FireEscape

Authors: Aidan Wagner, Jason Ledon, Neha Tarakad
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A distributed system of fire detecting nodes that are capable of guiding occupants to safest path out of a burning building.This eliminates the risk of leading occupants towards hazards when trying to escape the buildings and prioritizes the safety of users. The nodes are able to dynamically plan the most optimal paths depending on distance as well as temperature and smoke data, reacting in real-time to threats of fire throughout a building.**

***Index Terms*—Design, Distributed, Fire, Optimal, Planning, Safety, Sensors**

1 INTRODUCTION

In a typical building fire occupants have a limited time to escape a building safely. In order to maximize the probability of a successful escape, it is essential that the escapee is provided with information detailing a safe and quick route from their current location to the outside of the building. Currently, this information is provided through fire drills based on floor plans posted on the backs of doors in crucial, high-traffic areas. However, this strategy presents a problem: The same fire that the occupants are hoping to escape from could potentially be blocking their pre-determined exit plan. This exit plan could lead them down a path toward a fire, wasting time that should be spent moving toward a valid escape. In the time it takes to traverse back toward an unblocked exit, it may be too late to escape unharmed.

As a remedy to the aforementioned problem, we propose a distributed system of nodes, capable of not only detecting fires but communicating with one another and dynamically providing the occupants with directions toward a safe exit route. These nodes would be positioned at key locations on multiple floors of a building, from long hallways, intersections, corners, and stairwells. These nodes work together to form a path that leads an occupant outside a building. The optimal path is determined through the floor plan of the building in conjunction with the readings of the smoke and temperature sensors that are attached to each node. The shorter the distance traveled combined with the lower the temperature and smoke levels, the better the path for the occupant to take. If a node goes inactive, the pathfinding algorithm will generate the most optimal path for the nodes remaining as each node will generate an optimal path from itself to the exit. In addition to the temperature and smoke sensors, the nodes will either contain LEDs to display arrows in the direction that the occupant should follow to find the next node or an LCD display that will contain more in-depth instructions on how to get to the next node as well as display the optimal path to take to efficiently and safely exit the building in the event of a fire. This solution will not only provide an innovative fire detection system but also inform occupants of real-time exit strategies which will give users the chance to avoid fires while they exit the building.

In our report, we will introduce our design and provide an in-depth analysis of our system implementation and trade studies that provide the reasoning for our design choices and how we intend on testing our system to ensure we meet our design and use case requirements.

2 USE-CASE REQUIREMENTS

See table

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Our system relies on temperature and smoke sensors to accurately detect the presence of a fire and use these readings as well as the floor plan of a specific building to generate the most optimal path for an occupant to take to safely exit the building. When we were designing this architecture, we wanted to prioritize the user's safety in getting accurate data measurements that would affect the path taken to leave the scene. In this way, we wanted to be able to create a solution that would address other difficulties that a building fire could cause such as power and wifi outages. Furthermore, we wanted to create a solution that would be scalable in the number of nodes and floor plan of different buildings as well as maintained on a regular schedule as to ensure up-to-date and working components.

Thus, we have divided our FireEscape architecture into three main components: fire detection, pathfinding and communication, and output instruction with hardware integration. Each of our modules has a critical responsibility and relies on the proper functionality of each other.

3.1 Fire Detection Module

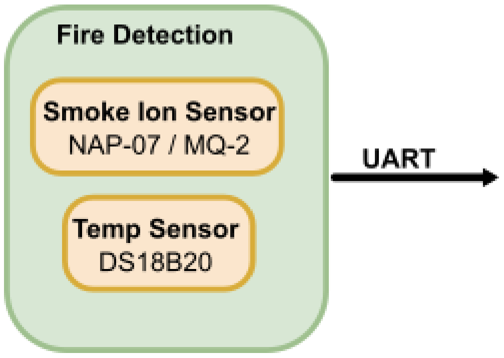


Figure 1: A snapshot of the section of our block diagram that handles sensor detection.

