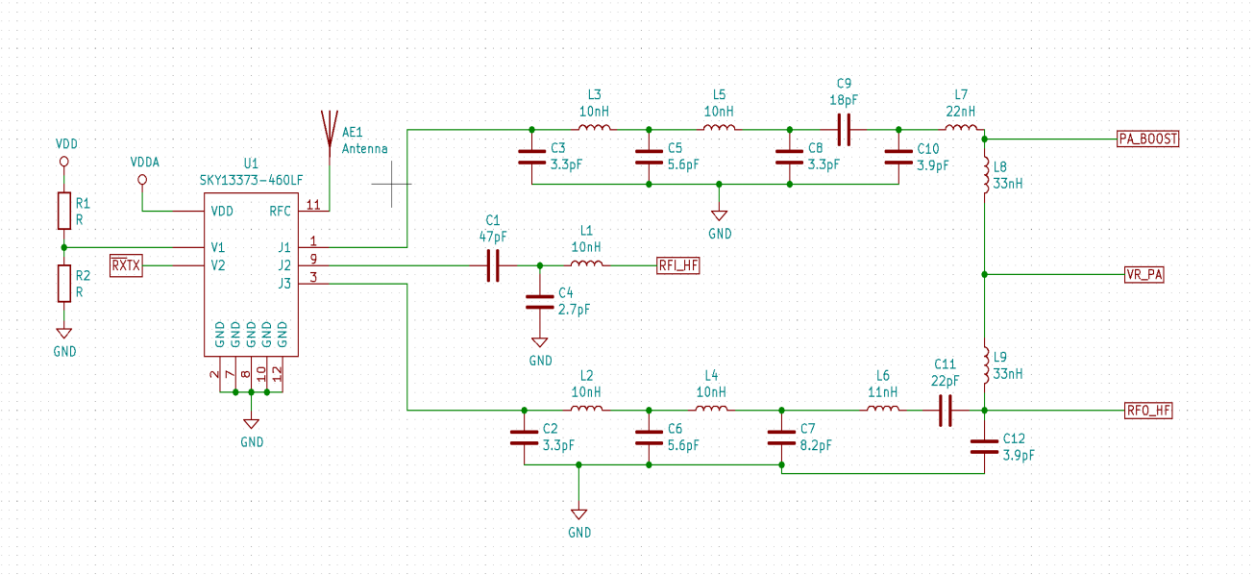


Figure 24: RF Switch Schematic

The schematic above in **Figure 21** handles switching the RF signals so that we can use a single antenna. In RX mode, the circuit is slightly different in the way it filters everything than in TX mode. Furthermore, this allows us to have two separate circuits on the TX side to select between different bands. For now, the idea is to solder a resistor between GND or VDD to select the band. The schematic for the SAMR35 in **Figure 22** is simple as of right now, since it is only the minimal parts to facilitate the RF side of everything and the chip function. It has the UART ports accessible as well as connections to the RF switch. Everything must be 50-Ohm impedance matched.

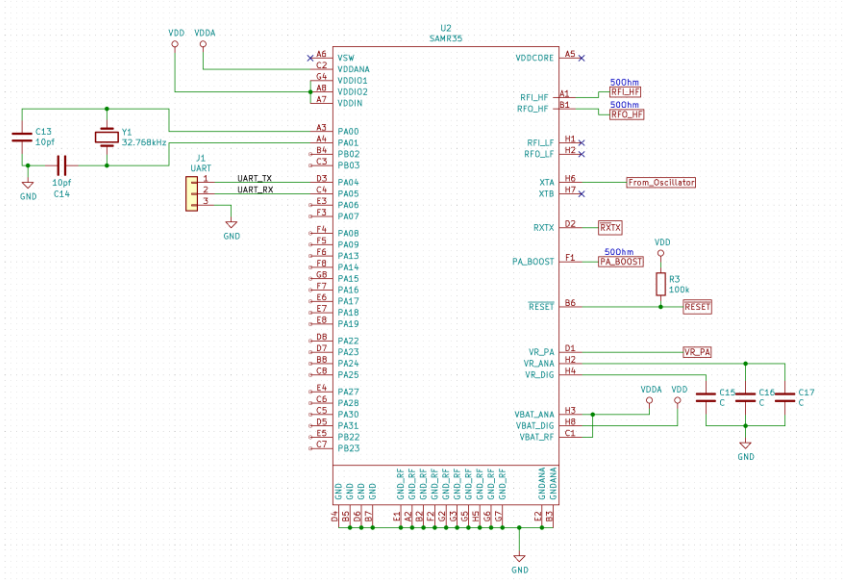


Figure 25: SAMR35 Preliminary Schematic

5.2.3.2. Preliminary Raspberry Pi System Schematics

Since the SAMR35 will do a lot of the RF work, the Raspberry Pi will do a lot of the heavy lifting for interacting with the sensors. It will be connected to the SAMR35 via serial connections and to all the sensors via I2C or SPI connections. This connection will allow it to easily collect data from all the sensors. The connections are shown in **Figure 23**. 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. It is possible though, if power consumption becomes an issue, that the SAMR35 does all the sensor readings and reports them to the Pi when applicable.

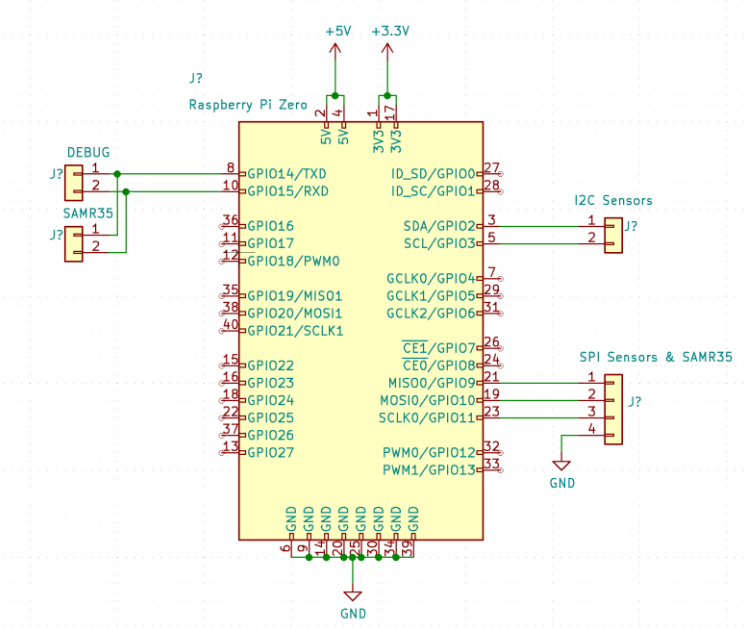


Figure 26: Raspberry Pi Connection Preliminary Schematic

5.2.3.3. Preliminary Power System Schematics

The initial design for the power system is to have the solar panel array hooked to a voltage simulator that regulates the 12-volt nominal source to a 5 volt 2 amp max source. This is done to handle the irregularities with solar radiation levels throughout the day caused from varying reasons. This keeps the charging IC constantly running at the most efficient state. To maintain a constant output the right solar panel is needed for this system.

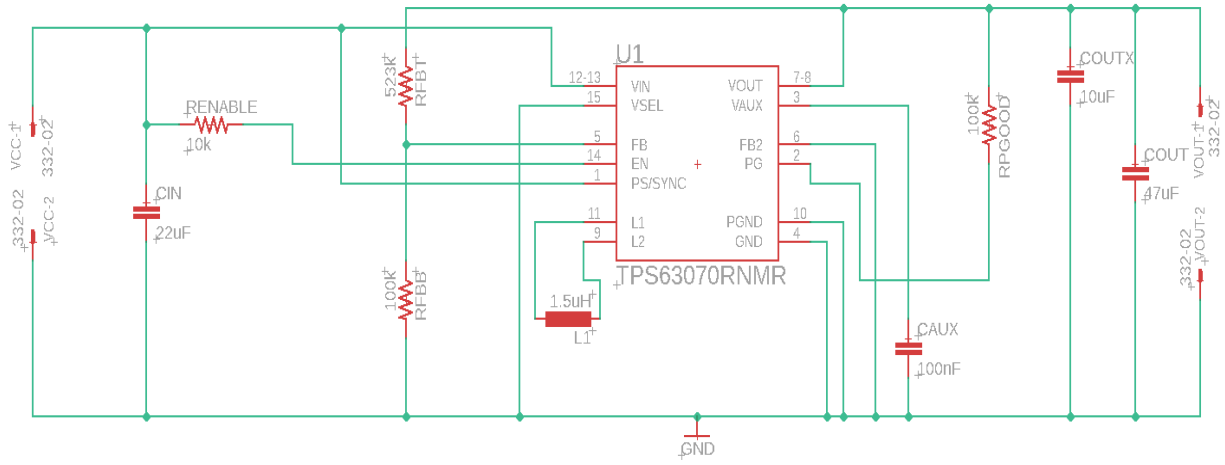


Figure 27: Voltage Regulator Schematic

For the solar panel itself the panel must be able to 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, a panel with many solar cells in series and parallel is needed to help with these issues. A panel similar in design to the one below 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.

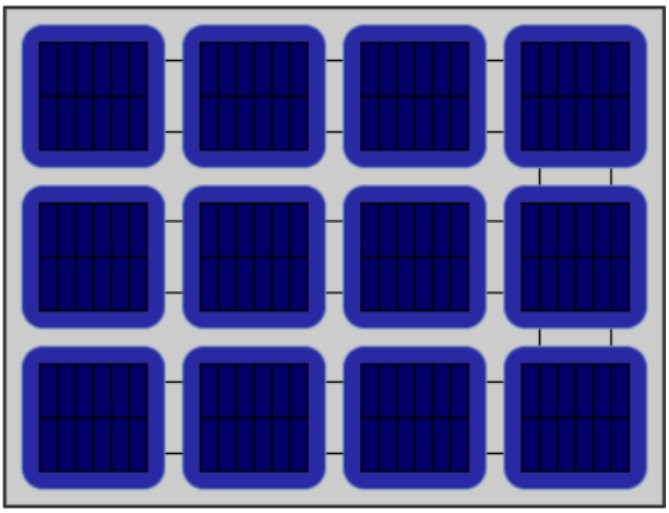


Figure 28: Top View Solar Panels

5.2.3.4. Preliminary Sensor Circuit Schematics

This section provides an overview of the initial sensor schematics for gas, smoke, and flame detection. These sensors are still undergoing examination and testing to determine their feasibility for the system.

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.

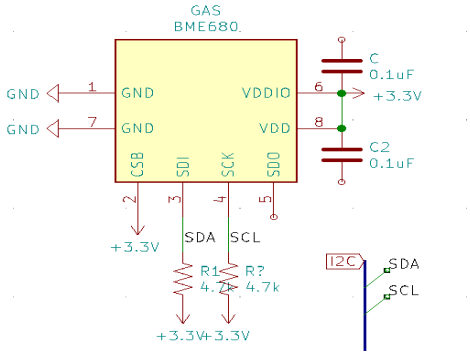


Figure 29: Gas Sensor Schematic

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[9]	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 30: Air Quality Table (Bosch)

Smoke Sensor

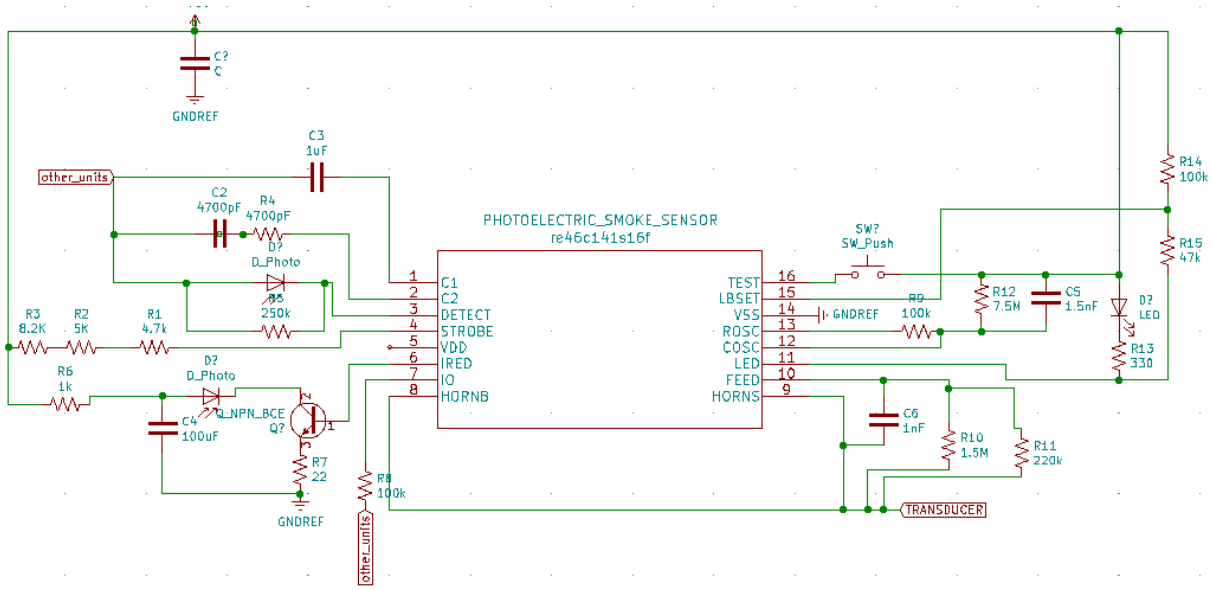


Figure 31: Smoke Sensor Schematic

For smoke detection, the RE46C141 CMOS photoelectric smoke detector will potentially be integrated into the system. 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. This device requires an supply voltage for approximately 12.5V, which must be taken into consideration when designing the power supply rails. The RE46C190 may be used instead since it fulfils the power requirement and is also designed for similar applications.

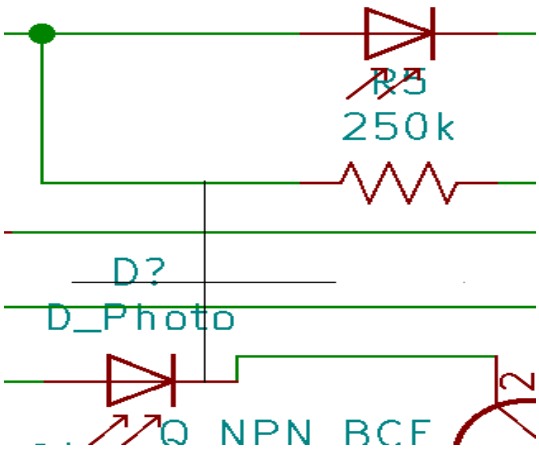


Figure 32: Smoke chamber

Flame Sensor

Two methods of flame detection will be used: non-visual and visual techniques. The non-visual technique will use the ADPD2140 infrared light angle sensor by Analog. This sensor includes a sensor array for 2-axis light angle measurement. It is able to provide a linear response to the angle of incident light within +/- 5° with an angular field of +/- 35°. This sensor is typically used in robotics to allow the robot to follow a beacon of LED or remote emitting infrared light for the robot to follow. In this application, the beacon will be the fire emitting infrared waves. The sensor can detect infrared rays from 800nm to 1080nm. More importantly this device has a built-in visible light blocking optical filter that is able to filter unwanted visible light such as sunlight and indoor lighting.

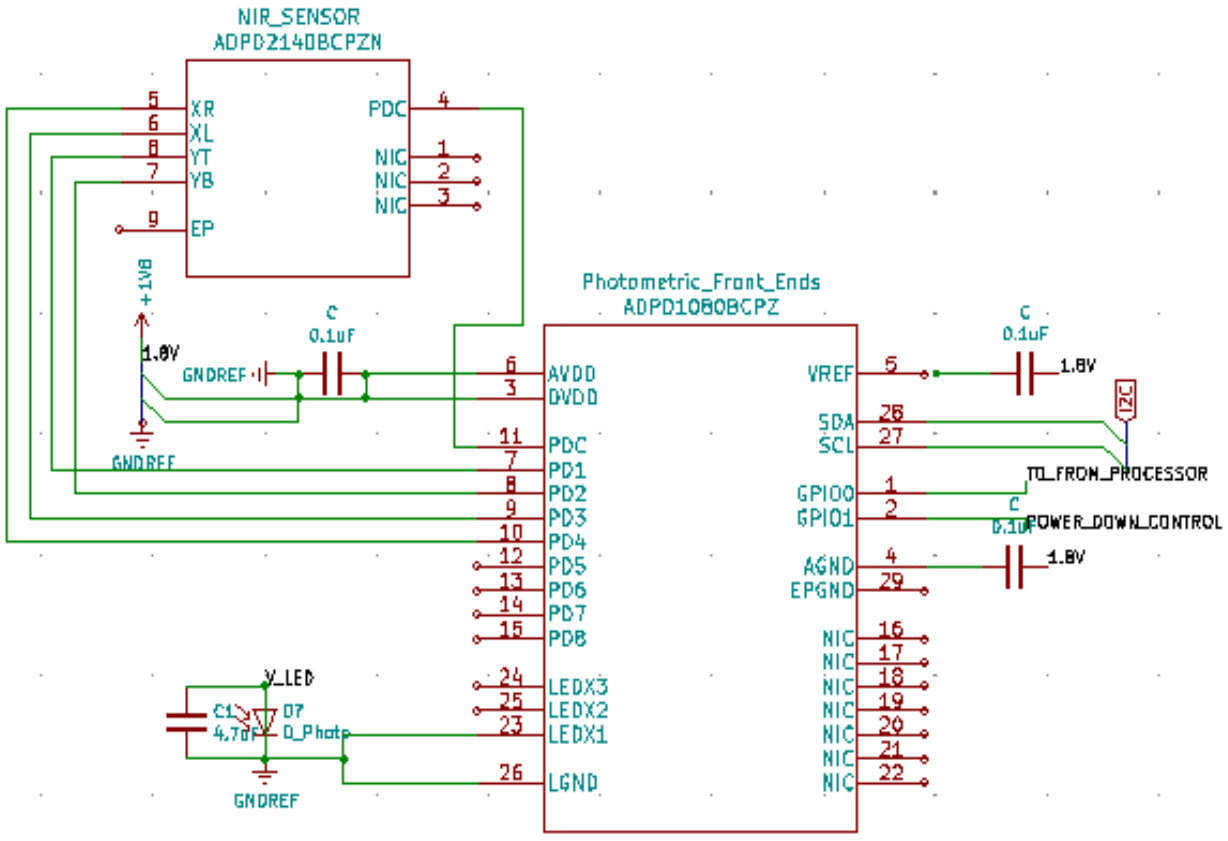


Figure 33: NIR Sensor Schematic

The ADPD2140 will be connected to the ADPD1080 front end which will process the data from the sensor array. More importantly, the ADPD1080 will communicate using I2C to send the data to the processor for data processing. This combination allows for additional ambient light rejection, low power operation, and analog-to-digital conversion of the ADPD2140 output signals.

The thermal camera will potentially be used as a visual technique for flame detection depending on budget allowance. The MLX90640 thermal camera by Melexis Technologies is small and low cost 32x24 pixel infrared sensor array that is I2C compatible. It can operate from -40°C - 85°C and is able to detect temperatures from -40°C - 300°C. The sensors come readily calibrated from the factory and contain 768 FIR pixels. Below is a diagram while illustrates the pixel position and the whole field of view. The field of view for the X axis can be in 110° or 55°, and for the Y-axis 75° or 35°, respectively. The noise of the pixels for high temperatures is lower than that for low temperatures. Pixels in the corner of the frame will be nosier compared to the sensors in the middle of the frame.

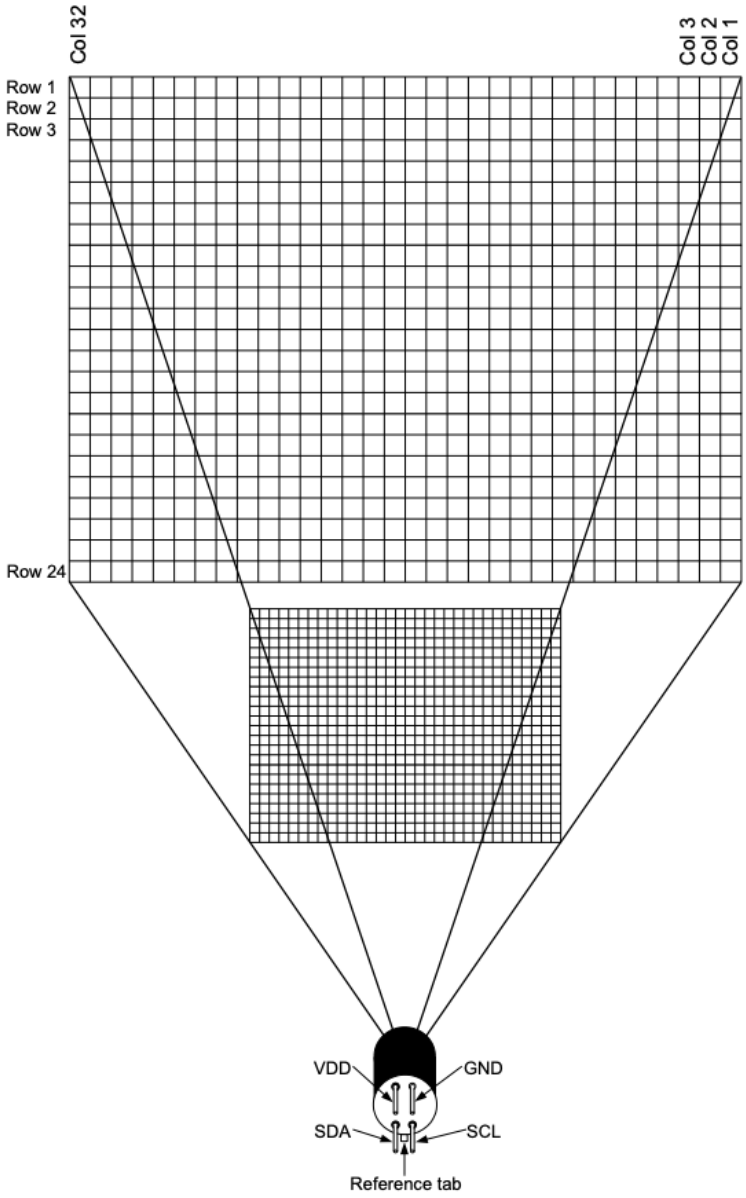


Figure 34: Thermal Camera Sensor Array (Melexis, 2012)

The schematic below is a simple representation of the electrical design. A 1k ohm resistor will most likely need to be connected between the SCL and the MCU input, as well as between SDA and the MCU. A 10uF and 100nF capacitors will eventually be integrated to VDD in parallel as recommended by the datasheet.