The Fire Detection Module is essentially the system that is within each one of our nodes that is capable of detecting a fire. Every individual fire detection node is comprised of temperature and smoke sensors that will work in conjunction to detect increases in heat and smoke levels consistent with a building fire. These nodes will be as discreet as current smoke detectors in buildings but placed in locations such as long hallways, intersections, corners, stairwells, and other critical points of buildings based on their floor plan.

As stated above, the purpose of the Fire Detection Module is for the nodes to individually mimic the functionality of current smoke detectors that are located in buildings. They have the purpose of determining whether or not that node is located at or within close proximity to a fire. The temperature and smoke sensors are driven by a microcontroller to obtain the real-time data readings for each node and compared against a threshold that would indicate a fire. The microcontroller that is driving each individual node must also be lower-power, such that it has the capability of being powered by a battery for 24 hours in idle mode and 5 minutes or longer in active mode in the event of a power outage; for our use-case, idle mode is defined as passively reading the sensor readings and reporting an ”ok” signal; active mode is defined as the state in which a node has sensed a fire, the node is actively pathfinding to find the optimal path out, and the frequency of status checks with other nodes has increased. While our plan is for the nodes to be hardwired to power, if we face power outages and we are dependent on an external backup battery source, we will be saving power by pulsing in and out of active mode. This microcontroller must also be able to communicate with the other nodes as well as pass information on each individual node to the pathfinding software.

3.2 Pathfinding/Communication Module

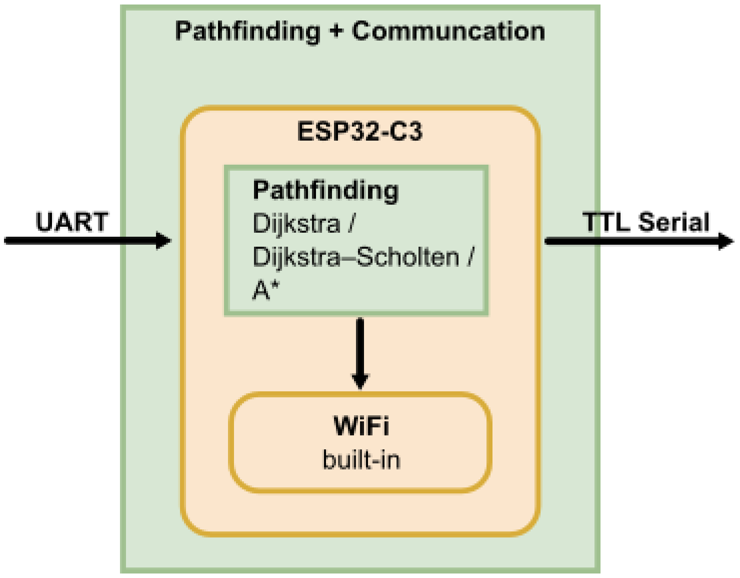


Figure 2: A snapshot of the section of our block diagram that receives sensor data from the current node, and sensor data from other distributed nodes, and with this information, plans the optimal path out.

The Pathfinding and Communication Module is essentially a single unit that has information on all of the individual nodes so as to generate the optimal path given the sensor readings and measurements from the inputted floor plans of the building of interest. This module is responsible for listening and gathering all of the fire detection sensor readings and comparing the data against each other; this data will then be used as one factor that will help generate the most optimal path. In addition, the pathfinding module is also responsible for using the inputted floor plan to generate a graph with weighted edges that depends on both the length of each path accessible from each node and the data readings coming from all node sensors. For example, the shortest path with the lowest temperature and smoke levels constitutes an optimal path. When generating the optimal path, we will consider the shortest path from each individual node to the exit in the event that a single node becomes offline so that we are always able to generate a path for the user.

Use-Case Requirement	Reasoning
100 second delay between node detection and occupant notification	NFPA requirement based on sprinkler system
Battery is recharged when power is on	The Pennsylvania Code General Fire Alarm Requirements
Must run for 24 hours during power outage	The Pennsylvania Code General Fire Alarm Requirements
Operate in fire mode for 5 minutes after 24 hours of no hardwired power and completely recharge battery once the power returns	The Pennsylvania Code General Fire Alarm Requirements
Fire must be detected 95% of the time	Current spec of smoke detectors, from National Fire Prevention Association
Planned paths are optimal and correct 100% of the time	Sending users toward a fire would mean project failure
Path planning will run and display on nodes in under 10 second for a system of 10 or less nodes	Time it takes to leave a room and therefore find a node
Distributed computation to allow scaling	System can be adapted to larger buildings
Recharge battery once power comes back on	Batteries must be recharged so that they will be ready for another fire

Table 1: Use case requirements paired with the reasoning behind them.