The device can be used to calculate temperature a well as used in “image mode”. This would allow for the thermal image to be obtained which will be used for visual flame detection. The computation flow for this would include calculating supply voltage for the pixels, ambient temperature, gain compensation, IR data compensation, IR data gradient compensation, normalizing to sensitivity, and finally image (data) processing. Image mode opens the scope of integrating concepts and techniques from machine learning to train the system to identify flames.

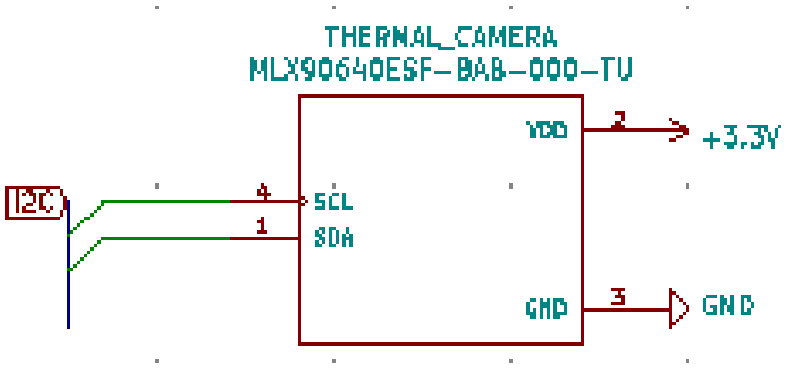


Figure 35: Thermal Camera Schematic

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 only 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 is designed in such a way that it is mounted to a tree or other tall structure. The first design in **Figure 21** 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.

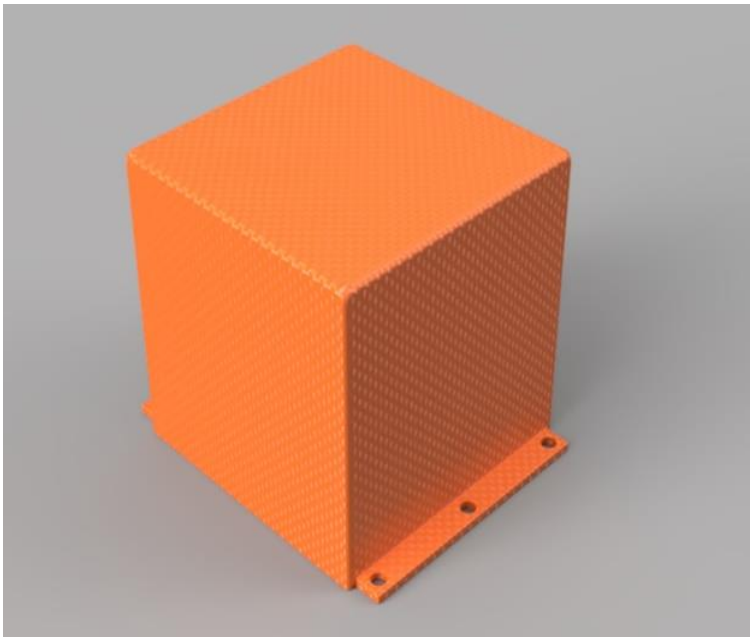


Figure 36: Mechanical Design A

The second design, shown in **Figure 22**, 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 the simplest design to install as it just requires one point to lock the mechanism to the surface. A downside to this design would present itself for items 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.

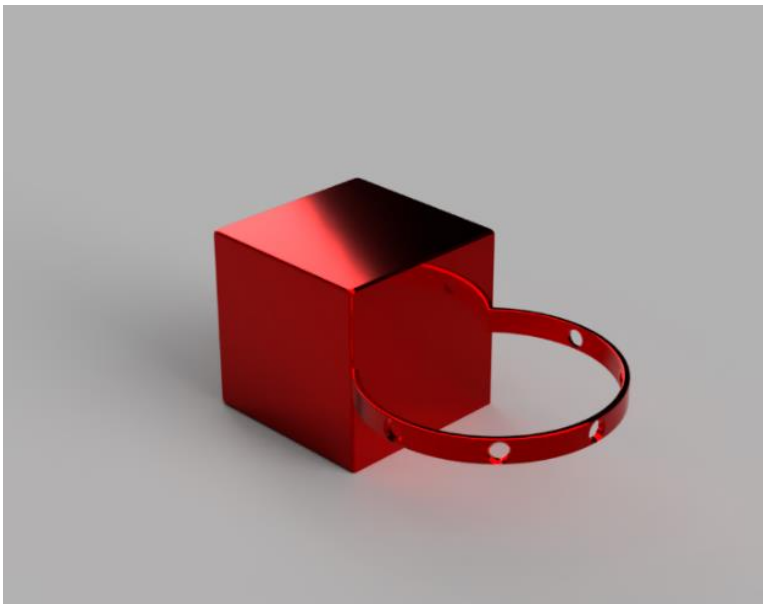


Figure 37: Mechanical Design B

The design idea represented in **Figure 23** 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.

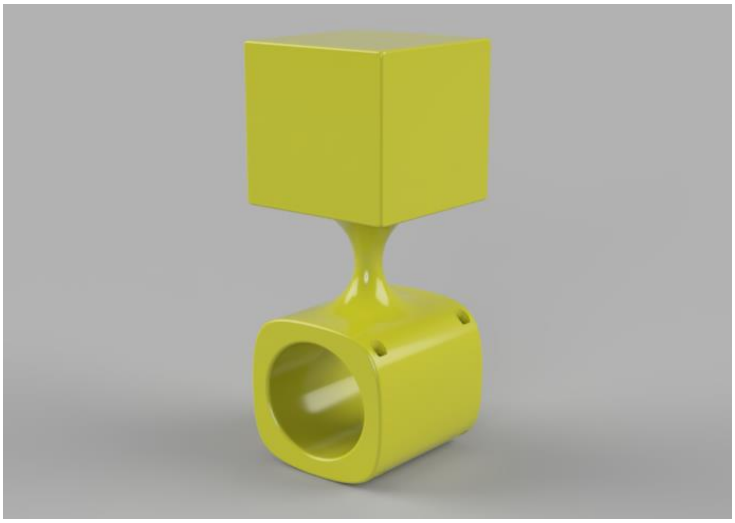


Figure 38: Mechanical Design C

Finally, the last design shown in **Figure 24** 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.

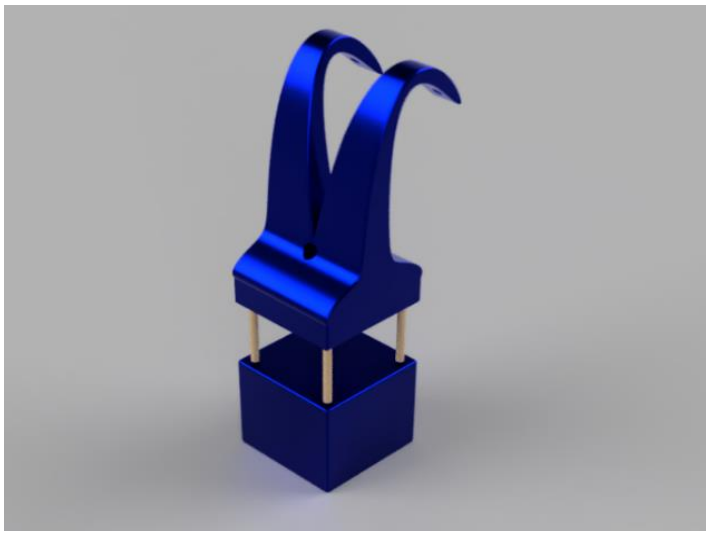


Figure 39: Mechanical Design D

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 subsections discuss the software designs and the methodology in place to make the controllers do their job.

5.3.1. Design Methodology

The design methodology that we are to employ 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.

5.3.2. Software Block Diagram

The software block diagram shown in the figure below 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. In reality, the “Main Loop” and the Network side of the diagram will be managed by the SAMR35 and the Sensor Data side of the diagram will be 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.

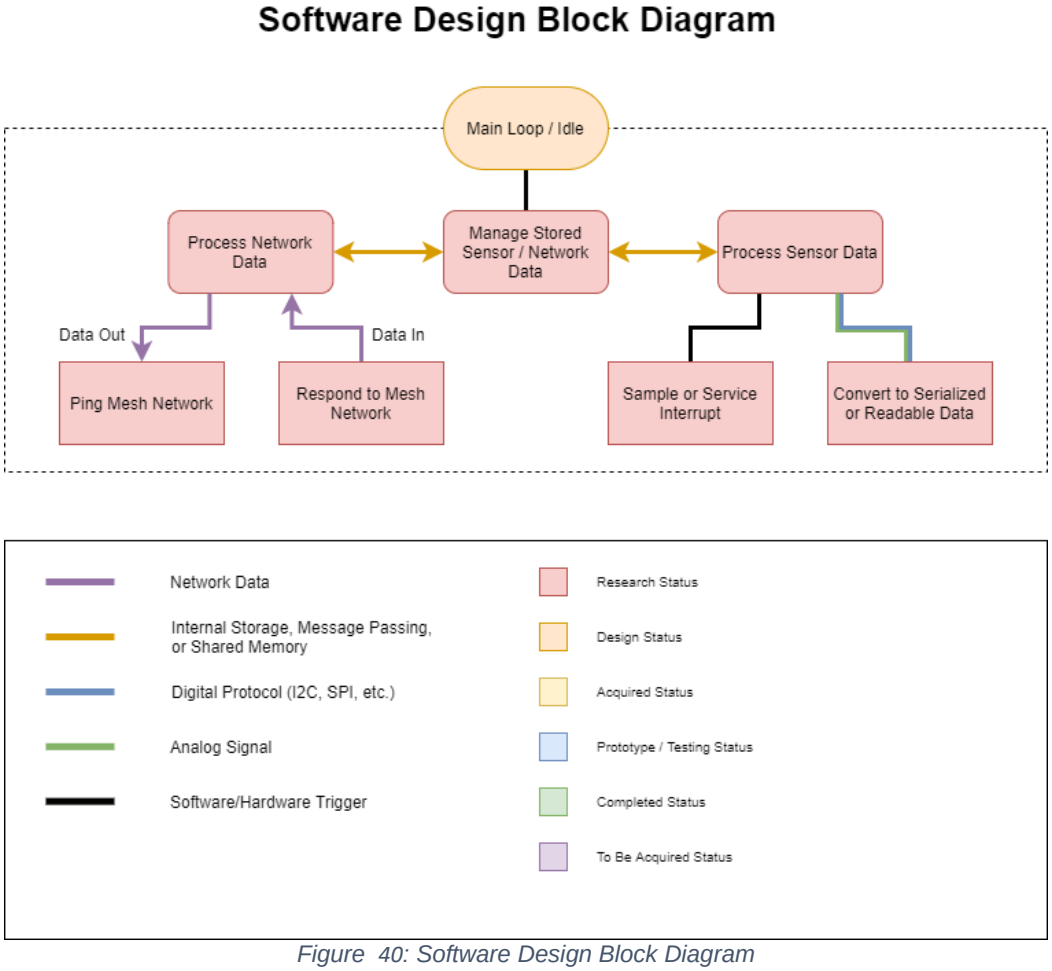


Figure 40: Software Design Block Diagram

5.3.3. Network Software

5.3.3.1. Network Flow

This section describes the “Network Flow” of the system. The network, in normal circumstances, will not be 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. The image below describes a Join Request case.

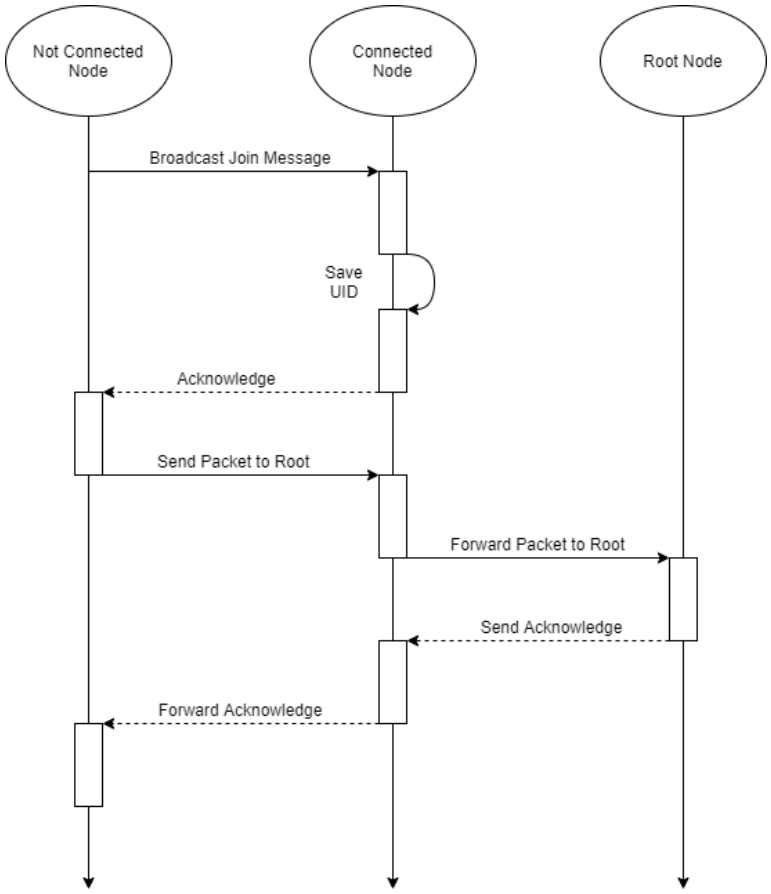


Figure 41: Join Request flow diagram

When a Join Request is received the currently “Not Connected Node” Will broadcast a Join Message and all connected nodes in the vicinity will 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 will send a generic message to the root 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 will now attempt to forward packets in the system. A “Not Connected Node” can send messages through the network but will not forward messages through it. Acknowledgements are generally not required in our mech scheme, but without them, there is no way for the node to know that it is correctly connected to the network.

In this section, only two figures are present to describe the flow of packets in the network. Other kinds of messages follow similar protocols as these two messages in some way. In one of the flow charts in section 5.3.4 Software Events & Flow, a generic case is reviewed where a generic message is received. The figure below describes a “Fire Packet” which follows a similar protocol as the generic case, but with some extra work.

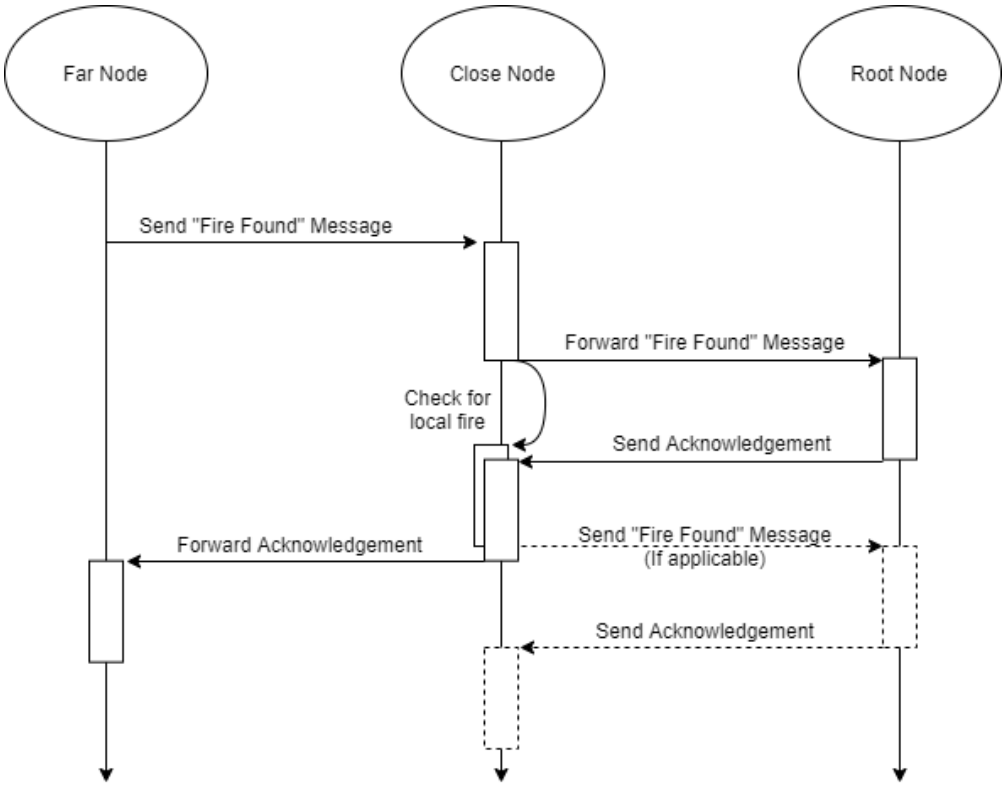


Figure 42: Fire Packet flow diagram

When a fire is detected at a “far node”, that node will send a message to the root alerting of the fire to the nodes closest to it in a broadcast. These “close nodes” will forward the fire found message to the root node. For every valid fire packet received, the node will wake up its sensors and begin looking for a fire to report. In this time, it will continue to normally forward packets as necessary. If the “close nodes” detect fires in their area as well, they will send the “fire found” message to the root node as well. When the original “far node” receives an acknowledgement, it will stop sending “Fire Found” packets regularly. This “stop” condition is based on two things: Acknowledgements and Timeouts. To stop the network from getting busy, the sending nodes of a fire message will stop sending packets if and only if an acknowledgement to stop has been received or a timeout. Eventually, all the nodes will stop reporting the fire continuously and will wait some amount of time before reporting the fire.

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 will be necessary.

First, it is important to understand that each node will maintain a packet buffer to store received messages before sending. The node will hold onto these packets until it is time to forward them. This buffer only contains packets that need to be transmitted, not packets that are invalid or are internally processed. This buffer will need 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) will 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 multiply 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 sounds 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. Packets going through the system will 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 will 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 point. 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 will 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 will persist for some time to avoid packets getting through. 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. Acknowledgements will get forwarded along this path. It is a stretch goal that the original join request and subsequent heartbeat packets will determine this ideal path and will 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 will not, initially, support load balancing of this form unless it is found to be necessary within the algorithm to do so.

5.3.3.2. Network State Machine

This section discusses the different states that the network controller has. 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.

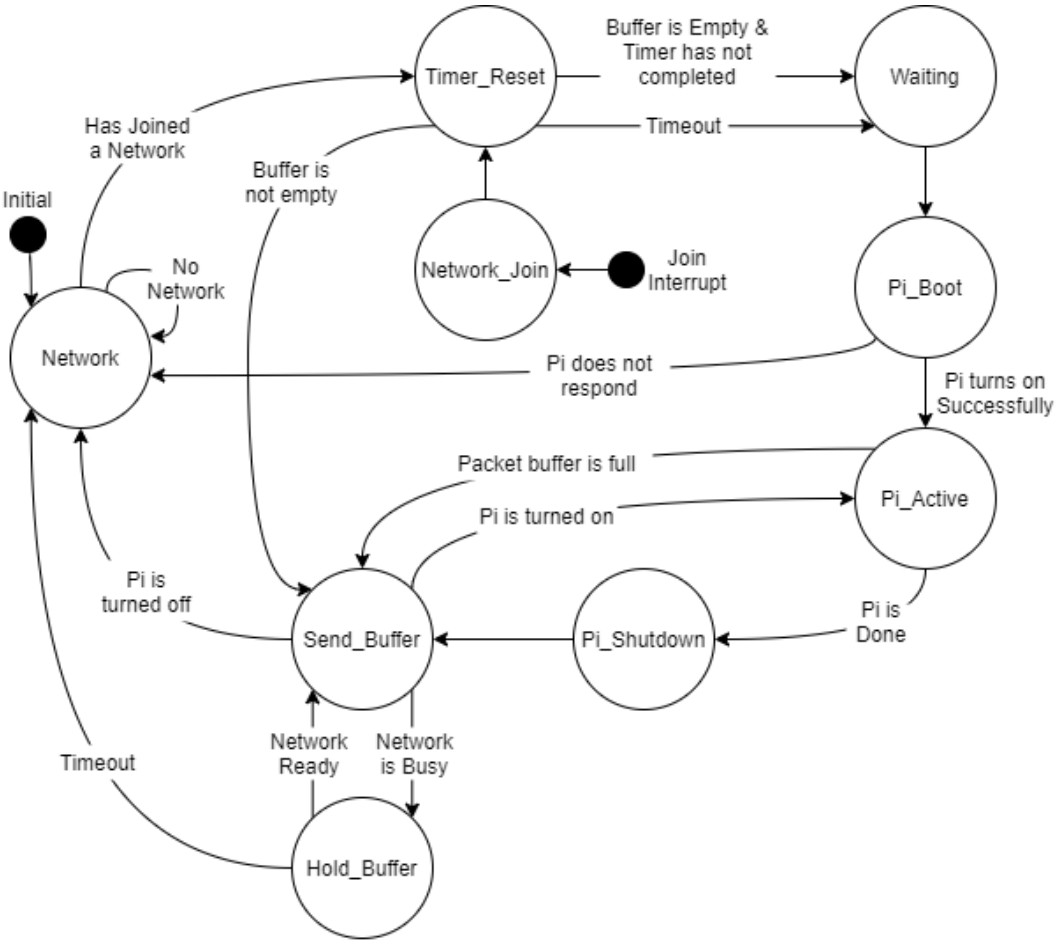


Figure 43: Network Control State Diagram

The state transitions are complex and there are many of them. Structuring the software in this way allows each individual state to be simple code compared to complex code.
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 we are 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. Interrupts will be discussed in a later section. After the Network state, the system transitions to “Timer_Reset” which resets the timer to its delay time.

From here, if there are packets to send then the system will do so in the “Send_Buffer” state. If everything is ready to go, or the system has been sending packets for too long then the “Waiting” state is invoked. During this state, nothing happens. A loop will wait until the timer is done running to move on. The next state in normal operation is the “Pi_Boot” state. This state will turn on the Raspberry Pi and will wait for the Raspberry Pi to acknowledge that it has turned on. If the raspberry pi does not respond, within some amount of time, the network controller will transition back to the Network state. Otherwise, with a successful turn on sequence, the network controller transitions to “Pi_Active”.

This state handles servicing the Request Queue and the request-respond structure. After each request, a response is expected from the Raspberry Pi. This response, in turn, gets immediately stored in the packet buffer. This is repeated until the packet buffer is full, or there is no more data to send. From that point, the network controller will go into the “Pi_Shutdown” state and send a “Finished” request and will only wait for the shutdown notification. The Raspberry Pi may do anything until that point. The once the Pi responds, the controller will enter the “Send_Buffer” state and attempt to empty the packet buffer. Assuming everything is good and the buffer is empty, the controller will transition back to the “Network” state and everything begins again. If the network is busy during the “Send_Buffer” state, then the controller might transition to the “Hold_Buffer” state for some time. If it cannot get a chance to empty the packet queue within some amount of time then a timeout will occur and the Network state will immediately be transitioned to. The table below describes the different states in a more concise manner.

State	Next State	Transition Condition	Previous State	Description
Network	Timer_Reset	A network has been joined	Initialization	Saves Network State and other tasks
Timer_Reset	Waiting	Buffer is empty & Timer is not completed or Timeout	Network	Starts Timer for waking up Raspberry Pi
	Send_Buffer	Buffer is not empty		
Network_Join	Timer_Reset	A network has been joined	Join Interrupt	Joins the Network
Waiting	Pi_Boot	Timer interrupt is fired	Timer_Reset	Waits for interrupt to fire from timer
Pi_Boot	Pi_Active	Pi responds	Waiting	Turn on the Raspberry Pi
	Network	Pi does not respond		
Pi_Active	Send_Buffer	Packet buffer is full	Pi_Boot Send_Buffer	

	Pi_Shutdown	Done sending data		Communicate with Raspberry Pi
Pi_Shutdown	Send_Buffer	Pi sends shutdown notification	Pi_Active	Turn off the Raspberry Pi
Send_Buffer	Hold_Buffer	Network busy	Pi_Shutdown Pi_Active Timer_Reset Hold_Buffer	Send packets from the packet buffer
	Network	Pi is turned off		
	Pi_Active	Pi is turned on		
Hold_Buffer	Send_Buffer	Network ready	Send_Buffer	Wait until network is ready
	Network	Timeout		

Table 10: State Transitions

5.3.3.3. Interrupt Based Events

The state machine described in the previous section is the main portion of the network controller. The other part of the network controller is related to the interrupts fired. There will be a few interrupts that are needed to handle everything directly. The LoRa transceiver the system uses only contains a 256 byte RAM data buffer. This means that when data is received by the LoRa modem, we must service that data immediately. The system can monitor up to 5 interrupts in the LoRa transceiver with configurable IO lines, as well as different interrupts directly relating to the SAMR35. When an interrupt is detected from the LoRa transceiver, an interrupt service routine will be fired that reads the transceiver's data and stores it in memory. There are three ways to handle incoming data.

The optimal solution will most likely be a mix of the three. The first way is to immediately service the packet. Packets that are to be forwarded will most like be serviced immediately and attempted to be forwarded. The second way is to service the packet but store the network packet response in the packet buffer and send it later in a different state. Likely, this is the method to handle non-forwarded packets. They will be serviced eventually, but not immediately. This may help with network congestion as there will be a decent delay before packets are sent through the network. Lastly, the received packets could be stored locally and ignored at first. Eventually these packets would get serviced in a dedicated state for them. This option will only be used if there is a significant amount of time lost by the waiting state in which this time will be used to process the packets. For now, received packets will be immediately serviced with network activity only happening for forwarded packets.

The second kind of interrupt that is generated is a button that causes the system to attempt to join a network. When the button is pushed the "Join" protocol is initiated. Overall, a node should never be added to the network unless there is a user attempting to install new nodes. In the nominal case, nodes do not need to join the network by themselves. Pushing this button will reset all the network data and re-randomize any of the unique values/keys in the system. It does this by setting the next state to be the "Network_Join" state. This state handles all of the join protocol and clearing of values.

In conclusion, there are only two major interrupt service routines that may be executed at any time: Received packets and Join Protocol. Any other interrupts that are monitored will be polled in their registers and used as a wait/delay.

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.

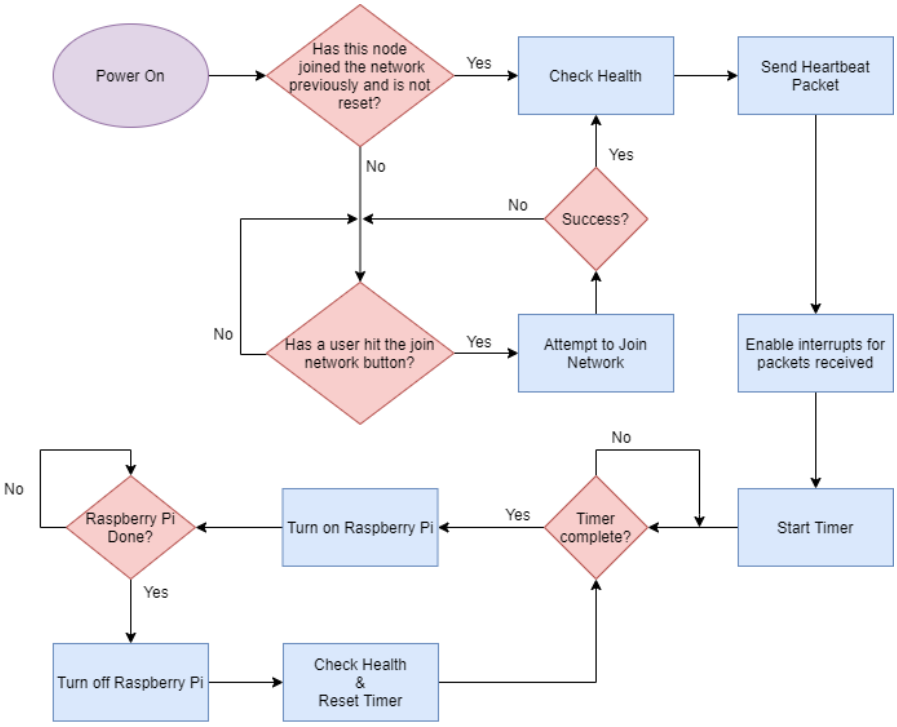


Figure 44: General software flow when power is applied to the system

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 can not receive messages. In this case, it can still send messages even though it is not “joined” to the network. If the system was not reset or trying to join a network, then it waits for a user to initiate a join by pushing a button on the device. When pushed (unless requested by the root node to join the network), the system will send a join request to the network. During this time the node will take 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.

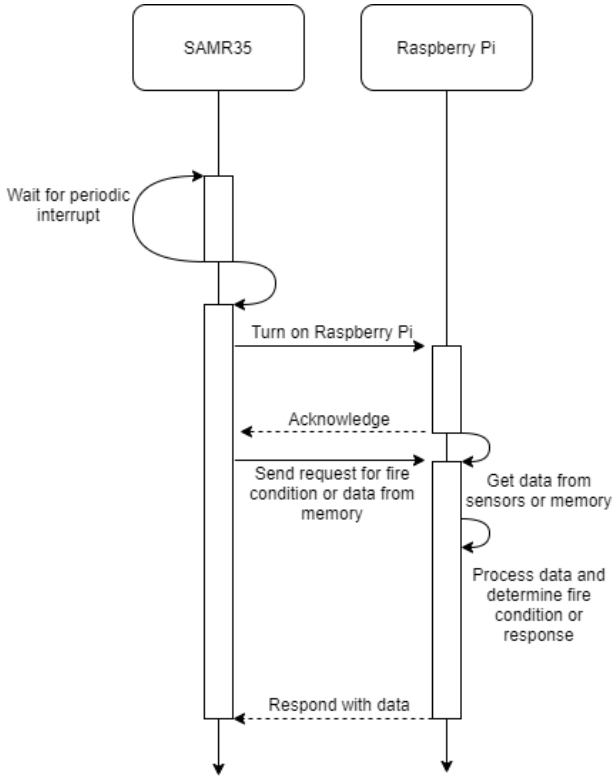


Figure 45: Raspberry Pi Flow

Once the system has been turned on and joined to a network, it will initialize its interrupts and timers and begin a timer. This timer will be set to turn on 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 Raspberry Pi is turned on, it will begin reading its sensors and determining if there is a fire. The Raspberry Pi will then report this data to the network controller and the network controller will decide whether or not to transmit the data. During this time, after it has determined if there is a fire or not, the Network controller may also pass packets to the Raspberry Pi if deemed necessary by other packets sent from the root node. This may be control packets or requests for data. The data will be accumulated by the Network Controller from the Raspberry Pi and it will then tell the Raspberry Pi it is done with it. The Raspberry Pi will determine if it is to shutdown or not and then will alert the SAMR35 that it is shutting down. When it is shutdown, the SAMR35 will remove power to the Raspberry Pi, as shown in the figure below.

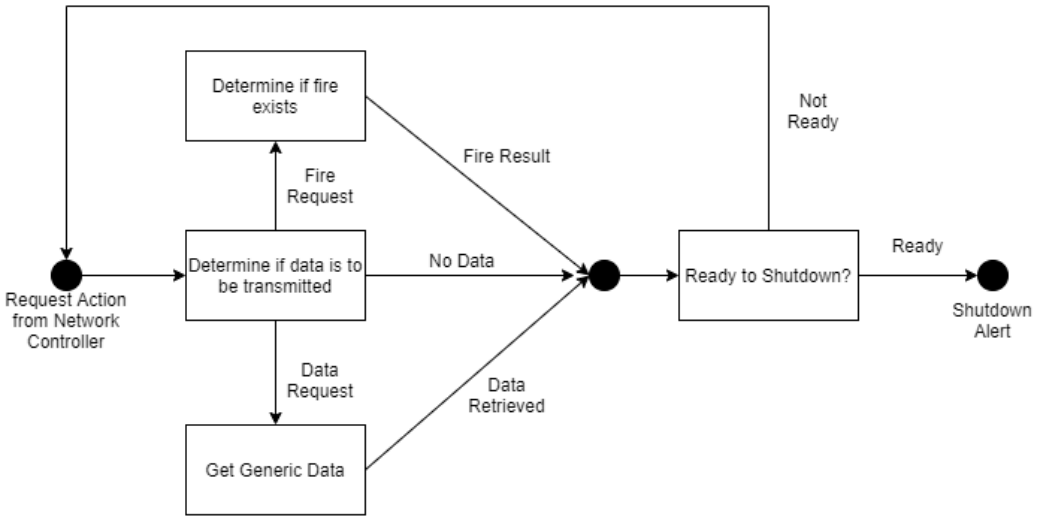


Figure 46: Raspberry Pi - Network Controller Communication Diagram

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. This will only be implemented if time permits or it is deemed necessary due to too many error events with communication over wireless. Otherwise the packet is rejected, and no action is taken. The next conditions that matter are 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.

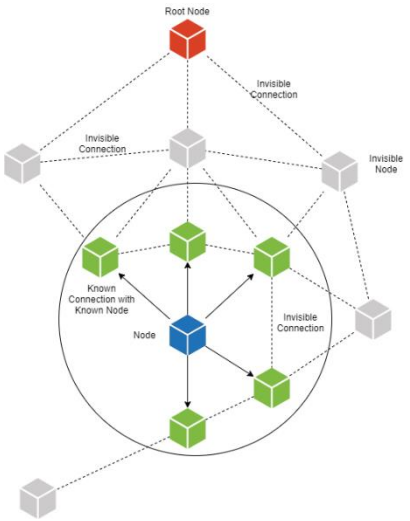


Figure 47: Known Connections Diagram – Mesh

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.

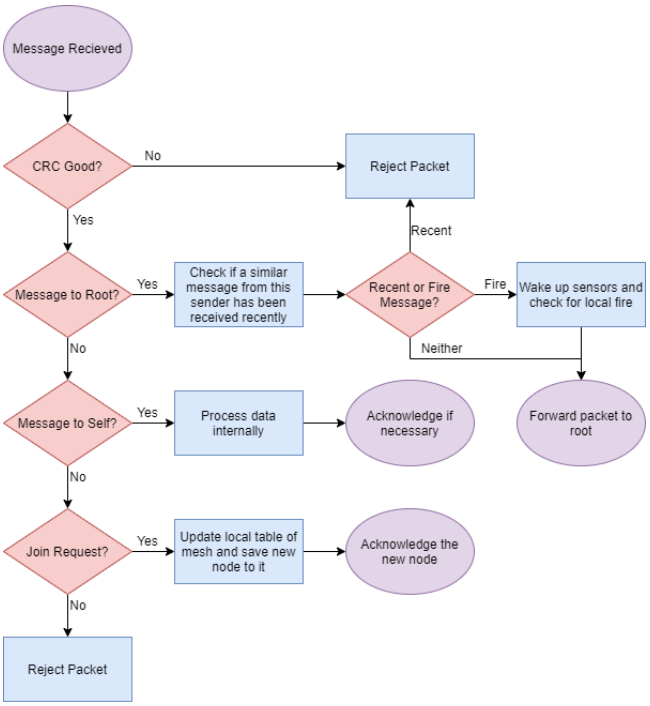


Figure 48: Actions taken on a Message Received event

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 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 grouped up on the network controller and sent, one at a time, to the Raspberry Pi to process. Once the Raspberry Pi has serviced all the tasks sent to it, the network controller will send a “Finished” request. This request informs the Raspberry Pi that the network controller has nothing else to request and that the requests have stopped until the next startup. It is up to the Raspberry Pi to decide if it is to shutdown or read sensors or do whatever it needs to do. Prior to shutdown, the Raspberry Pi will send a “shutdown notification” to the network controller to inform it that it is done. From this point, the Raspberry Pi does not need to wait for the network controller and can immediately shutdown.

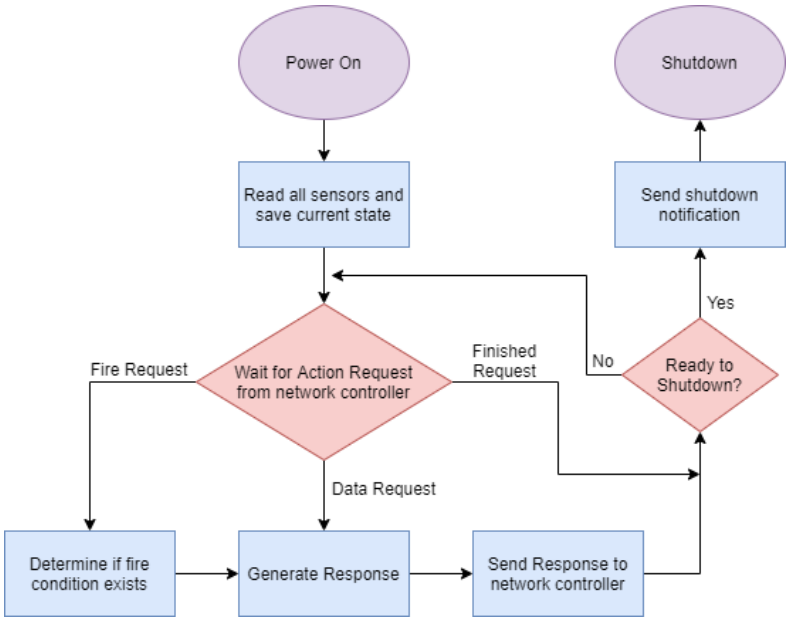


Figure 49: Raspberry Pi decision making

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

The system will store some information in non-volatile memory to ensure that upon power loss all configuration items stay intact. Power is removed from the Sensor Controller after it decides that it no longer has any tasks to complete. Therefore, 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 Controller, on the other hand, only saves its network map to non-volatile memory and some statistic counters. 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 by the people 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.

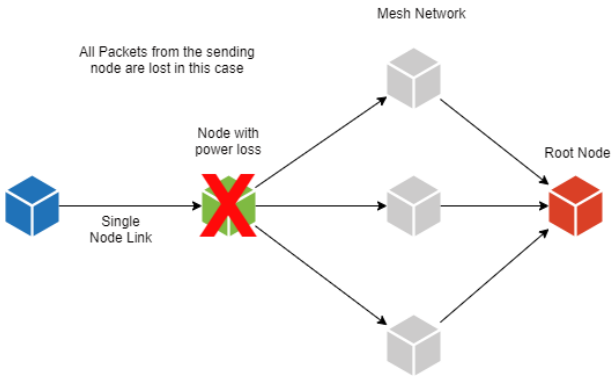


Figure 50: Lost Packets Diagram

The Raspberry Pi will also store data in a non-volatile way. This is important because the models used to detect fires 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. Before shutdown, it will save any data it needs to, to its SD card so that it can load it on startup.

5.3.6. Network Packet Types

To transmit and understand information effectively, the system will utilize 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 will respond and let 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 area. The heartbeat can receive an acknowledgement so that the sender knows it is still in the network and can decide which nodes are ideal to send to. 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. Machine Learning

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. This section will discuss the different kinds of methods to implement machine learning.

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 power consumption while being able to output good performance fast enough.

5.4.1. Methods

This section will cover different methods/models we considered to use for detecting fire using our system. These will cover different filters and adjustments we can do 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.

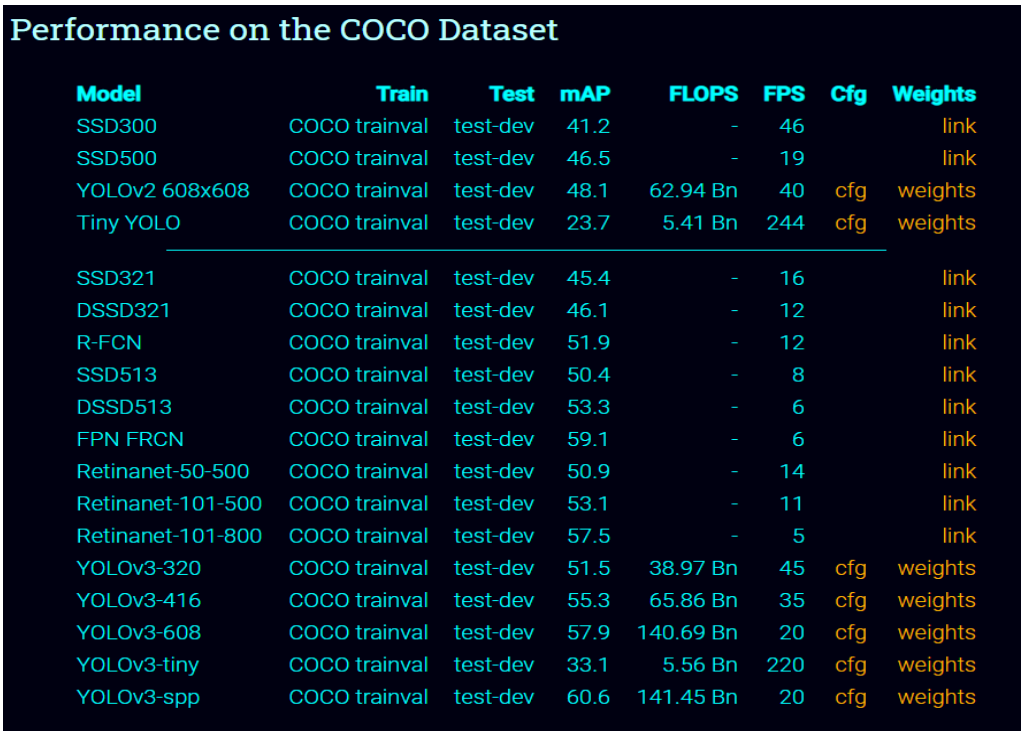
Our system will be able to ignore the background and its noises and identify the fire that is within an image or a sequence of images with minimal processing power via Raspberry Pi through these methods/models.

5.4.1.1. Generic Object Detectors

There are several accessible neural networks such as YOLO and Faster RCNN via GitHub. We were also able to discover other neural network that focuses on detecting fire instead of having functions such as object classifier. We need to identify which of these models and neural networks will provide out system the best performance possible within the limited time frame using Raspberry Pi. This subsection covers the comparison between them.

5.4.1.1.1. 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 (Redmon, YOLO: Real-Time Object Detection, 2018). 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 (Redmon, YOLO: Real-Time Object Detection, 2018). 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 51: Comparison of other state-of-art models on the COCO dataset (Redmon, YOLO: Real-Time Object Detection, 2018)

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.

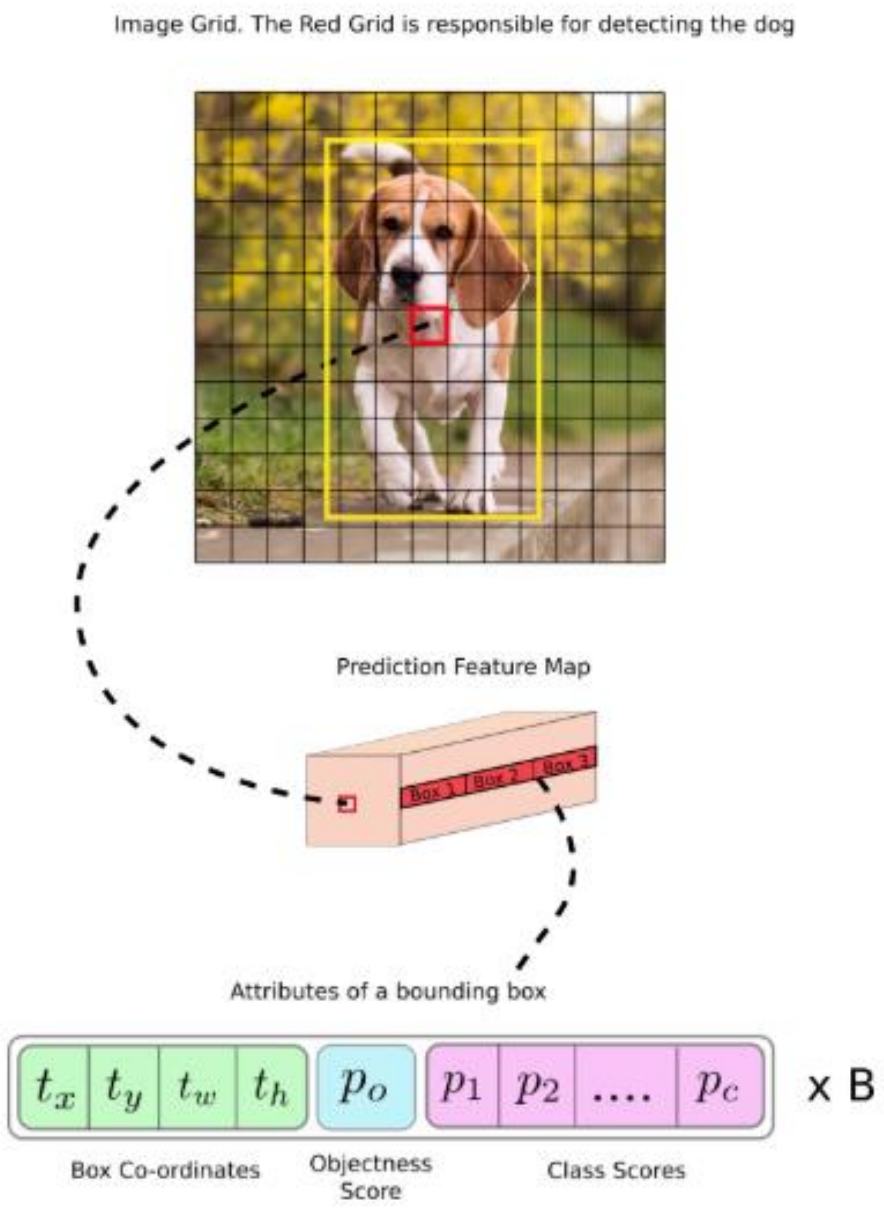


Figure 52: YOLO Bounding Boxes (Kathuria, 2018)

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 may not be needed for the purpose of our system. Our system focuses on binary classification of whether the flame exists in the image or not. If time allows, we would like to scale it to be able to differentiate the forest fires from other flames such as campfire.