This module's primary work exists as a software algorithm running locally on the microcontroller; however, the only way the software is able to retrieve the data necessary is by receiving wireless and UART data from other sources. The range of communication will be dependent on each node utilizing the local WiFi network. However, to account for the possibility of a WiFi outage in the event of a building fire, we will show how we can utilize the ZigBee networking protocol which will be explained at a deeper level in the system implementation description.

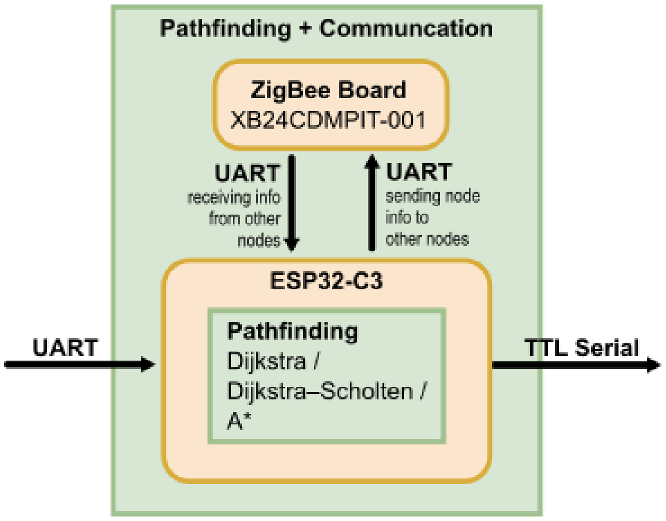

Pathfinding + Communcation
ZigBee Board
XB24CDMPIT-001
UART
receiving info from other nodes
UART
sending node info to other nodes
ESP32-C3
Pathfinding
Dijkstra / Dijkstra–Scholten / A*
UART
TTL Serial

Figure 3: A snapshot of the pathfinding section that is made to use ZigBee.

The software will have to maintain a method of determining which nodes are online and store real-time updates of their sensor readings to update the optimal path accordingly. With all of the data collected from the nodes, an optimal path will be generated and outputted for use by the occupant within 100s in accordance with our use case requirements which will take into consideration the alert upload time, download time, and the time to display instructions. The pathfinding software will be tested and analyzed to prevent all bugs and memory leaks.

3.3 Direction Instruction Module

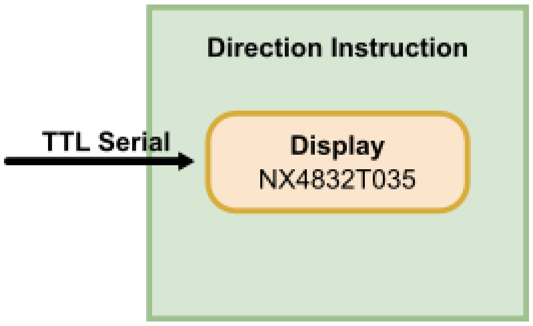


Figure 4: A snapshot of the section of our block diagram that handles directions out of the building using an LCD.

The Direction Instruction Module is also essentially the system of our distributed nodes as each node will individually have the functionality to provide directional instructions for the user to follow. This module will take the output from the Pathfinding/Communication Module in the form of the optimal path. Depending on the type of node, either directional arrows for relative direction will be provided or an in-depth set of instructions with the highlighted path to follow will be provided.

Our system of nodes is divided equally into two kinds: LCD nodes and display nodes: both kinds of nodes have the same fire-detecting functionality, but differ in the output instruction method. The LED nodes have five LEDs organized such that an arrow is lit up pointing to the next node in the path to follow.