But if the other methods without these features still perform less than YOLO, then we will continue to use YOLO for our system. These extra features may also lead to better scalability for the computer vision of our system by adding more information to our system.

There are multiple versions of YOLO available. The latest one would be YOLOv3. However, one suitable for usage with Raspberry Pi would be TinyYOLO which is YOLO adjusted for embedded systems such as Raspberry Pi. This should still be fast enough to identify and detect objects. We only desire this model to detect fire, so the full capability of YOLOv3 is not really what we seek for our project. Since we do not expect the model to identify and classify in detail, the lower accuracy of TinyYOLO should pose no issue as long as it can detect fire. Furthermore, since our project focuses on the detection of fires in the forest, we are expecting these models to easily distinguish the background from the fire and detect it as it will have drastic differences such as colors or shape.

Some of the concerns when using YOLO is that the available pre-trained model is trained with a dataset that contains big distinguishable objects within an image. This may cause the model to miss some of the minor fires. This is a drawback we are considering accepting, since the fire should eventually be large enough for the model to detect and not large enough for it to be too late. In addition, the pre-trained model may not be able to detect the fire since it is not trained with any fire object beforehand. This means we may need to find an alternate object that is similar to fire that the model detects or find a different model that is trained with a dataset containing fire. The accuracy of the pre-trained model can be obtained by running it with a dataset containing fire. Another option is to train the model ourselves, but it will be time consuming as we need to create our own dataset and train the model using CPU.

5.4.1.1.2. Faster RCNN

Faster RCNN is another state-of-art object detection method. It is easy to implement from scratch, and many resources are also available online. It can detect much smaller objects compared to YOLOv3. This is due to it having nine anchors in a single grid while YOLOv3 contains two anchors. This aspect of the Faster RCNN is more desirable to us than what YOLOv3 offers as fires can be subtle and small. However, the Faster RCNN is much slower and may not be ideal for real-time detection. We need to test the speed of faster RCNN with our hardware to properly conclude that this is not optimal or better than the other models we searched. Due to the already proven time consumption and processing time compared to others, this is the least expected model to be implemented to our system.

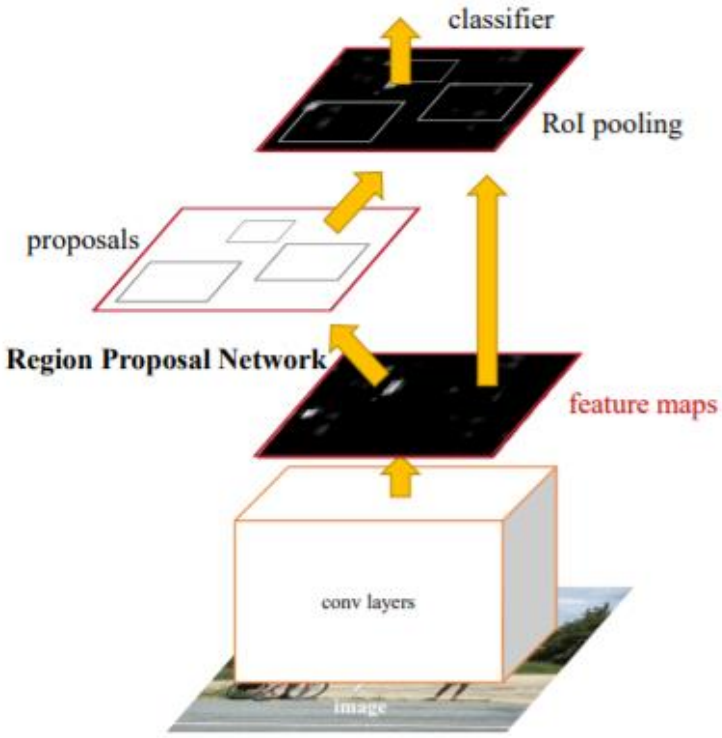


Figure 53: The structure of Faster RCNN (Ren, He, Girshick, & Sun, 2016)

Faster RCNN has a similar structure to YOLO where it has regional proposals or bounding boxes for objects. The main difference between them would be that Faster RCNN is not trained to do classification and bounding box regression at the same time. This makes the Faster RCNN much slower than YOLO. Unlike YOLO, it does not effectively eliminate the predicted bounding boxes which results in much larger computation than what YOLO can do.

Although Faster RCNN is used widely, there is not much documentation of it being used with the Raspberry Pi. This is most likely due to the fact that it is not suitable for real-time detection and its speed. Not to mention YOLO is also available and is a much optimal model for most cases.

We will attempt to implement Faster RCNN to the Raspberry Pi if possible, but if it deems to be too time consuming due to troubleshooting and adjustments, then we are highly considering to focus on other methods such as YOLO. We still included this model as one of our options since it is a state-of-art and known to detect smaller subtle objects better than YOLO.

5.4.1.1.3. MobileNetV2

The other models mentioned are well known and are expected to return good results after implementation. However, as mentioned earlier, the main concern lies in whether our embedded system can handle all the computations and processing fast enough to detect the fire. Thus, we started looking into other models that are commonly used in a similar set up as our system.

One of the models we 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 (Sandler, 2019). These can also be used for real-time object detection and can be easily implemented by Raspberry Pis. Many examples are available online.

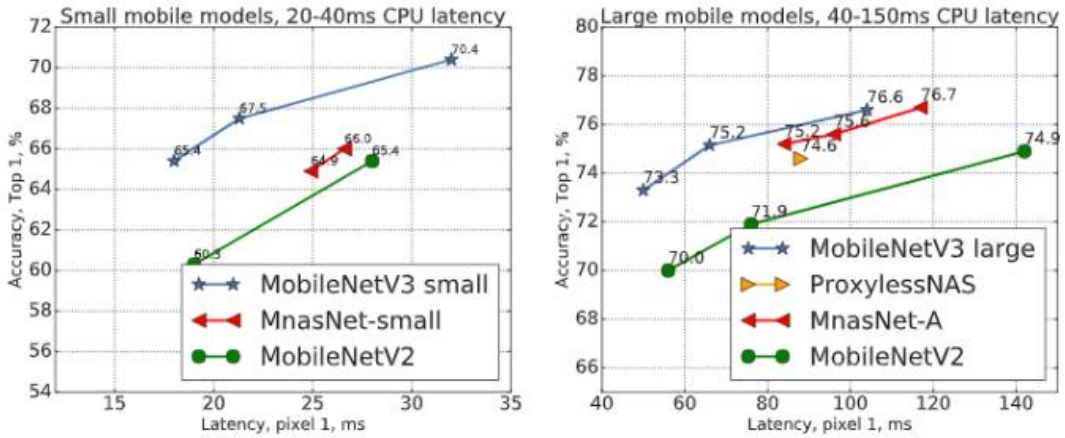


Figure 54: Comparison between MobileNetV2 and MobileNetV3 (Howard, et al., 2019)

As seen in the figure above, MobileNetV3 has better accuracy compared to MobileNetV2. However, since the MobileNetV3 is new, MobileNetV2 is preferable to use when it comes to training the models ourselves. There are details that could be found about the hyperparameters of MobileNetV2 in the GitHub and not of MobileNetV3. For this reason, we prefer to use MobileNetV2.

This model should be able to work well with our system, but we are not sure how better or worse it performs compared to TinyYOLO or Faster RCNN. This is something that we can find out only through testing and repetition. We hope to be able to implement these models and provide a comparison between them to determine most optimal model for our system.

5.4.1.2. Frame Differencing

One of the methods we can add to our models is frame differencing. It is an easy adjustment but may prove effective in identifying small objects between frames. 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. Other concerns we had was distinguishing the flames from other movements such as leaves swaying or movements of animals in the forest. These can be solved by frame differencing by eliminating background noises and noticing a big difference is an animal is detected. Then, 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. We may use this information to further ease the computations and determine the flame or area of it at an early stage.

Figure 55: Frame Differencing (True)

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 we are expecting to be able to identify smoke in our system to warn or notice fire in great distance beyond what the camera can capture. By being able to distinguish fire from smoke, it can add better scalability to our project as smoke may provide additional information along with the other information. We may be able to estimate distance of the fire or its intensity by understanding and learning how these smokes are detected in our system.

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 our desired subjects of fire and smoke in order to alert our 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.

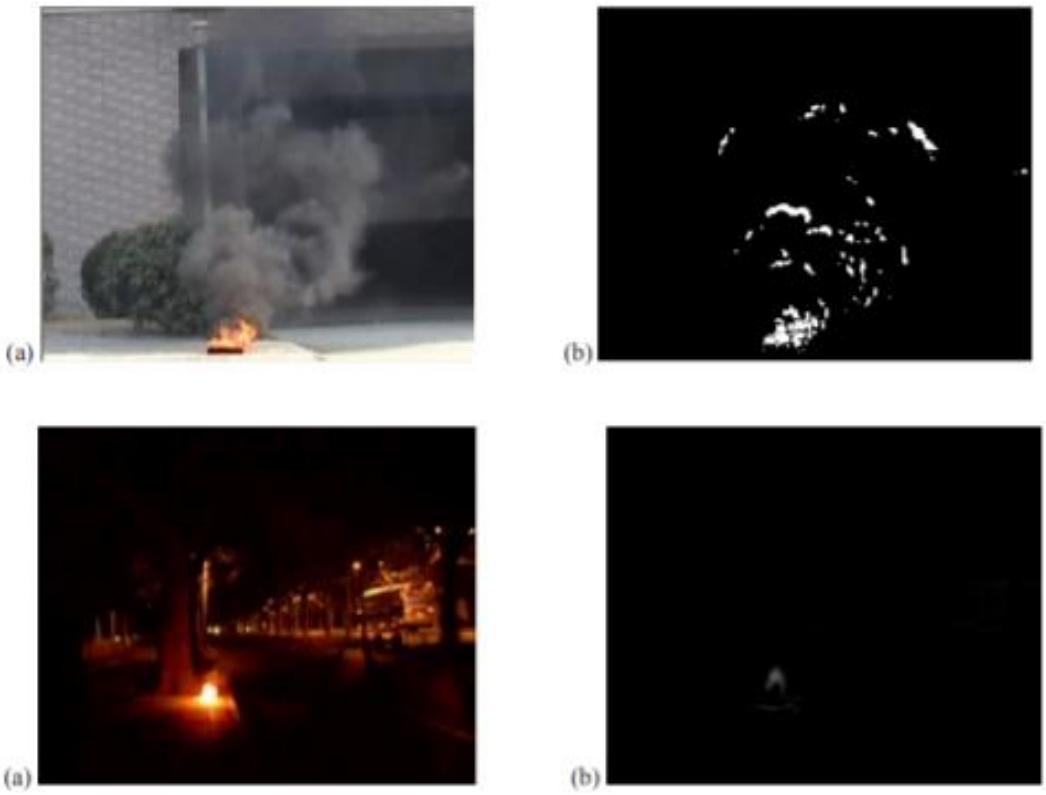


Figure 56: Frame differencing continued (Yu, Mei, & Zhang, 2013)

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 our system's effectiveness and utility. For instance, even if we miss to detect the small flame, the smoke can trigger the system earlier rather than waiting for the flame to be large enough to be recognizable. Adding this feature to our model should increase accuracy and effectiveness. Not only that, it also adds better scalability and utility for our system.

5.4.1.3. Color Classification

Another method that may help us greatly in distinguishing the fire from the background is 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 our system. Since our focus is forest fires, the fires should have significantly different color from the other objects in the background including possible other subjects like animals crossing by. Thus, we expect our 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 increase 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 may still have objects with similar color to the flame such as red flowers, woods, and leaves. To avoid confusion between different objects, we plan to focus on specific unique color that is most applicable to flames to increase accuracy. However, choosing a specific value of color to detect fire as a threshold may become tricky as it may increase false alarm rate or decrease predictions too much and lower the accuracy overall with slight changes in the value.

But the application of color classification to our system should be much easier compared to frame differencing. We could apply OpenCV's color classification or dissect our images into the RGB layers and focus on the R layer alone to help the model detect the fire. OpenCV is a reliable library that provides color classification and shape detector. It also has a package 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. However, the example is performed with small complexity and we still need to perform this function to our own dataset or examples to determine how much color classification of OpenCV is effective with our system and goal of detecting fire. If it is not performing as well as we desire, we can create our own color classifier by analyzing the L*a*b* color space or RGB and HSV layers. We are also aware that the L*a*b* color space is better than RGB or HSV space as it has actual perceptual meaning. But since we are focusing on forest fires. Looking at RGB and focusing on red intensity may also work, possible better, for our system. We can also consider the brightness and redness of an object as well.

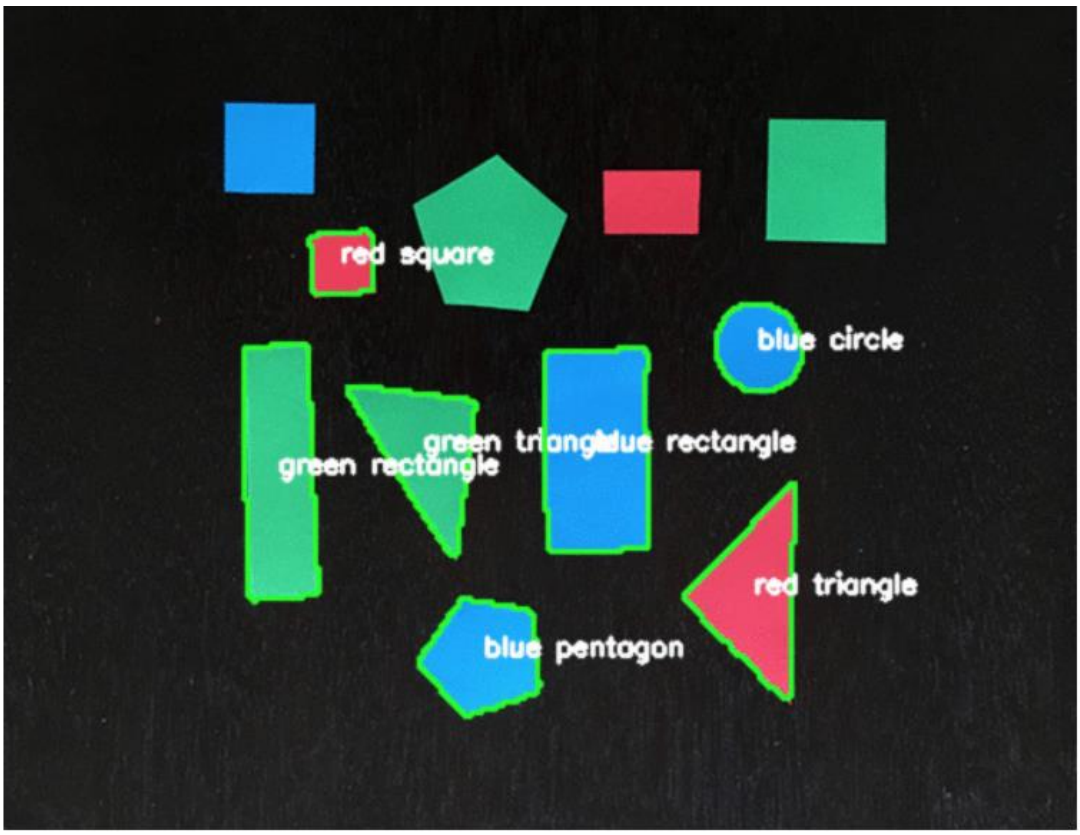


Figure 57: Color Classification using OpenCV (OpenCV, 2020)

There are other researches that also implemented color classification in order to detect fire. Figure 57 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 58: Color of Fire Classification (True)

(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 in terms of colors most of the times. Adjusting different setting such as saturation and filters may also help in identifying the fire and distinguish it from the rest of the background. We expect that combining color classification along with motion detection (I.e. frame differencing) will help our model increase its accuracy further.

We will start with a dataset or sample sets with no or least number of ambiguous objects that may interfere with the identification of a flame. This is to test the effectiveness of color classification and possibly along with motion detection either by frame differencing or optical flow. Then, we will further test it with more ambiguous objects to validate the accuracy and effectiveness of our system using color classification.

Color classification should be one of the optimal methods we can add to our system because it does not require heavy computations and libraries are available to easily code the method into our Raspberry Pi. We hope to see significant improvement by adding this

method on top of our machine learning models and possibly additional method to detect motion.

5.4.1.4. Optical Flow

Another method to detect motion is using optical flow. This method can be implemented using OpenCV [12]. 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 59: Dense Optical Flow (OpenCV, 2020)

Top image in the figure 59 is the original image while the bottom image shows the result of dense optical flow via OpenCV. Optical flow is another method we consider implementing as the other motion detector like frame differencing. Optical flow adds a bit more complication than the frame differencing, but the available OpenCV library help us implement this method with ease much like color classification. Which is one of the reasons why we chose to add optical flow into our system.

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

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 seems to be able to ignore the background noises which can help in not having leaves or trees as false positives. Thus, this method is very effective in detection motion while identifying its density and vector. Compared to color classification and frame differencing, this seems more promising and easier to apply while capturing enough motion to detect the fire.

We can also use it to pre-determine whether there is a fire before sending it to the CNN. We can provide threshold for the density of movement to determine if there is a possible flame. The additional information of density and vectors may also help us have better scalability in our system by providing additional information. This will help in having more data and ways to inform our users about the situation of the forest better. We may be able to detect other movements and information in addition to the sensor information we have.

5.4.1.5. Superpixel Localization

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

A research from Durham University (Dunnings & Breckon) 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.

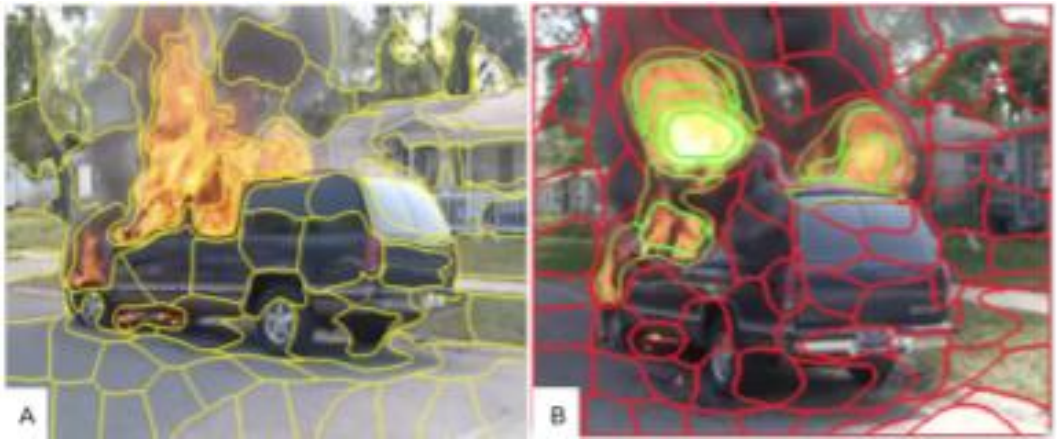


Figure 60: Superpixel Localization from Durham University (Dunnings & Breckon)

This method is also available in GitHub created by Toby Breckon (Deshmukh, Breckon, & Dunnings, 2019). 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.

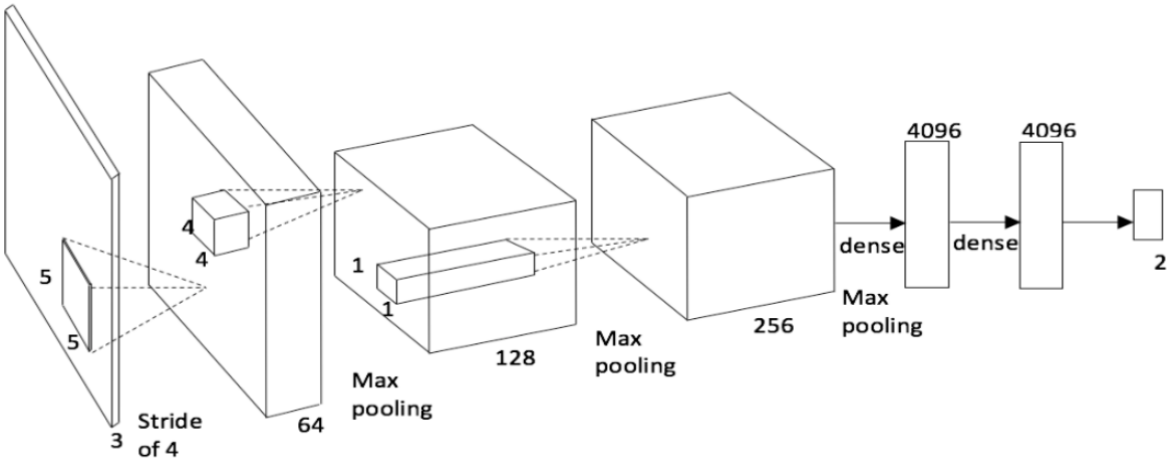


Figure 61: FireNet Architecture (Deshmukh, Breckon, & Dunnings, 2019)

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

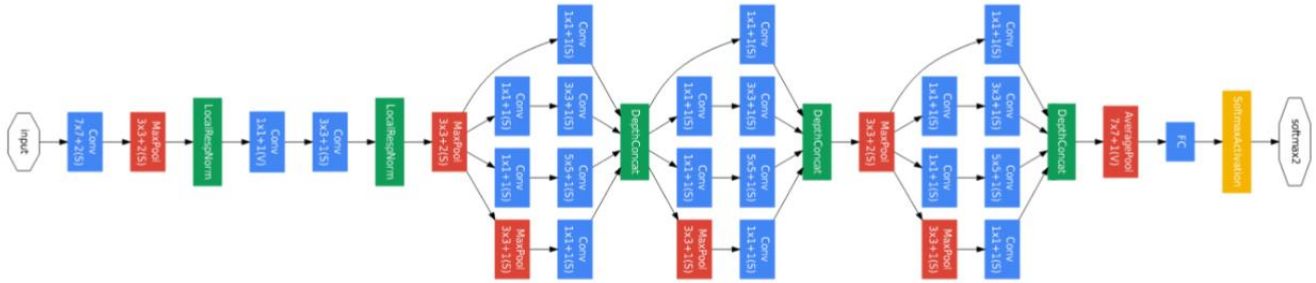


Figure 62: InceptionV1-OnFireNet Architecture (Deshmukh, Breckon, & Dunnings, 2019)

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.

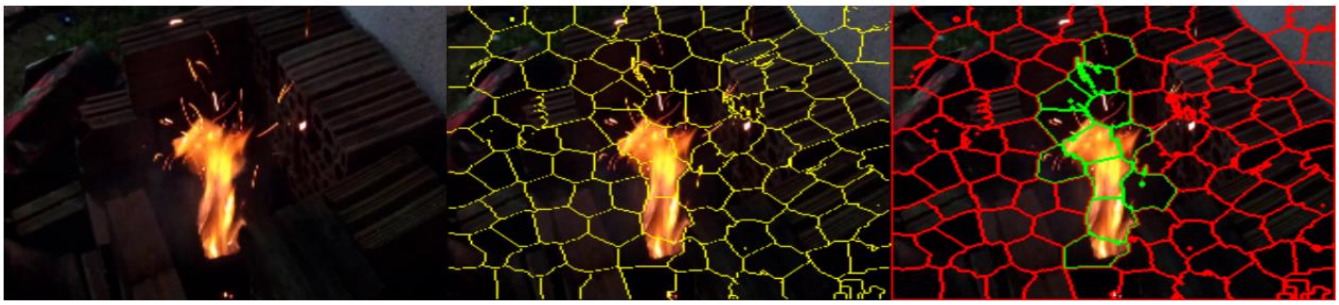


Figure 63: : Implementation of Superpixel Localization with CNN (Deshmukh, Breckon, & Dunnings, 2019)

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.
Thus, we believe that using superpixel localization may also help us improve our system. We expect that this will provide much better results rather than using OpenCV shape detector to identify the flame.
One of the other ways to utilize superpixel into our system is to use OpenCV just like color classification. OpenCV provides three different algorithms we can choose from to perform superpixel. They are SLIC, SLICO, and MSLIC as shown in the figure below. (OpenCV, 2020)

Figure 64: Superpixel Localization using OpenCV (OpenCV, 2020)

We thought that the superpixel localization was an interesting way to tackle our goal of fire detection in forests without temporal information. We hope that by adding this method, we will be able to lessen the computation needed from the networks while keeping the accuracy high.

5.4.1.6. Original CNN (Convolutional Neural Network) Design

There are other available models out there but not much that are publicly available for us to implement especially with our embedded system. To solve this problem, we can create our own CNN architecture and train it to have the most optimal outcome out of all the

CNN we mentioned. However, this means finding a novel way in such 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, we will most likely avoid creating our own model 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 we will test, we can create a much simpler CNN architecture concentrating on binary classification and just identifying whether the fire exists or not. We can improve this model by adding the other methods mentioned such as color classification or motion detector to eliminate false positives in the early stage.

In considering making new CNN architecture, we have to keep in mind the time constraint we have as well as processing limit of our embedded system. Further tests will be necessary to determine if this route will be in further consideration to utilize in our system.

5.4.2. Neural Network Frameworks

Different frameworks exist to help with the implementation for 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. We will also take in consideration of which framework will be most optimal for the capacity of our embedded system.

5.4.2.1. Keras

Keras is a high-level neural networks API, written in python. (Keras, n.d.) 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.

5.4.2.2. PyTorch

PyTorch is an open source machine learning framework that excels in researching prototyping and production. (PyTorch, n.d.) 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 that we have good understanding of machine learning.

5.4.2.3. TensorFlow/TensorFlow Lite

TensorFlow Lite is an open source deep learning framework for on-device inference. (Abadi, et al., 2015) It is commonly used with the Raspberry Pi Zero. To implement deep learning, we would need to install this to our MCU. Most of the tutorials we encountered use this framework especially for Raspberry Pi. It is also most optimal for integrating AI into a product. TensorFlow is also another well known framework similar to PyTorch that many industries utilize. It is also beneficial for us to properly understand and know TensorFlow just as much as PyTorch.

5.4.2.4. OpenCV

OpenCV is not a framework, but an open source computer vision library (OpenCV, 2020) that has many computer vision applications that supports the mentioned frameworks such as PyTorch or TensorFlow. 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. We would like to first implement these pre-trained models available in OpenCV to test with Raspberry Pi Zero and determine if we need to make improvements for the models.

5.4.3. Settings for Machine Learning

There are different options available in the setting up machine learning for our system. For instance, the programming language we would like to learn and use, or the embedded systems we have to choose from different types of Raspberry Pi. This subsection will cover what we aim to use and the reasoning behind them.

5.4.3.1. Programing Languages

Most of the available models are in the Python, if not in C++. We will most likely conduct our codes in Python, but we are also considering learning C++ when using OpenCV or

Pytorch. The reason we chose Python as primary is for the ease of use and because it is widely used in computer vision applications. Thus, making tutorials and resources much easier to acquire in Python compared to C++. The reason why we chose to consider C++ and would like to code using it is due to its demand in industries. Many industries and positions require or prefer applicants with decent proficiencies in programming C++ as it has become one of the standard languages to be used within software engineers. However, due to the time restriction and ease of use, we will primarily use Python for machine learning.

5.4.3.2. 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 other systems. Thus, we decided that the Raspberry Pi Zero should sufficiently perform and meet our goals while saving us significant cost.

5.4.4. 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. But due to time constraint and efficiency, we are aiming to have about 400 images with the fire and 400 without a fire. We plan to combine the images we took ourselves and images online to create this dataset. We may increase the number of images if the model does not perform any better with other possible adjustments such as its hyperparameters.
We can also find one of the datasets we found when looking for researches for fire detection using computer vision which can be found on Durham Collections. It has a focus on flames which would be ideal for our use.

However, we will prioritize implementing and improving the pre-trained models if it has enough accuracy and results. With different methods and models, we may need to adjust the dataset as well to include different types of fire. We may even need to add smoke as part of our dataset to see if it can successfully detect and distinguish the fire and smoke to alert regarding the forest fire. We may also add different subjects such as animals or passerby to test how the system would distinguish them from fire and if it can detect the fire with increased distractions. It is also important to keep in mind that the system may be confused with objects that has similar colors as the fire such as leaves, woods, or structures. It will be better if we could also include these types of sets and test it to our system to see how it would effectively detect forest fire.

If we are creating a dataset, it would have different categories such as by noises (i.e. similar colors, more obstructions), distances, or amount of flames (possibly including fire). These can be expanded as we see how much our system can grow or may need to grow. Our base would still be a dataset with fire and simple background with minimum noise to test our system properly.

5.4.5. Summary of Machine Learning

Through looking at multiple existing researches regarding utilizing computer vision in fire detection, we can safely assume that we can create and implement computer vision in our system to detect fire in the forests. There will be many issues we have to tackle such as processing limitation and memory capacity issue from our chosen embedded system, and necessary adjustments to make our computer vision appropriate for forest application. We will most likely need to improve existing models by re-training them or adding other methods such as color classification, frame differencing, optical flow, or superpixel localization to it.

To ensure we succeed in implementing computer vision into our system, we would like to have a basic computer vision built using pre-trained models and open source libraries such as OpenCV to know how they are effective in detecting fires. This is also to test how fast Raspberry Pi Zero works with our setup. Then, we may use state machine for color classification, frame differencing, or optical flow to have a threshold and detect fires. Optical flow is our top method as it provides density of movement and direction to help us distinguish the flames from other movements. If this is successful, we can add those to our model. For instance, if one of the methods returns high threshold, then it should send the image to the model. If the model also concludes that there is a high threshold of a flame being detected, then fire should be detected and alert the other systems. This should improve the accuracy of detecting fire compared to using the available object detectors alone.

Our top choice for the pre-trained models would be the one with super pixel localization as it is specifically trained with a dataset of fires in order to detect the fires. The main issue we need to tackle when implementing this model is if we can make it work with the Raspberry Pi Zero.

If time allows, we can create our own dataset and train the models using the images through the methods (color classification, frame differencing, or optical flow). This will allow our model to learn from a different perspective and be able to ignore the unnecessary information (I.e. background). We expect that our model should then return better accuracy. Further improvement can be done by adjusting the architecture or the hyperparameters of the CNN. This is very experimental and needs time, but if successful, we will have a novel way to detect fires in the forest.by adding superpixel localization, we may create our own network without much complexity as the super pixel localization works very well with such architecture.

6. Testing and Prototyping

Testing our hardware and software is almost as important as the project itself. To ensure that everything is going to plan, we will make sure to test each component and subsystem separately first and move on to more integrated testing as time goes on. This will, ideally, cause our full prototype to be as functional as possible since all the bugs are worked out on a smaller scale before integration. The first section below deals with the advancement of our knowledge and experience with the subsystems to prepare for a 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 needs to be designed is the power regulation. We have 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. The designs for the switching buck regulators will be built onto a breadboard to prove they work, and a variety of expected voltages will be applied to the input to ensure that it can give us the required output voltage. From this point some calculations will be 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 is investigated, it is 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 will be put together on a breadboard and we will attempt to charge and discharge the batteries. Safety must be observed as the batteries are an energy storage device. This proof of concept will prove that we can charge batteries and use them. A few charge and discharge cycles will be 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 will be used to generate some power from either the sun or a bright light. Use of the solar panel will be investigated so we can become familiar with the characteristics of the device and how it will affect our circuit. In the end, all three parts of this subsystem will be combined with a simulated load such that we can see if the power system can support everything. After testing this system a layout can be put together to put this system on a PCB and the design for the mechanical support structure for the solar panels can be finished.

6.1.2. Sensor Subsystem

Testing the Sensor Subsystem will take place in two parts: retrieving sensor data reliably and retrieving accurate data. To facilitate these tests, a Raspberry Pi will be used so that the team can become familiar with the Raspberry Pi's interface and operating system. The goal of the first portion of testing is to investigate the ease of use of the sensors. A sensor will be hooked up, on a breadboard, to power and all the supporting hardware will be given to the device. Then, a Raspberry Pi will attempt to retrieve data from the device through SPI or I2C. At first the result's values will not matter; just that results exist. All of the sensors will be tested this way until each sensor has been used enough to determine its difficulty in using it. It is at this point that a sensor may be deemed too complex or difficult to use and we may decide to use different sensors. This step is important to determine what sensors will be used in the final product.

Finally, the sensors will be checked for accuracy. This may be determined with sensors that we know are accurate (such as a carbon dioxide detector for gas) and our sensors and see how close the values agree. For something like a smoke detector, we check to see if the smoke detector can actually detect smoke (or something equivalent). We must reach some level of accuracy with our sensors or else the system will not detect fires with accuracy either. Lastly, if we have a camera, we will take some pictures and save them. If the sensors' data is not accurate enough or the pictures are not clear enough, then we may have to investigate other options for different sensors and/or cameras. After testing this system, a layout can be put together to put this system on a PCB and the design for mounting the sensors and where everything must go can be completed.

6.1.3. Network Subsystem

Testing the Network Subsystem will consist of a few parts to ensure that the Network software and hardware works all together. The first step will be 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 will allow us to get a feel for the software and hardware requirements since we plan 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. From there a program will be written to allow a button push to turn on the built-in user LED on the other device. Ideally, the LoRa protocol is 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 will og 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 will send this string to the other device and the other device will print this data to another terminal. After this test, the software is 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.

The next part of testing for the Network Subsystem is the internal timers and GPIO pins. On a breadboard, the Network Subsystem will connect to a button, two LEDs, and a serial terminal. Using the button, the network subsystem will turn on an LED and send data to the terminal. This is to simulate the user's input on the final system. The other LED will blink based on a timer interrupt within the network controller. This will be used to simulate the interrupt for turning on the Processing Subsystem. In the final system, a transistor or solid-state relay could be used to allow power to get to the Raspberry Pi. The last part of this testing will be to have data from the terminal saved to memory in the network controller. When a specific sequence of characters arrives (as in a command) the network controller will also turn on the LED and write a response to the terminal to simulate a packet arriving and the network controller taking action based on a specific field in the packet. This testing may or may not use LoRa to complete most of the tasks. After testing this system, a layout can be put together to put this system on a PCB that is similar to the evaluation kit so as to keep the RF characteristics consistent from testing.

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 must be set up with all the software necessary to preform machine learning algorithms and run Python code. From there, the Raspberry Pi runs through some sample images to test out the algorithms. The subsystem, moreover, needs to have the training data loaded. This may be previously collected data, our own training sets, or purely data retrieved from our sensors. The final result will likely be training data that is a mix between all three. From this point on, the machine learning parts of the subsystem will need to be fine-tuned to be able to detect a fire with a decent level of accuracy. This "fine tuning" process will probably continue throughout the project until it is complete.

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 will be introduced into the system one at a time and we will ensure that the connections between the raspberry pi and the sensors works appropriately. To verify this, we may use a logic analyzer to see the signals sent to and from the Raspberry Pi and the sensors. This testing is meant to focus on the software interaction between the Raspberry Pi and the sensors, not that the sensors work.

This will have been tested in the sensor subsystem testing. Finally, the Raspberry Pi will need to be tested with the Network Subsystem. Ideally, this is not a complex system. The Raspberry Pi will attempt to send information to the Network Subsystem and vice versa. 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 we can send messages reliably between the two devices. After testing this system, a layout can be put together to put this system on a PCB with the mounting style for the Raspberry Pi as well as the power control circuitry for the system. Some testing may go into ways to limit power

to the Raspberry Pi since it cannot turn on from being shut down without the power fully turning off and then turning back on again.

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 might be 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 will include testing the physical capabilities of the sensor. Some sensors have digital output and can output different digital protocols like UART, I2C, or SPI. Others have analog and differential outputs which require and amplifier and some other supporting circuitry. This means that it must function correctly,

or the software cannot determine what values are correct and what isn't. Therefore, a testing procedure must be put in place.
Step-by-step:

a. Power any sensors and see if they appear to function.
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 sensors will be exposed to nearby smoke and fire to test detection of organic gases.
    b. NIR sensors will be exposed to nearby flames to detect infrared waves.
    c. Thermal camera will be exposed to nearby flames to record fire and non-fire data as explained in section 6.3.2 Computer Vision.
    d. Smoke sensors will be exposed to nearby smoke to asses if the sensor alarm is triggered.
c. Hardware testing will go 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. If the sensor is analog, check to see if the output falls within an expected and acceptable range and ensure all the amplifier circuits are working correctly.
e. If the sensor is digital, write software to only read the result and try to get meaningful data. Check to see if the output falls within an expected and acceptable range.
f. Try to convert the sensor reading to meaningful "real world" values and ensure they are acceptable for real world scenarios (especially the current scenario the sensor is in).
g. 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 will 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.

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.

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.

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 will be written to make the hardware do what we want or similar. This will allow us to see if the implementation and ideas we have are feasible or if they are just not quite what we need to do. These small, simple programs will allow us to see what is going on and later on expand their complexity into a full-scale prototype.

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 can be used to add more data and possibly, if done correctly, may be able to provide insight about the type of fire or what is burning. Sensor operation is depicted in the diagram of figure 64.

Step-by-step:

a. Write simple software to interact with the sensors and get raw data.
b. Obtain the data from hardware sensor testing data and see if that data can be made useful by mapping it to “real world” value.
c. Train the machine to establish fire and non-fire conditions by analyzing smoke, gas, and flame characteristics.
d. Complete software that will read the sensors and determine the possibility of a fire condition.

6.3.3. Computer Vision

One way to detect and identify fire is using computer vision and machine learning. There is much research done in creating different neural network architectures and methods to best identify flames. These can be done with minimal to no temporal information which helps our system to avoid heavy computations.
Without the processing subsystem: we can test the computer vision software using our webcam. We can run a real-time object detection using our computers and test to see how well it would detect different types of fires in forest especially the minor and small ones. Since there will be adjustments needed to accommodate to our processing subsystem, we will only test the pre-trained models and simple methods such as frame

differencing, color classification, or optical flow in real-time. By doing so, we can ensure that our code is properly constructed and should return proper results when transferred. By having simple tests for the models and methods, we can find the accuracy appropriate for detecting fire in the forest.

To do so, we would need to perform simple test runs multiple times and document the results for each models and methods. These test runs will follow just like the general diagrams below. The model will receive an image and process it to output its result. Then, the accuracy will be calculated by comparing its output to the ground truth.

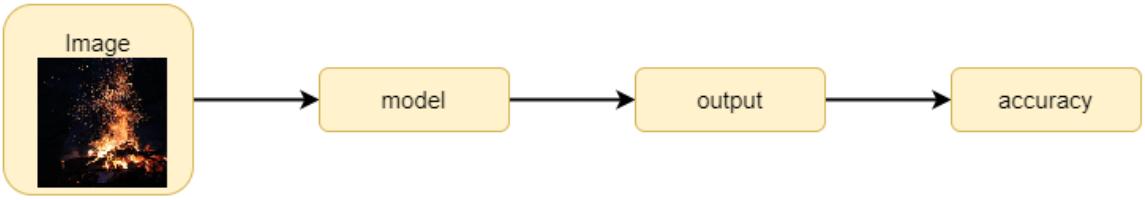


Figure 65: General diagram for testing models such as YOLO and Faster RCNN

Although the models may differ significantly from each other, the testing plan should be very similar to each other. For each model, there will be testing procedure we will follow which is illustrated in the diagram below.

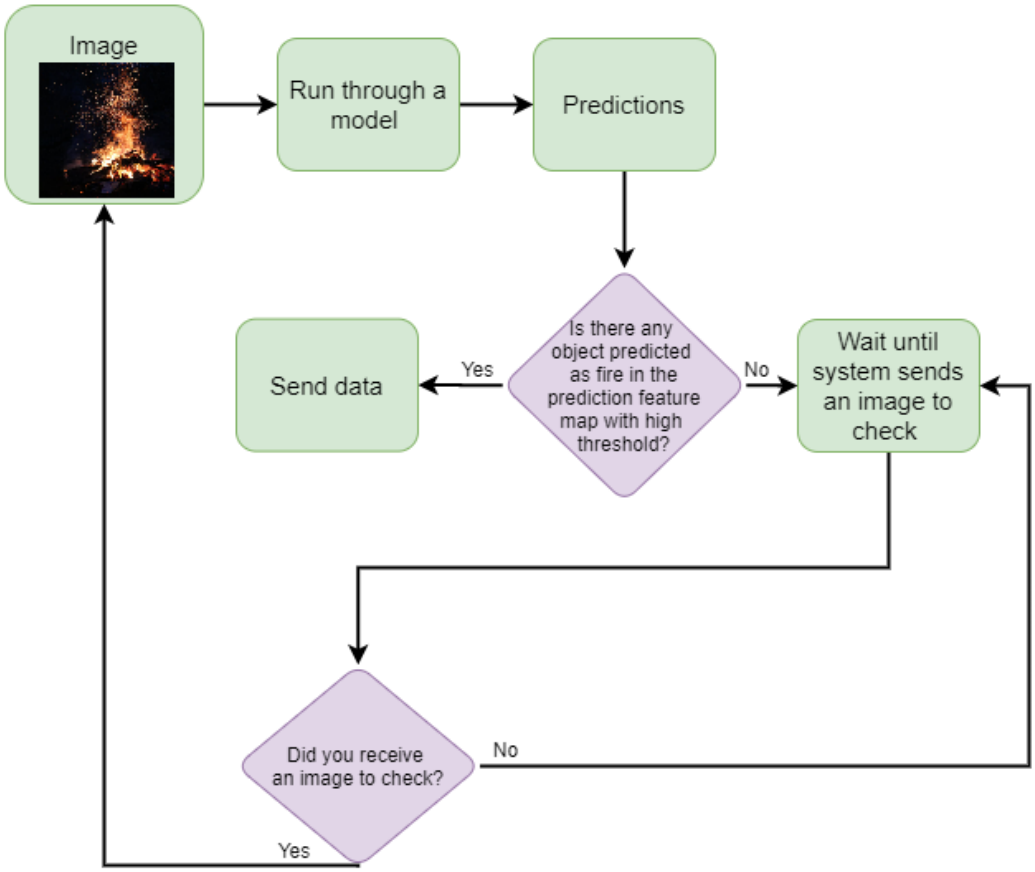


Figure 66: General software flow of the object detector through a model

After running a model, it should return an output (i.e. predictions feature map) containing information such as the location of bounding boxes of the objects, class score, and confidence score. Using the information, we will determine if there is a fire. Thresholding the confidence score and including an array of acceptable classes will most likely be needed to detect the fire properly. This is also another experimental situation where we will need multiple runs to determine the values and appropriate thresholding. We would also like to avoid constantly running the model unnecessarily, so we need to incorporate an ability for it to wait and turn on again when needed. It will also need to turn off after sending the results to other systems. We would also need to ensure that the model will work properly once it is turned on again. To test this using without the processing subsystem, we need to add a function to turn off after outputting prediction.
In addition to the model, we will be adding another method such as color classification or optical flow to help our model determine if there is a fire. By having this extra step, we are hoping to achieve better accuracy while helping the system lighten its average processing time.

Testing these methods will have similar construction to testing the models. We will run a method on a given image. Then, it should output its results. And from there we calculate the accuracy of the method by comparing it to the ground truth. We will test these methods individually, just like the models, and then compare the results to each other to see how effective each method is. Ultimately, we would like to also have a comparison of combinations of different methods to see how we can successfully construct these methods to produce best results.