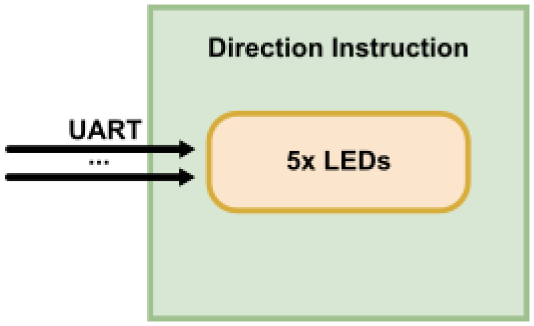


Figure 5: A snapshot of the section that shows the LED variation of Direction instructions.

The display nodes have a screen attached that will provide more details as to how to find the next node along the presented optimal path. In this way, we want to limit the confusion of the user in attempting to follow the nodes to the exit as it is a critical, quick-thinking setting.

As the Pathfinding/Communication Module updates regularly and returns an optimal path, it will be fed to the Direction Instruction Module for each node to keep the user up to date on the best path toward the building exit. This module heavily depends on hardware-software interaction in that the software module must be able to provide instruction that can be presented on the LEDs and display programmed through the microcontroller. The user will be interacting and heavily relying on the visual components of the nodes in this module meaning that there is a high priority on readability and accurate directional output.

4 DESIGN REQUIREMENTS

Drawing from our use case requirements, we have created a list of design requirements. These requirements specify aspects of our implementation that need to be met in order to ensure our product is beneficial to the user. The list is as follows:

- Alert upload time + Pathfinding time + download time + time to display directions less than 100s

- Battery capacity must be around 1600mAh
- Once power is restored, diode biases flip resulting in current charging the battery at 6.6mA (For our NiMH 2000mAh battery)
- Smoke and Temperature sensor threshold values are exceeded 95% of the time when exposed to flames
- Pathfinding software is tested and analyzed to prevent all bugs and memory leaks

5 DESIGN TRADE STUDIES

Please see the Word template and the guidelines on Canvas for more details about trade studies.

Trade studies of sub-systems can also be included in this section. You should use sections with the `subsection` command to split up this section as dictated by content.

5.1 Batteries

When we were selecting which batteries we wanted to use for our final product, the biggest concerns that we needed to take into consideration were the operating voltage of our system and the power consumption of our system. For nodes that used LEDs as displays, we knew that the largest operating voltage would be the ESP32 at 3.3V. For the nodes with LCD displays, we knew that the LCD screens would have the largest operating voltage, at 5V. To solve this, we felt that we could use AA batteries with a 1.2V potential, and use 3 of these to power the LED nodes, and 6 of them to power the LCD nodes. We felt that this would be the cheapest and most efficient option, as we would not need to order separate batteries for our two different nodes.

For the capacity, we needed to some some research on the power consumption of our components. We found that the ESP32 draws a maximum of 260mA while in active mode, and a maximum of 20mA in its modem sleep mode(Last Minute Engineers). We also found that the LCD display draws 500mA(NX4832T035). Additionally, we found that the Zigbee s2c pro draws 31 mA when fully active(Power Requirements). We will perform our calculations for an LCD node, as these be have the limiting factor for capacity. When a fire has not yet been detected, each node will periodically sleep for 25 seconds and then turn on for 5. One of our requirements is that we can operate in this state for 24 hours. To find out the capacity we needed, we used the following equation.

$$24 * ((50/60)(20) + (10/60)(260 + 31) = 1564 \tag{1}$$

We also need to ensure that we can power each node for 5 minutes after a fire is detected. In this case, the ESP32 will be in active mode, the LCD will be on, and the Zigbee card will be active.

$$\frac{5}{60} * (260 + 500 + 31) = 65 \tag{2}$$

We can see that the first capacity is our limiting capacity, so we have found that our battery will need a capacity of at least 1564mA.

Additionally, we need to design a circuit that will allow us to continuously charge our battery. We found that NiMH batteries can be continuously charged at $\frac{1}{300}$ of its capacity per hour, and we believe that this is an aspect that will be beneficial for our system. The batteries we have chosen have a capacity of 2000mA, so we will design our circuit to charge the batteries at 6.6mA.