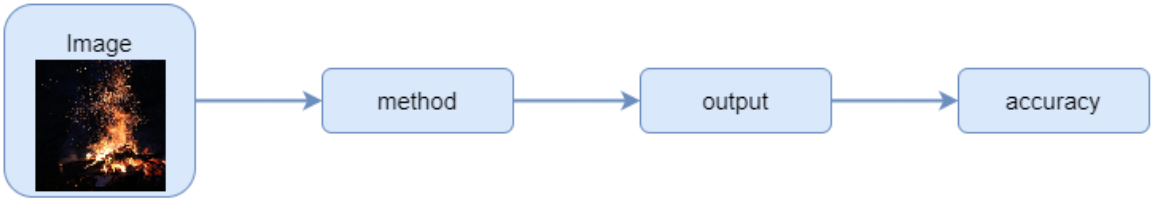


Figure 67: General diagram for testing methods such as color classification and optical flow

We are planning to test three different methods which are frame differencing, color classification, optical flow, and super pixel localization. These tests should mostly follow simple steps as illustrated in the figure above.

Since each method is unique, we would need to perform these tests slightly different from each other. Specifically, applying superpixel localization to other models may get tricky as we need to accommodate the models to read by meaningful segments instead of bounding boxes or per pixels. Thus, we may not test the superpixel by itself if time seems limited. However, there is already a model publicly available through GitHub that performs superpixel localization with a neural network which we can test similar to how we test other neural networks such as YOLO.
One of the methods, frame differencing, can be done by subtracting the frames to obtain a new image. To test how this will help our system achieve better accuracy, we will need to perform frame differencing along with filters and additional methods to identify an object

through the difference image. Filters such as blurring and normalizing can help isolate the object from the background noises.

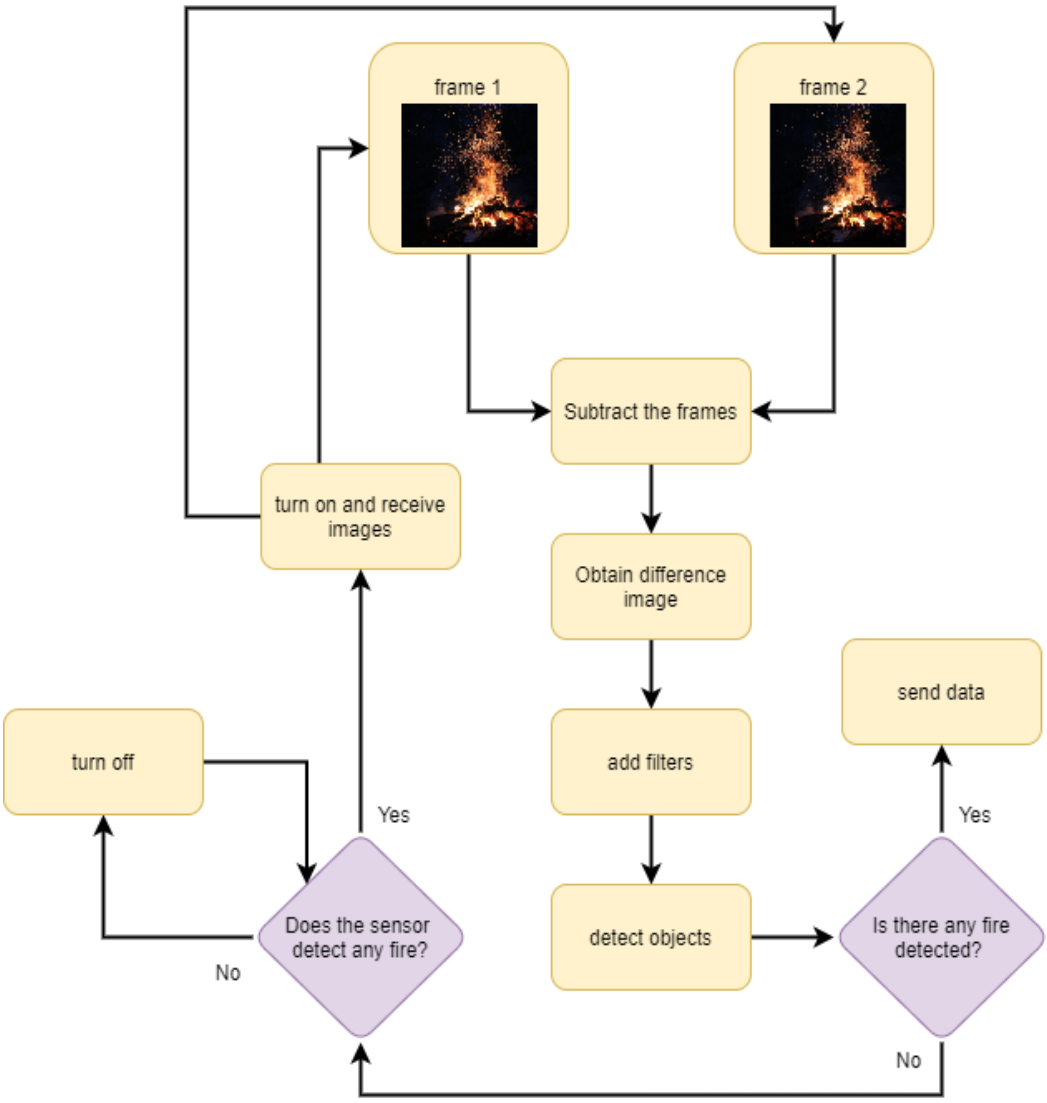


Figure 68: Diagram for frame differencing

To test the frame differencing properly, we would need to output the subtracted image to ensure it is correctly done. We would also need to compare the filters and see which ones will give the highest accuracy. By using shapes or other detectors, we may be able to detect fires through these subtracted images without the help of a neural network.
Another method we would like to test is color classification. Since our system is specialized in detecting fires in the forest, the fire itself should have a distinguishable color difference compared to the background (woods, grass, etc.). We can utilize this difference by applying color classification in our system to identify where there is an intensity in the colors similar to flames.

To apply color classification into our system, we would need to focus on identifying red colors in the image. Then, isolate them from the background. We can use shape or other object detector like grouping in order to identify object in the image. Then, we can focus on the area with high intensity of red instead of looking over the whole image. Color classification itself can be done using OpenCV as well as the detectors such as shapes.

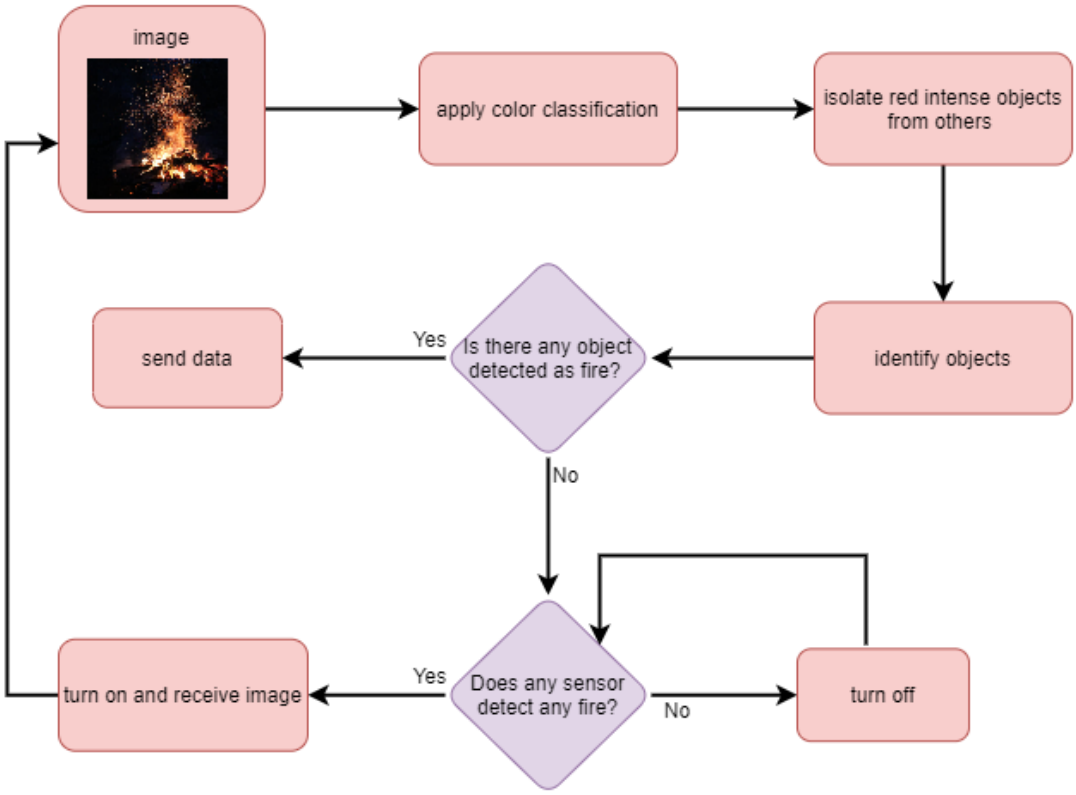


Figure 69: Diagram for color classification

There are different ways we can apply color classification by looking at different spaces such as RGB and HSV. We believe that focusing on RGB should be enough to be able to distinguish flames using color classification, but we have yet to test it using our system. We should have a better idea of this once we gather more data and images that the system will more likely encounter.

Last method we would like to incorporate is optical flow. Similar to color classification, optical flow can be implemented using OpenCV. We can use it to identify an area with the densest movement and isolate that from the background. And then have a detector to help identify if the detected object is a fire. The basic flow of the optical flow is illustrated in the diagram below.

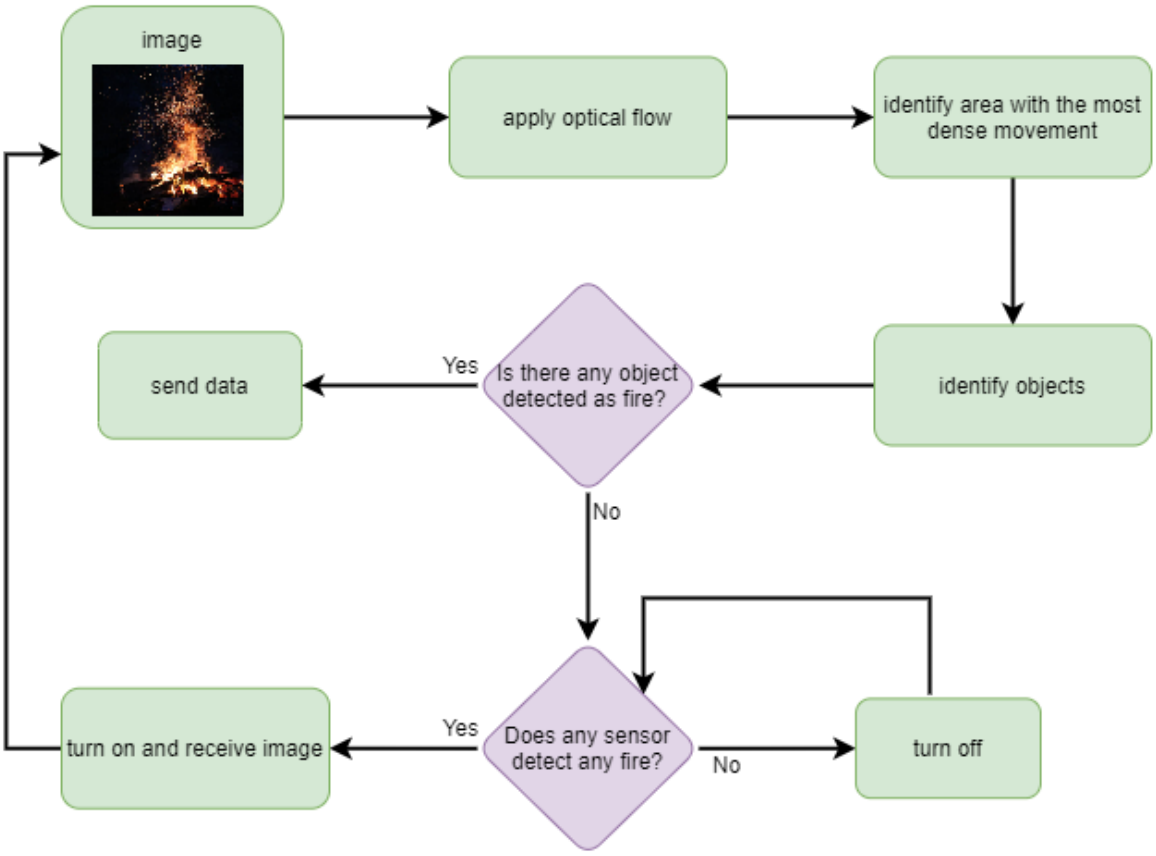


Figure 70: Diagram for optical flow

We should test the optical flow with few example sets in order to ensure optical flow is being applied properly. We also need to ensure that optical flow is appropriate for detecting fires using OpenCV. We expect it to work well with easy implementation.

The three methods mentioned previously (frame differencing, color classification, and optical flow) can help isolate and help the model identify the flame. After ensuring each method works properly, we can implement it to work with the selected neural network to produce better results. We can apply thresholding to identify if an isolation of an object happened after applying these methods. Then we can crop the image to focus on the suspected fire object and send it to the model. By doing so, we would reduce the amount of computation needed while decreasing other distraction for the model to make false positives. Object detectors also tend to work better with larger or focused object in the image. We expect that by combining methods and models should return better accuracy than running a pre-trained model alone.

We would also like to add superpixel localization to our system if possible. This is a way to segment the image in a meaningful way to make it easier to identify fire with reduced complexity of neural network architecture. This may help in reducing the heavy computation compared to other models we discussed. It can also make it easier to isolate the detected fire before sending it to the neural network.There will be three stages of testing this method if we decide to utilize it to different models or our own neural network.

This method will have slightly different formatting of test plan as it is a way to segment whole image and not a color or motion detector.

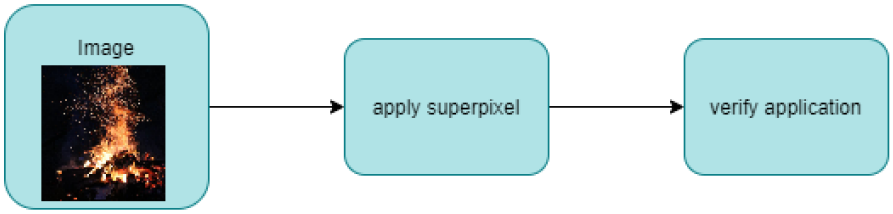


Figure 71: Diagram for Testing Superpixel Localization 1

The first stage in testing superpixel localization is to make sure the function works properly and segments the image correctly. We would need to verify this by running it through different example sets and images. Then, we would manually check if the segmentation was successful or not. We need to ensure that the segmentation is effective around the flame. These steps are illustrated in the diagram above.

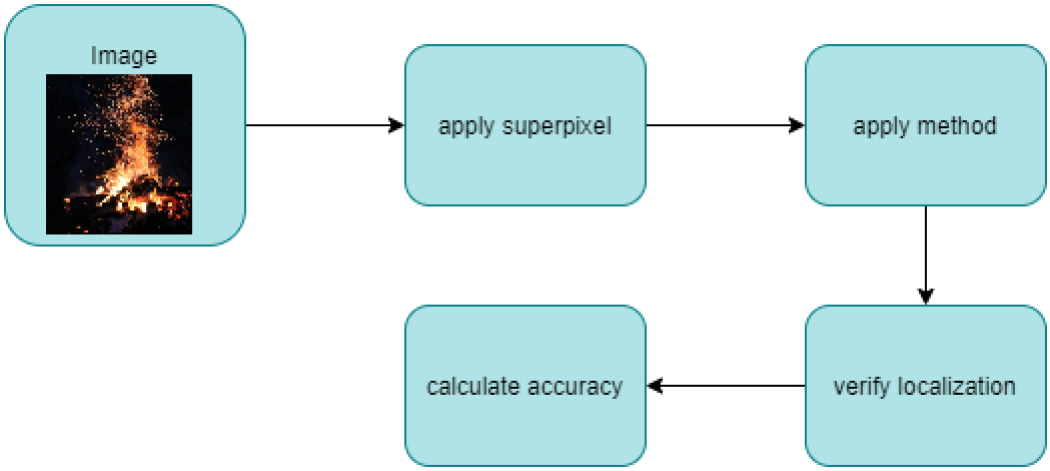


Figure 72: Diagram for Testing Superpixel Localization 2

The second stage of testing superpixel localization is to use the localization and rate how successful it segmented the objects especially flames. This is important for us to verify in order to help the model identify the flame effectively. If the localization was not successful, then there is not much benefit in applying and performing extra computations for our system. Based on the examples provided in OpenCV, it seems very reliable and we expect high accuracy of it segmenting the objects.

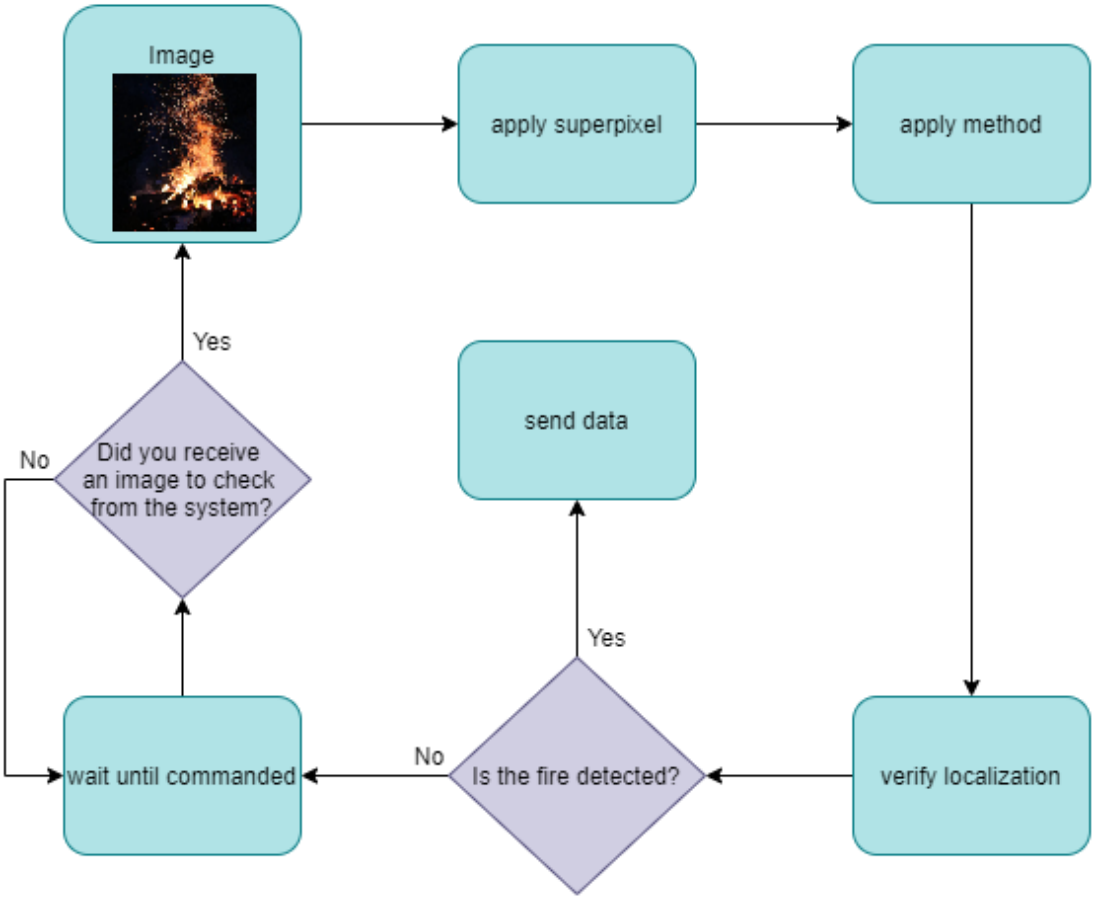


Figure 73: Diagram for Testing Superpixel Localization 3

The final stage would be incorporating it with a neural network to verify if the fire is detected within that segmentation as seen in the diagram above. After verifying the superpixel localization, we could further apply color classification, frame differencing, or optical flow to further narrow down the image before sending it to the network. By doing so, it will greatly help the model learn and identify the flame. Segmentation may help in properly encompassing the flame instead of approximately applying bounding boxes or using shape detector.

By having these methods, we can effectively eliminate unnecessarily processing by the neural network while achieving high accuracy. We would also need to manually verify the results of each one and calculate the accuracy in order to compare and choose which method is the best fitted for our system.

After comparing the results through these tests, we would like to combine different models and methods together into one system as shown in the diagram below.

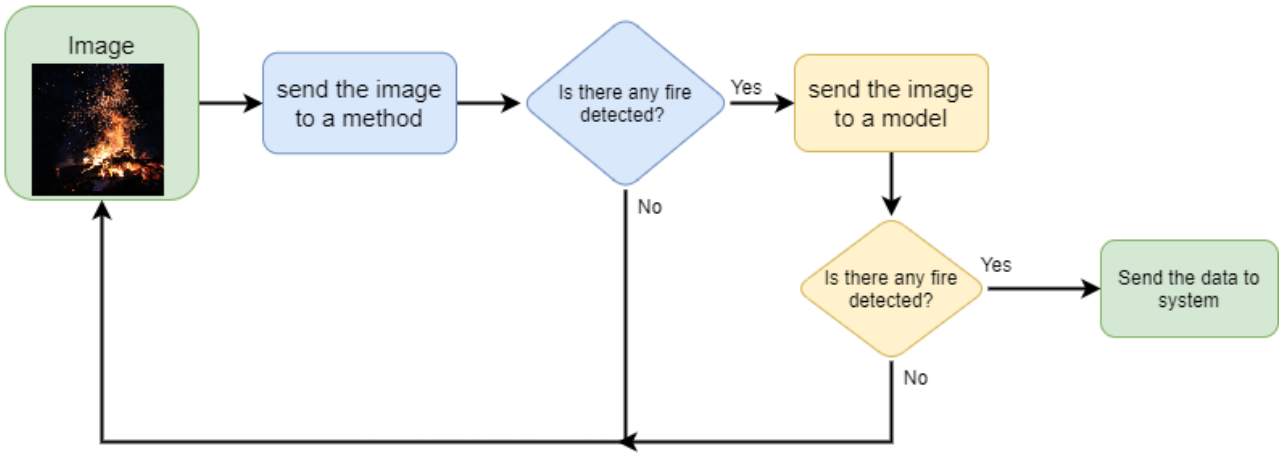


Figure 74: Diagram illustrating combining the method and model into one system of machine learning

By doing so as illustrated in the diagram, we should be able to achieve better accuracy. We can also combine multiple methods or models together and cascade them into the system.