5.2 Pathfinding

There are a lot of pathfinding algorithms available, so there was a significant amount of work that went into deciding which one we should end up using. While there is still some metrics testing that needs to be done before a final decision can be reached, there were some algorithms that we were able to quickly rule out.

5.2.1 DFS

DFS is a very popular pathfinding algorithm but wasn't particularly well suited for our application. DFS, by definition, traverses deep into the graph first, exploring further nodes earlier. This isn't ideal in our scenario, as we want to expand out first, trying to find the closest node that might lead us to an exit. While it has a runtime of

$$O(V + E) \tag{3}$$

It isn't compatible with the short-circuit exiting once the shortest path has been found.

5.2.2 BFS

BFS is the next logical alternative to DFS. BFS is actually much more suited for our application, as it expands outward first, checking the nodes nearest to it, and only then progressing deeper into the graph. This behaviour suits what we are looking for. The major downside is that BFS has no option for different weights of the edges. It has the same time complexity as DFS, but because of the lack of edge weights, we decided not to go with it.

5.2.3 Dijkstras

Dijkstras is the next logical alternative; BFS was very close to working but didn't support edge weights, and that is exactly what Dijkstras algorithm provides: it finds the shortest path out, prioritizing searching shorter paths first. It has a time complexity of

$$O(V + \lg V) \tag{4}$$

which is comparable to DFS and BFS, especially when you take into account that we will be allowing the algorithm to end early once a valid exit has been found. Additionally, these optimal exits should be found faster because of the fact that we are able to take into account edge weights.

5.2.4 Distributed Dijkstras

This is very similar to Dijkstras, except each node only has information about its immediate neighbors. This is to help with scalability and to allow nodes to be inserted into the graph later with minimal overhead. We think this might be a very viable option, but because of the increased amount of data that will be sent over the network, we will need to perform real-world tests to see if this performs better than base Dijkstra's.

5.3 Microcontroller

There are quite a few Microcontrollers available that would have been suitable for this task, but the one that we went with was the ESP32-C3. We specifically chose this microcontroller for a variety of reasons. 2) The ESP32 series has purposely been made to be compatible with the Arduino ecosystem; this is important to us because a significant amount of the hardware we were researching was listed as "Arduino compatible", therefore making it ESP32 compatible as well. 2) It was much more affordable compared to an Arduino Uno, which was important because we needed to make a system of nodes. 3) It has Bluetooth and WiFi built-in, which means we wouldn't need to buy adapter cards to support this.

One of the primary reasons that we decided to use the C3 variant of the ESP32 was that it is using a RISC-V based processor, allowing it to be more power efficient than some of the other variants. This was a key consideration as we have tight power constraints due to Fire Code.

5.4 Networking

There are a variety of different wireless communication protocols that we could use. The main 3 are WiFi, Bluetooth and ZigBee.

5.4.1 WiFi

This method is easy to use, however, it has the risk of going out during a fire. A local Zigbee network may be a better solution in the event of a WiFi outage. Using the local WiFi is a choice based on cost and resource availability but we want to be able to show that we are capable of scaling up and addressing possible risks.

5.4.2 ZigBee

This is what we deemed to be the ideal networking for our use-case. ZigBee is primarily used for IoT devices where there isn't 1 central hub that every device talks to. That is ideal in our case, because we don't want one central hub; if the fire were to take down that hub, the entire network would go down. ZigBee is ideal because it allows us to set up our own fully distributed network. We are choosing to rely on WiFi for the sole reason that it is already built into the boards, and the cost of buying XBee boards would push us over budget. Therefore we have come to the decision to use a few XBee boards for a proof of concept but rely on WiFi for the general functioning of the project.

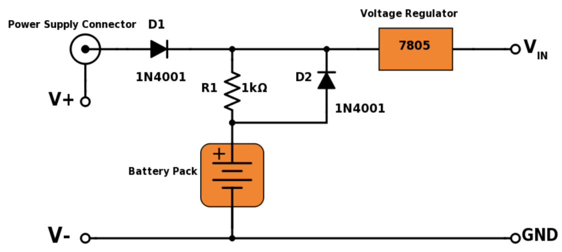
6 SYSTEM IMPLEMENTATION

Our system implementation can be broken into descriptions for the 4 following subsystems: Circuit design, sensor interfaces, communication software, and pathfinding software. section 3 rather than redundant. You can refer back to earlier figures in section 3 using Fig. 1, Fig. 2 and Fig. 4.