With the processing subsystem: we can run the previously tested models or methods and see how the results may differ. By doing so, we can compare how the accuracy may change due to specification differences such as camera. It is also crucial to check the difference in time as Raspberry Pi Zero has lower processing power. If the software returns passing performance with our system, then we may further improve our system by testing the other models and methods.

To set up the processing subsystem, we will need camera along with the board to capture an image. We can set up LED light or communication monitor between the board and computer to monitor the output of the board and see if it is properly receiving inputs. We also need to ensure that the camera is working properly by outputting the captured image to a monitor. Then, we can run the model or methods to see if it is functioning properly. We can set it up so that the board will respond if there is a fire detected and notify the system. For instance, having the LED light up when the fire is detected and off when the fire is not present. We can also use the communication monitor to do similar process. We can set it up so that the board will respond if there is a fire detected and notify the system. For instance, having the LED light u

One of the hardest hurdles in applying this to our processing subsystem is accommodating the code to fit with the function of our embedded system as some may be limited and not applicable. We need to have functioning machine learning and computer vision applied to our embedded system while keeping the best accuracy as possible with best speed possible. If the current system cannot handle the computations or consume too much time to produce results, then we are also considering upgrading the system to Raspberry Pi 3 as it can handle more memory and more processing than Raspberry Pi Zero.
Step-by-step:

a. Code in computer to test the available pretrained models from YOLOv3.

b. Code in computer to test the optical flow using OpenCV
c. Record the processing time and accuracy. How accurate did YOLOv3 detect and optical flow to detect (have most density in flame).
d. Transfer the code to the Raspberry Pi and test after installing necessary libraries and adjusting the code to accommodate our MCU
e. Record the processing time and accuracy
f. Compare the results from computer and Raspberry Pi Zero. If they are not similar or significantly different, then we may have an issue with the code.
g. Repeat with different models and methods to compare results from each other Or choose to have a method to pre-determine if there is a flame and then pass it on to a model to determine if there is a fire. This combines and considers the thresholds from both methods and models to determine the flame instead of reading two different outputs. We expect the accuracy to increase by combining them in such way.

6.3.4. Networking

The network hardware can be tested in a step-by-step procedure. First we start by sending simple strings between the devices. Once we can send strings, we can worry about meaningful strings and possibly streaming data.

Step-by-step:

a. Code in a computer to test the methodology
b. Code in a computer to test the flow
c. Program the microcontroller with a simple script to broadcast a string
d. Program a different microcontroller to continuously receive all broadcasts
e. Program the two to recognize each other's unique ID and attempt to create a network where they can communicate with each other.
f. Implement a blacklist so that close-range demonstrations can be made.
g. Refine the software so that three or more units can be used to demonstrate their feasibility.

6.4. Testing Environment

In order to test the system, it will require to be in a place that has three conditions. The first one is direct line of sight testing which means it has to have large empty spaces half a mile to a mile long to 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, each of us will purchase the components we believe would be most suited for the project based on our individual background research. This will include each peer purchasing a raspberry pi, SD card, DC power source as well as the components we are responsible for in the project. This process will also include downloading any relevant software applications and becoming familiar with the chosen programming language, python.

During at home testing, the peer will attempt to test their components in a controlled environment. The sensors will be 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 thermal camera will be tested in front of fire and non-fire conditions to generate data that will be used for machine learning. This dataset will be used to train the system and produce a model that can attempt to identify a fire and look at the conditions for future predictions. The LoRa module will be tested by attempting to establish communication with at least two devices to see if data can be sent and received. The solar panel system can be 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 (LEE, 2020). 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 us to experiment with various mechanical designs and understand which design is best suited for this project. In addition, we would also like 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, we would also like to test the range of the devices using the LoRa module. We would test the devices at 10m, 50m, 100m, and 150m apart to observe if the communication and data transmission is still maintained. Moreover, we would also want 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, we are also 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 be the ideal testing environment since it is used by students from other colleges for educational purposes.

Figure 75: Controlled fire at the UCF Arboretum (LEE, 2020)

7. System Integration

7.1. System Design

An overall glance at the system shows the solar array hooked up to a voltage regulator which will output 12 volts and a max of 2 amps with the input from the array. That will then be feed into a Li-ion IC to handle the charging and battery health. That battery will then, with help from excess power from the solar panel, be inputted into two buck-boost converters to make a 5 volt and 3.3-volt rail. These two rails will handle power to the entire system as some components require specific voltages. The 5-volt rail will run the LoRa module and the Raspberry Pi and this rail will pull the most power out of the system due to how much power a Raspberry Pi requires. The 3.3-volt rail will handle powering all the sensors and will pull less power.

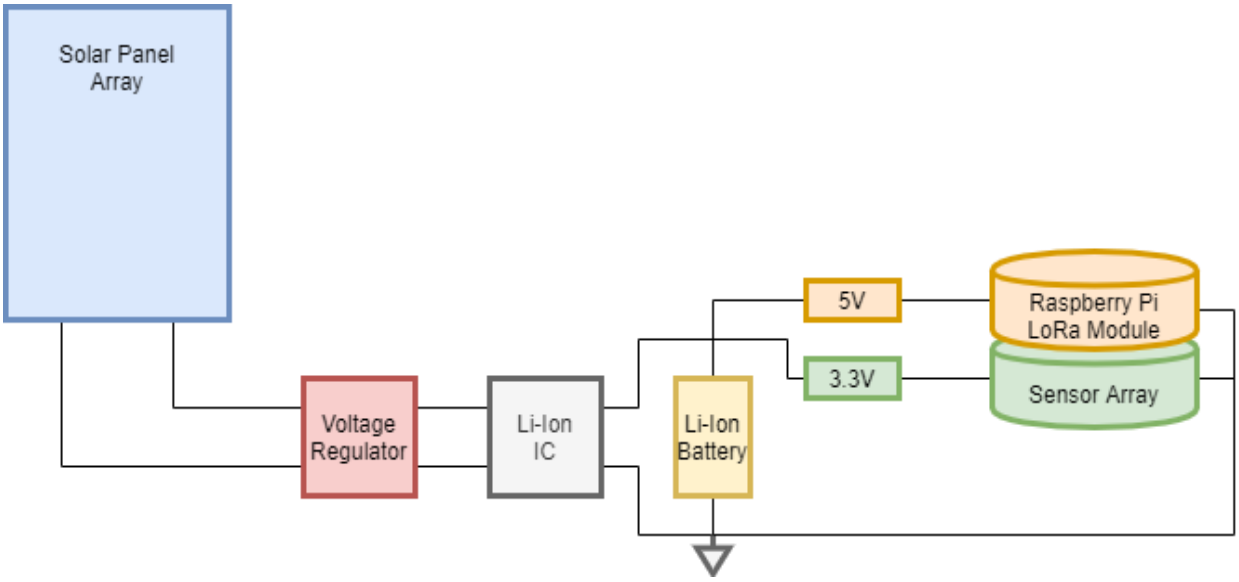


Figure 76: High Level System View

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 didn't 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 will operate in a cyclic fashion turning on to cycle through all the sensors to check if there is a fire under certain conditions. It will then process that information using computer vison on the Raspberry Pi for the infrared array and use state-space to store and check if the other sensors are within their constraints. This is depicted in the diagram below. 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.

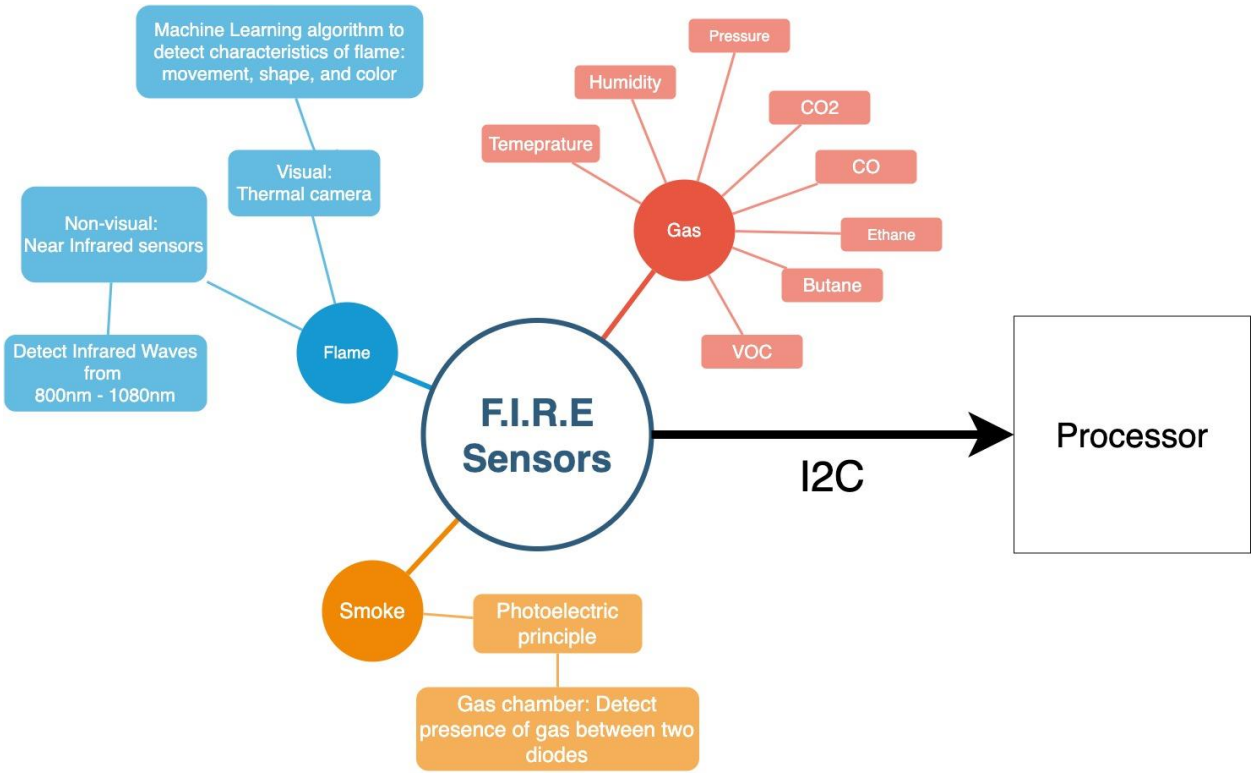


Figure 77: Sensors data sent to processor

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 9** 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 10** provides a detailed description.

Table 12: Division of Labor

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

Table 13: Division of Labor Breakdown

Engineering student	System components	Description
Noora	Flame Sensors: Near Infrared, Camera Smoke Sensors. Gas sensors Temperature/Humidity sensors	Noora will focus 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 will work closely with Arisa to send raw data for data analytics and processing.
Nicholas	Solar Panel Power Battery Charging Protection. Power Regulation	Nicholas will be 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 will choose appropriate solar panels that will efficiently supply enough power to the system; this will include selecting the right type, model, and size of panels. Subsystems will need a 3.3V and 5V supply, thus Nicholas will design 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 will be working on the communication between the devices. This will be done through radio frequency using the LoRa RF module. Jonathan will design the system to ensure communication between the devices is maintained and data from sensors can be sent to the hub using RF waves. Jonathan will test the range of the system for which data can be sent and design the communication network of the system.
Arisa	Sensor Data Processing Software Machine Learning' Raspberry Pi	Arisa will be responsible for interpreting the data coming from the sensors through the raspberry pi. By recording and analyzing the raw data from the sensors, Arisa will be able to train 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. The signal will be sent to the microcontroller to trigger the alarm.

8.2. Project Milestones

The following two tables provide the target milestones. Table 9 is the intended 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 are 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 will have a reasonable understand of the project's scope so that the prototyping stage can begin.

Table 14: 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 (components, functions, design)	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		
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 Acquire materials and components for prototype	April 06, 2020	April 17, 2020
15	Submit Final Documentation		April 21, 2020

Table 10 below shows the milestones for the summer 2020 semester. Summer is 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 marks the beginning of the projects prototyping, testing, and implementation stage. The aim is to complete fulfilling the project's intended purpose by July 31st. This final product, as well as electrical and CAD designs will be handed off to the sponsor. Moreover, the senior design 1 paper will be modified to reflect the project's realistic achievements.

Table 15: Summer 2020 Milestones

Week	Milestone (Tasks)	Start Date	Deadline
1 to 2	Assemble/ Build prototype Test components	May 11, 2020	May 22, 2020
3	Acquire components for final product Adjust documentation	May 25, 2020	May 29, 2020
4	Build final product's architecture	June 01, 2020	June 05, 2020
5 to 6	Integration Testing (hardware and software)	June 08, 2020	June 19, 2020
6 to 7	Make necessary adjustments	June 22, 2020	July 3, 2020
8	Final testing	July 6, 2020	July 10, 2020
9	Finalize product	July 13, 2020	July 17, 2020
10 to 11	Finalize documentation	July 20, 2020	July 31, 2020
11	Final Product		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 (Siemens STEM Day, n.d.). 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 (Siemens STEM Day, n.d.).

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 (Siemens STEM Day, n.d.). 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.

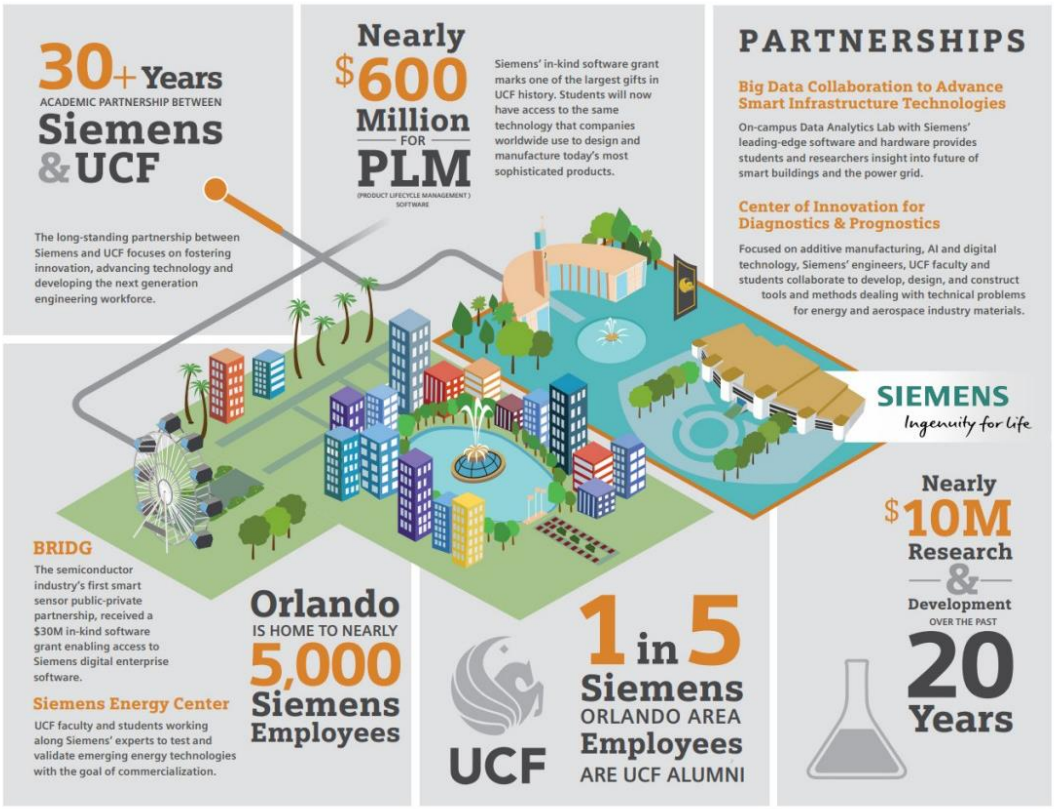


Figure 78: 30+ Years of Academic Partnership Between Siemens & UCF to foster the goals of Siemens Foundation (Siemens STEM Day, n.d.)

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 (Siemens STEM Day, n.d.). 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 (Siemens STEM Day, n.d.). 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 (Siemens STEM Day, n.d.).

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

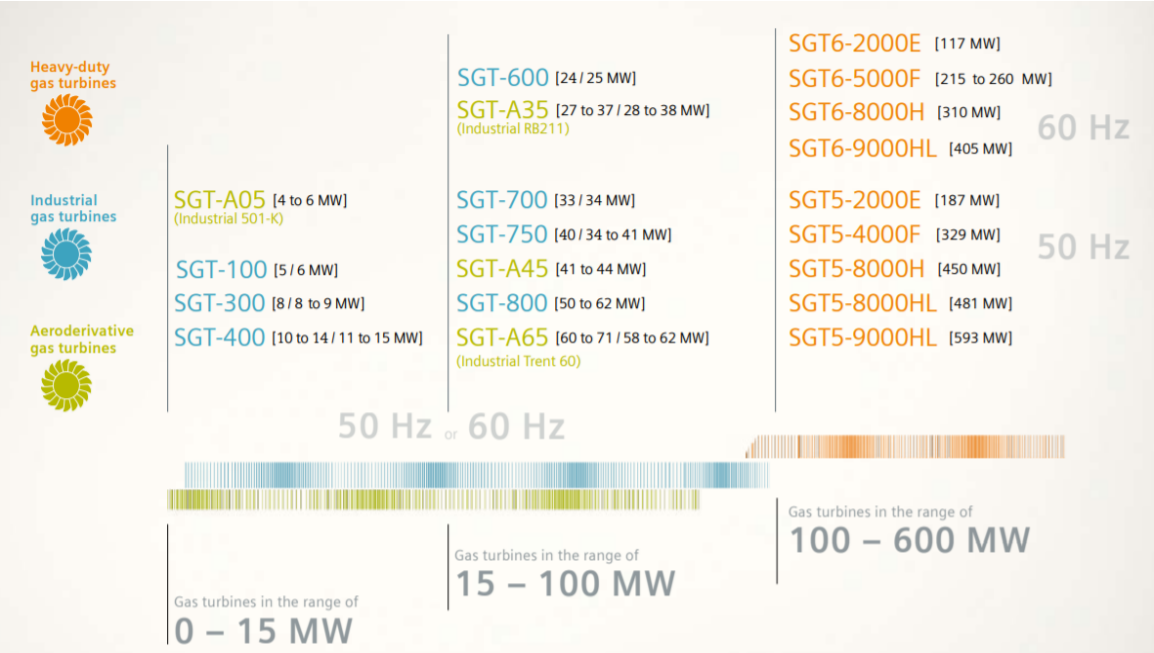


Figure 79: Overview of Siemens gas turbines (Siemens, 2019)

textiles. Siemens gas turbines fall into one of three categories: heavy-duty, industrial, or aeroderivative (Siemens, 2019; Nancy H Ulerich, 2013).

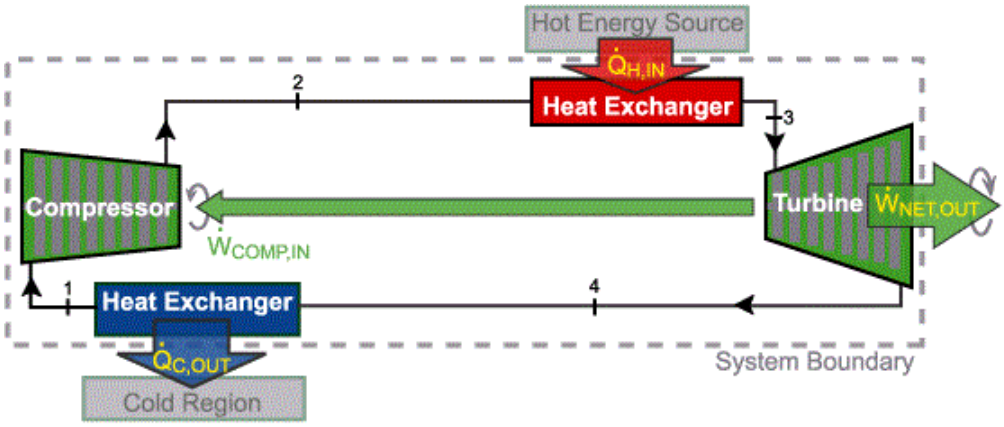


Figure 80: Typical gas turbine cycle as stated in (Isiadinso, 2015), the figure shows where a fire and gas sensor would be needed.

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 (Isiadinso, 2015; RAITHATHA, 2013).

A thermocouple is composed of two dissimilar metals connected together creating a junction through welding. (Isiadinso, 2015) One end of the connection is taken for reference and other end of the junction is used for measurement (Isiadinso, 2015). Temperature measurement is possible when there is temperature difference between the two junctions; this causes an electric current to flow in the circuit (Isiadinso, 2015). By understanding voltage-temperature relationships of metal combination, the temperature can be measured (Isiadinso, 2015). 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 (Siemens AG, 2019). The SGT-A05 KB/KH also uses the TOT Thermocouple or the TIT

Figure 81: Thermocouple used in Siemens SGT (Siemens AG, 2019)

thermocouple to improve overall accuracy in temperature monitoring (Siemens AG, 2019).
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” (Isiadinso, 2015).
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 (Isiadinso, 2015). Siemens SGT-750 uses infrared cameras to measure the temperature of the blade surface (Isiadinso, 2015). Temperature is recorded each rotation and is used for the cooling system (Isiadinso, 2015). Below is an image of the infrared temperature sensor used in the SGT-750.