There should be a subsection for each of the subsystems as shown below.

6.1 Circuit design

To connect all of the components of an individual node in our system, we will be designing a Printed Circuit Board (PCB). This board will contain traces allowing us to connect our temperature and smoke sensors to our ESP32 microcontroller. Once we have acquired these PCBs we will solder our components to the board, creating a clean final product that is more durable than a breadboard.



Additionally, our PCB will contain the circuitry needed for our backup battery. The circuit that we are planning on implementing is shown in the above diagram. The D1 Zeener diode is used to prevent any power leaking from the backup battery toward the main power supply. The resistor allows for our backup battery to be consistently charged at a slow enough rate to prevent overcharging. If the main power is to go out, which is very possible in a fire, the backup battery will be able to utilize the low resistance path through the D2 zeener diode, which will then provide power to our microcontroller and any other components that need power to function. We found that NiMH batteries can be continuously charged at $\frac{1}{300}$ of its capacity per hour, and we believe that this is an aspect that will be beneficial for our system. The batteries we have chosen have a capacity of 2000mA, so we will design our circuit to charge the batteries at 6.6mA(Smith). By adhering to this guideline we will be able to keep our backup battery charged in the case of a power outage.

6.2 Sensor interface

To detect fires, each node will have a temperature sensor and a smoke sensor. We are using the Ds18B20 temperature sensor and the MQ2 smoke detector. Until the detection of a fire, our nodes will spend the majority of their time in sleep mode to preserve battery power. Once every 30 seconds, our system will switch into an active mode. In this active mode, measurements will be taken of the environment, and if the sensor readings exceed the threshold value that we set, a fire alert is sent out to all of the other nodes. We have decided to use a threshold of 135 degrees Fahrenheit as our temperature threshold, as this is a typical temperature for fire, but not a temperature that would be encountered on a daily basis. For the smoke detector, we do not have a smoke threshold yet, as we have not yet tested with the smoke sensors. Once we begin testing with the smoke sensors we will determine a valid smoke measurement threshold.

6.3 Communication

In order to provide occupants with real-time dynamically changing directions, our nodes will need to have a fast and reliable method of communication between them. While idle, nodes will periodically wake up (once every 30 seconds), and during this awake period, they will send a heartbeat message to the other nodes. Additionally, the node will also check for messages from other nodes in the system. Here, there are three scenarios: 1) Node A receives a heartbeat message from Node B, 2) Node A receives a fire alert from Node B, 3) Node A does not receive a message from node B. In case 1, Node A continues on and does not take any new actions. In cases 2 and 3, Node B is assumed to be in contact with fire, and Node A switches to active mode until the threat is resolved. In this active mode, the path-planning software will begin running.

We present two possible methods of communication between the nodes in our system. The first method is allowing each node to access the WiFi of the building in which the FireEscape system is installed. After this, the nodes are able to communicate with one another. This method allows for simple installation and does not require the setup of an external network. Additionally, many microcontrollers come with onboard WiFi compatibility. However, using WiFi introduces the risk of WiFi outages due to a fire.

To solve this issue, a local ZigBee network can be established between the nodes. This will allow the nodes to communicate even in the event of a WiFi outage. For our demonstration, we will be using WiFi communication across all of our nodes, but we will also provide a zigbee demonstration between two of our nodes to prove the validity of this alternate solution. Ideally, a system would operate entirely on Zigbee communication, however, due to supply and budget issues, this is not feasible for us.