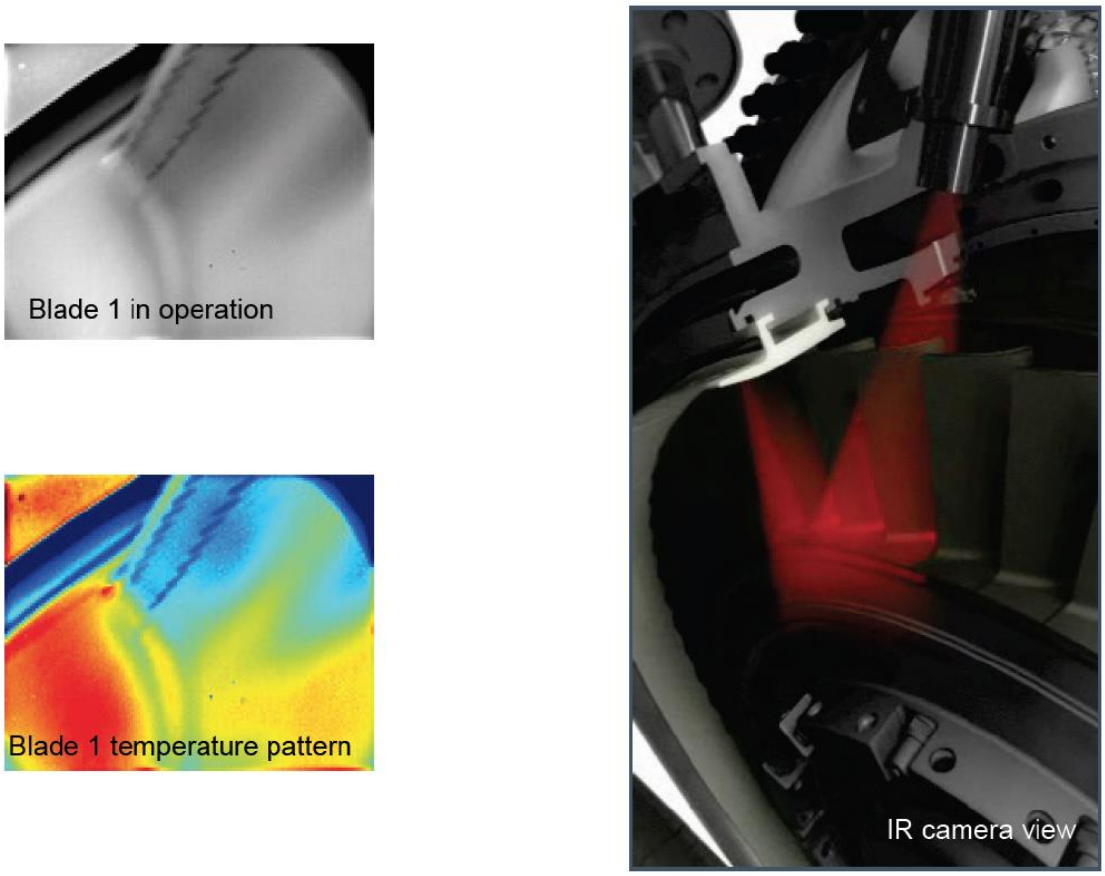


Figure 82: Infrared temperature sensor used in the SGT-750 (Isiadinso, 2015).

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 (Siemens, 2019). There are five phases that Siemens uses for successful IoT implementation detailed below in the diagram (Siemens, 2019).

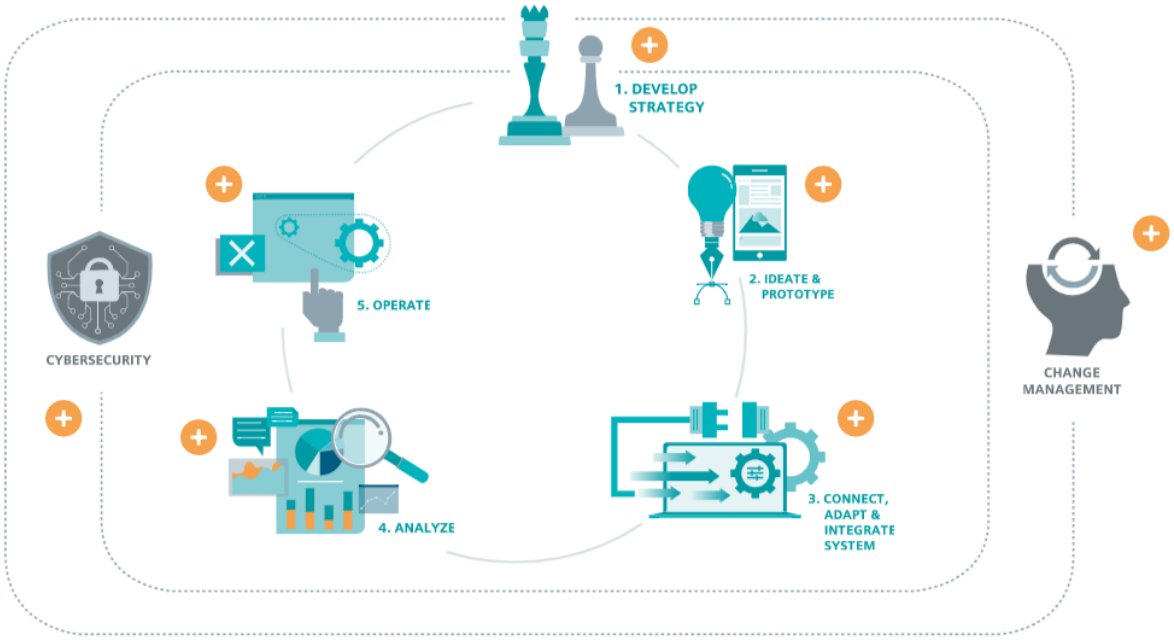


Figure 83: IoT integration cycle developed by Siemens

IoT as discussed previously in this paper has the potential to digitalize many industries 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 (Siemens, 2019). The data is then transferred and stored where it can be used to conduct data analytics and conduct machine learning algorithms (Siemens, 2019). From here, it can be used to provide insight for efficiency and create better business models (Siemens, 2019).

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 (Siemens, 2019). 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 (Siemens, 2019). This data is then used to forecast the city's air quality for the next 5 days (Siemens, 2019). 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 (Siemens, 2019).

Another case were Siemens was able to utilize IoT was in the case of the Sello shopping mall in Finland (Siemens, 2019). 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 (Siemens, 2019). 2 MW batteries were installed with a solar panel system that included a microgrid with smart building automation and cloud analytics (Siemens, 2019). The process took a few years using an iterative approach and followed the five phases depicted in the diagram (Siemens, 2019). 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 (Siemens, 2019). 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 (Siemens, 2019). This implementation helped reduce carbon emissions and saved the business €643,000 ($690,00) (Siemens, 2019).

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 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 send 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 (Siemens Gamesa, n.d.). 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 (Siemens Gamesa, n.d.). Siemens Gamesa's businesses is primarily

Figure 84: Nacelle of a wind turbine where the AFFS is installed

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 (Froese, 2016). 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 since it is designed with various mechanical and electrical components where a potential malfunction could start a fire (Froese, 2016). 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 (Spatuzza, 2020). Similarly, a G80 2MW wind turbine caught fire in Japan in 2017 (Foster, 2017). 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 (Froese, 2016). Repair costs are very high and put technicians who must conduct the offshore repairs at risk of injury or death (Froese, 2016). 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 (Froese, 2016).
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 (Froese, 2016).

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 (Garus, 2013). 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 (Garus, 2013). The system then activates nitrogen gas to extinguish the fires, operating on principles of oxygen displacement, using the Sinorix gas fire extinguishing system (Garus, 2013). 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 (Garus, 2013). The added extinguishing feature prevents the fire from spreading nearby and reduces the need for fire helicopters (Froese, 2016). 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 (Garus, 2013). 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 (Garus, 2013).

Figure 85: ASA fire detectors by Siemens

Figure 86: 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 (Garus, 2013). 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 16,** is an estimated list of costs associated with our project. A major target of this 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 Table 3 is a rough cost estimate on items that we believe could be implemented or critical to the system. The system will be composed of 3 to 4 devices that will communicate data with each other. Thus, the cost below illustrates the total cost of designing and implementing a system with multiple devices.

The table acts as a guide to see the general cost for the system and initial plan. Cost is determined by the distributor price when purchasing a single item, not in bulk. As we progress further into the project, potential areas to cut cost will become apparent through

careful research, design, and testing. The total cost below does not take into account the cost of thermal camera/sensor array.

Table 16: Estimated Cost

Item	Estimated Cost ($)
Solar Panel System	100
Sensors*	
Gas sensors	50
Infrared sensors (flame detection)	50
Particle sensors (smoke detection)	20
Thermal Camera / Sensor Array	200
Temperature	1
Humidity	1
Electronics*^	
Controller	20
General components (resistor, capacitors, inductors, connectors) ^	30
Specialized components (voltage regulation, MPPT, radio frequency) ^	30
PCB Manufacturing*^	60
Prototype (machine shop labor if applicable)	80
Development kit (for software)^	30
Miscellaneous (solder and jumper wires)	40
Total Cost**	≈ $500.00***

** Shipping not included in cost approximation.*
*** Assuming one of each was purchased and each is used in the final design. Some items in this list may not be used.*
**** Cost does not include thermal camera.*

^*This item is not* necessarily inclusive, i.e. it does not include administrative or other cost.

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

Abadi, n., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). *TensorFlow*. Retrieved from TensorFlow: https://www.tensorflow.org/lite

Alkhatib, A. A. (2017, Dec. 20). *Forest Fire Monitoring.* Retrieved Jan. 30, 2020, from https://www.intechopen.com/books/forest-fire/forest-fire-monitoring

Arduino fire alarm system using temperature and smoke sensor with Android connectivity. (n.d.). (Microelectronics Technologies) Retrieved Mar. 4, 2020, from https://www.projectsof8051.com/arduino-fire-alarm-system-using-temperature-and-smoke-sensor-with-android-connectivity/

Aslan, Y. E., Korpeoglu, I., & Ulusoy, Ö. (2012, Nov.). *A framework for use of wireless sensor networks in forest fire detection and monitoring.* Retrieved Jan. 30, 2020, from https://doi.org/10.1016/j.compenvurbsys.2012.03.002

ASQ. (2020). *House of Quality Tutorial - How to Fill Out a House of Quality | ASQ*. Retrieved from https://asq.org/quality-resources/house-of-quality

Bluetooth. (2020). *Understanding Bluetooth Range*. Retrieved Jan. 30, 2020, from https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/range/

Bosch. (n.d.). *Low Power Gas, Pressure, Temperature and Humidity Sensor.* Retrieved April 1, 2020, from https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf

Bouckaert, S., Poorter, E. D., Mil, P. D., Moerman, I., & Demeester, P. (2009). *Interconnecting Wireless Sensor and Wireless Mesh Networks: Challenges and Strategies.* Retrieved Jan. 30, 2020, from https://ieeexplore.ieee.org/abstract/document/5425861

De-Chang, W., Cui, X., Park, E., & Jin, C. (2013, Oct.). *Adaptive flame detection using randomness testing and robust features.* Retrieved Mar. 19, 2020, from https://www.researchgate.net/publication/257410367_Adaptive_flame_detection_using_randomness_testing_and_robust_features

Deshmukh, A., Breckon, T., & Dunnings, A. (2019, Dec 19). *fire detection cnn.* Retrieved April 18, 2020, from https://github.com/tobybreckon/fire-detection-cnn

Designer, A. (2018, 2 16). *Embedded RF Design: Ceramic Chip Antennas vs. PCB Trace Antennas*. (Altium Designer) Retrieved from https://resources.altium.com/p/embedded-rf-design-ceramic-chip-antennas-vs-pcb-trace-antennas

Donovan, J. (2012, 11 08). *Selecting Antennas for Embedded Designs*. (Convergence Promotions LLC) Retrieved from https://www.digikey.com/en/articles/selecting-antennas-for-embedded-designs

Dunnings, A. J., & Breckon, T. P. (n.d.). *EXPERIMENTALLY DEFINED CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE VARIANTS FOR NON-TEMPORAL REAL-TIME FIRE DETECTION.* Retrieved April 18, 2020, from https://breckon.org/toby/publications/papers/dunnings18fire.pdf

Edje, E. (2019, Oct. 19). *Tutorial to set up TensorFlow Object Detection API on the Raspberry Pi*. Retrieved Feb. 28, 2020, from https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/blob/master/README.md

Electronic Code of Federal Regulations. (1996, May 28). *e-CFR*. Retrieved March 17, 2020, from https://www.ecfr.gov/cgi-bin/text-

idx?SID=7248d37fdd25d0947f5611197fd5c6c8&mc=true&node=se47.5.101_1113&rgn=div8

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

Fonollosa, J., Solorzano, A., & Marco, S. (2018). *Chemical Sensor Systems and Associated Algorithms for Fire Detection: A Review.* Retrieved Jan. 30, 2020, from https://www.mdpi.com/1424-8220/18/2/553

Foster, M. (2017, August 22). *Gamesa turbine catches fire in Japan*. Retrieved April 18, 2020, from https://www.windpowermonthly.com/article/1442624/gamesa-turbine-catches-fire-japan

Frenzel, L. E. (2008, Aug. 13). *Welcome To Antennas 101*. (Electronic Design) Retrieved March 12, 2020, from https://www.electronicdesign.com/technologies/passives/article/21769333/welcome-to-antennas-101

Friis, H. (1946). *A Note on a Simple Transmission Formula.* IRE Proc.: 254–256.

Froese, M. (2016, June 24). *Fire prevention and protection for wind turbines offshore and on*. Retrieved April 18, 2020, from https://www.windpowerengineering.com/fire-prevention-protection-wind-turbines-offshore/

Garus, K. (2013, July 23). *Siemens' fire detection and extinguishing system is certified*. Retrieved April 18, 2020, from Offshore Wind Industry: https://www.offshorewindindustry.com/news/siemens-fire-detection-and-extinguishing

Gaur, A., Singh, A., Kumar, A., Kulkarni, K. S., Lala, S., Kapoor, K., . . . Mukhopadhyay, S. C. (2019, May 1). *Fire Sensing Technologies: A Review.* Retrieved Mar. 19, 2020, from https://ieeexplore.ieee.org/document/8625538

Ghoslya, S. (n.d.). *LoRa: Symbol Generation*. Retrieved from https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., . . . Adam, H. (2019, May 6). *Searching for MobileNetV3*. Retrieved Feb. 28, 2020, from https://arxiv.org/abs/1905.02244

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

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

IPC-2221. (1998, Feb.). *Generic Standard on Printed Board Design.* Retrieved March 5, 2020, from http://www.ipc.org/TOC/IPC-2221.pdf

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

Jamestown Distributors. (n.d.). *Kevlar Cloth - Plain Weave.* Retrieved March 18, 2020, from https://www.jamestowndistributors.com/userportal/show_product.do?pid=4022

Jurvélius, M. (2003). *FOREST FIRES AND INTERNATIONAL ACTION.* Retrieved Jan. 30, 2020, from http://www.fao.org/3/XII/0820-B3.htm

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

Keras. (n.d.). *Keras: The Python Deep Learning library*. Retrieved Feb. 29, 2020, from https://keras.io/

LEE, J. M. (2020, January 14). *Fighting Fire with Fire: How Controlled Burns Keep Us Safe*. Retrieved April 10, 2020, from https://www.ucf.edu/news/fighting-fire-with-fire/

Leens, F. (2009, Feb.). *An introduction to I2C and SPI protocols.* Retrieved Jan. 30, 2020, from https://ieeexplore-ieee-org.ezproxy.net.ucf.edu/document/4762946

Legal Information Institute. (2020). *47 CFR § 15.209 - Radiated emission limits; general requirements.* Retrieved March 10, 2020, from https://www.law.cornell.edu/cfr/text/47/15.209

Leng, F., Tan, C. M., & Pecht, M. (2015, Aug 6). *Effect of Temperature on the Aging rate of Li Ion Battery Operating above Room Temperature.* Retrieved April 16, 2020, from https://www.nature.com/articles/srep12967

Liqiang Wang, M. Y. (2011). Hybrid fire detection using hidden Markov model and luminance map. *Computers and Electrical Engineering, 37*, 905-915.

Melexis. (2012). *MLX90640 32x24 IR array.* Retrieved April 1, 2020, from https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90640

Mordvintsev, A., & K., A. (2013). *OpenCV-Python Tutorials*. Retrieved Feb. 29, 2020, from https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

Nancy H Ulerich, G. K. (2013). *CONDITION BASED MONITORING OF GAS TURBINE COMBUSTION COMPONENTS.* Orlando, Waltham, San Antonio: Siemens Energy, Inc., Jenetek Sensors, Inc., K Science, GP LLC.

Noci, J. (2020). *Antenna Design Overview — Copter documentation*. (Ardupilot.org) Retrieved March 10, 2020, from https://ardupilot.org/copter/docs/common-antenna-design.html

Nörthemann, K., Bienge, J.-E., Müller, J., & Moritz, W. (2013, Nov. 1). *Early forest fire detection using low-energy hydrogen sensors.* Retrieved Jan. 30, 2020, from https://pdfs.semanticscholar.org/b807/d5144095c15f7805fd272cb71a8a023a9516.pdf

OpenCV. (2020, April 18). *Open Source Computer Vision*. Retrieved April 18, 2020, from https://docs.opencv.org/3.4/df/d6c/group__ximgproc__superpixel.html

Ott, H. W. (2001, Feb. 14). *Henry Ott Consultants.* Retrieved March 8, 2020, from http://www.hottconsultants.com/techtips/freq-wavelength.html

Ouni, S., Ayoub, Z. T., & Kamoun, F. (2019, Jan. 16). *Auto-organization approach with adaptive frame periods for IEEE 802.15.4/zigbee forest fire detection system.* Retrieved Jan. 30, 2020, from https://doi.org/10.1007/s11276-018-01936-x

PyTorch. (n.d.). *PyTorch: From Reasearch to Production*. Retrieved from PyTorch: https://pytorch.org/

RAITHATHA, M. H. (2013). *SIEMENS-UV OPTICAL FLAME DETECTION.* Berkley: College of Engineering University of California.

Redmon, J. (2018). *YOLO: Real-Time Object Detection*. (arXiv) Retrieved Feb. 27, 2020, from https://pjreddie.com/darknet/yolo/

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement.* Ithaca, New York: Cornell University.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* Ithica, New York: Cornell University.

RF Wireless World. (n.d.). *Smoke Detector basics | Smoke Detector types*. Retrieved Mar. 19, 2020, from https://www.rfwireless-world.com/Articles/smoke-detector-basics-and-smoke-detector-types.html

RoHS. (2005). *RoHS Guide.* Retrieved Mar. 4, 2020, from https://rohsguide.com/rohs-faq.htm

Rouse, M. (2020, 2). *internet of things (IoT)*. (TechTarget) Retrieved from https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

Roy, S. S. (2018, Oct. 23). *Real-Time Object Detection on Raspberry Pi Using OpenCV DNN*. Retrieved Feb. 28, 2020, from https://heartbeat.fritz.ai/real-time-object-detection-on-raspberry-pi-using-opencv-dnn-98827255fa60

Sandler, M. (2019, Nov. 12). *MobileNet.* Retrieved Feb. 28, 2020, from https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet/README.md

ScienceDirect. (n.d.). *Kevlar.* Retrieved April 1, 2020, from https://www.sciencedirect.com/topics/engineering/kevlar

Semtech. (2015). *AN1200.22 LRa Modulation Basics.* Camarillo, CA: Semtech.

Siemens. (2019). *Siemens Gas Turbines.* Retrieved April 18, 2020, from https://assets.new.siemens.com/siemens/assets/api/uuid:10f4860b140b2456f05d32629d8d758dc00bcc30/gas-turbines-siemens-interactive.pdf

Siemens. (2019, April 1). *Siemens IOT Assest .* Retrieved April 18, 2020, from 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

Siemens AG. (2019). *Siemens Assets.* Retrieved April 18, 2020, from https://assets.new.siemens.com/siemens/assets/api/uuid:d9283b58-4f74-4f05-a8fb-0ccf0439ee82/version:1557162462/sgt-a05-service-solutions-2019.pdf

Siemens Gamesa. (n.d.). *Siemens Games Renewable Energy*. Retrieved April 18, 2020, from https://www.siemensgamesa.com/en-int/about-us

Siemens STEM Day. (n.d.). *Siemens Stem Day*. Retrieved April 18, 2020, from http://www.siemensstemday.com/

Smith, P. (2014, February 17). *Siemens develops automatic offshore fire-fighting system*. Retrieved April 18, 2020, from Wind Power Monthly: https://www.windpowermonthly.com/article/1281184/siemens-develops-automatic-offshore-fire-fighting-system

Spatuzza, A. (2020, February 3). *Recharge News*. Retrieved April 18, 2020, from https://www.rechargenews.com/wind/siemens-gamesa-investigates-after-wind-turbine-rotor-crash-in-brazil/2-1-749505

Syed, Z. A. (2016, Sept.). Frequency, Range and type of Wireless Communication. *NSA, University of Oslo.* Retrieved from https://its-wiki.no/images/3/3f/Frequency_range_Zyyad.pdf

Tan, J. (2019, Sept. 29). *How to Choose Battery for Your Emergency Lighting Wisely?* Retrieved from Sanforce: www.sanforce-tech.com/how-to-choose-suitable-battery-emergency-lighting-wisely/

This Plastic's on Fire! 4 Types of Flame Retardant Plastic Additives. (n.d.). (Craftech Industries) Retrieved April 1, 2020, from https://www.craftechind.com/this-plastics-on-fire-4-types-of-flame-retardant-plastic-additives/

True, N. (n.d.). *Computer Vision Based Fire Detection.* La Jolla, CA: University of California.

United Nations Enviroment Programme. (2020, Jan. 3). *Governments, smart data and wildfires: where are we at?* Retrieved Jan. 30, 2020, from https://www.unenvironment.org/news-and-stories/story/governments-smart-data-and-wildfires-where-are-we

WallpaperAccess. (n.d.). *Carbon Fiber.* Retrieved March 18, 2020, from https://wallpaperaccess.com/carbon-fiber

Xu, X. (2018, Aug. 10). *Everitt's blog*. (github) Retrieved Feb. 27, 2020, from https://everitt257.github.io/post/2018/08/10/object_detection.html

Yu, C., Mei, Z., & Zhang, X. (2013, Aug. 14). *A Real-time Video Fire Flame and Smoke Detection Algorithm.* Retrieved Feb. 28, 2020, from https://www.sciencedirect.com/science/article/pii/S1877705813013222

Zhao, T. X.-P., Acherman, S., & Wei, G. (2010, Oct.). *Dust and Smoke Detection for Multi-Channel Imagers.* Retrieved March 19, 2020, from https://www.researchgate.net/publication/47380702_Dust_and_Smoke_Detection_for_Multi-Channel_Imagers

Zima, D. (2020). Lora Best Design Practices EMC Compliance. *Lora Workshop, UCF.* Orlando. Retrieved from https://info.semtech.com/hubfs/RF%20Laboratories%202020%20UCF%20Workshop.pdf?utm_campaign=LoRa%20Boot%20Camps&utm_source=hs_email&utm_medium=email&utm_content=84057861&_hsenc=p2ANqtz-9DugGP-dmBjbpSMsTi9Ujxg9YtKcFV7fWKsulexvk7jHn4c0SP4JY6jja1MA4CO3Jvrr-3jZ