6.4 Pathfinding Software

Once a fire has been detected in our system, all of the nodes will continuously be running their pathfinding software in order to determine the safest route to the exit. We are currently developing two different methods of pathfinding, which we will test on our finished hardware to determine the optimal method. The first method we are developing is a standard Dijkstra's search. In this strategy, each node contains information about all of the other nodes as well as information about the graph that represents the building floor plan. The benefit here is that pathplanning is able to be done without the need for other nodes to generate their own shortest paths out, which would hopefully reduce the time it takes to get an initial display of the path out. That being said, while the node is no longer relying on the data for the shortest path out from other nodes, it is still waiting for the sensor readings from other nodes.

Our other method is the Dijkstra-Scholten(Mani) algorithm redinclude citations. In this algorithm, each node is only required to receive information from each of its neighbors. The nodes which have exits as neighbors set the edge to the exit as their shortest path. Then this shortest path is passed to the other neighbors, who use this information to develop their own shortest paths. One advantage that this has over the previously mentioned Dijkstra's algorithm is that we expect it to scale better with larger systems. As each node only performs computation involving its neighbors, we believe that scaling up the system will not excessively increase the computation cost if we implement this version of pathfinding. That being said, we would still need to conduct Metrics studies to see at what point the added latency of sending each node's shortest path outweighs the cost of every node doing pathfinding out of the entire building: what magnitude of the number of nodes would one option surpass the other.

The directions from this pathfinding software will then be sent to our LED and LCD displays to guide the occupants toward the exits. For LCD's we are using NX4832T035's; these displays should allow us to put a detailed description out of the building: with a resolution of 480x320, we will be able to display the floorplan of the building, highlighting the best path out, or at the very least, the layout of the next few nodes along the shortest path. The LED variation of the node will be used in less critical areas, such as 3-way intersections which have limited paths; each node will simply point in the relative direction of the next node that falls along the shortest path out of the building.

7 TEST & VALIDATION

Since the nature of FireEscape relies on multiple moving parts, we have planned to create an extensive testing plan for each individual component as well as the integration of parts that they rely on. The motivation behind this testing plan is to ensure that the subsystems work well individually before attempting to integrate them together such that we are confident in our ability to scale up the system. Below, we have five important functionalities to test: the ability to detect a fire, the path finding algorithm, the communication between nodes, displaying directions on the LEDs, and generating the path and instructions on the display nodes.

While we will go into depth the testing plan for each categories, we also want to ensure that we are testing for performance and power usage that we are offering the user the best case scenario as well as preparing for outages. While our system would be connected to power, we do want to ensure that in the event we lost power, we need to rely on our batteries and use them wisely before and ensure they can last long enough before they can be fully recharged once we regain power. In order to do so, we plan to test the current draw and voltage levels for each node when in active and idle mode to order to ensure that there isn't too much current being drawn at each instance. We should also be keeping track of how much power is consumed by each node over a couple of days when utilizing the backup batteries in order to figure out the time it will take for the charge to be depleted before it needs to be charged again. This would determine if we meet our requirements and allow us to develop a maintenance plan.

7.1 Ability to Detect Fire

Evaluating the correctness of an individual node's ability to detect a fire will be based on fire thresholds obtained from researching the fire code. The threshold we will set for the temperature sensor will be 135^{o} Fahrenheit. The smoke sensor doesn't have the same threshold-like boundary that needs to be passed, though: the smoke sensor will need to be calibrated to the base state of the air quality in the room it has been placed in. After this setup calibration has been done, the sensor will be able to detect changes in air quality; when smoke enters the chamber, the flow of ions between two places will be disrupted, activating the output of the sensor. While it would be hard to simulate these thresholds, for the sake of our smaller-scale demo we will be utilizing a candle as our test fire and bringing it close to these sensors. Based on the temperature and smoke readings of the two sensors when the candle is near, we can determine our smaller-scale threshold for fire detection. One risk here is our ability to test the high temperature consistent with a building fire and how the flame would have to be very close to detect a high temperature for our test input. In an actual building fire, we would expect the surrounding air to be at a very high temperature, but when we are testing with a candle it is harder to mimic that scenario. So for our testing plan, we will be utilizing the thresholds based on the candle being placed near the sensor or not and ensure the sensors detect its presence.

7.2 Path Finding Algorithm

Evaluating the correctness of the path-finding algorithm relies primarily on test cases for the algorithm itself. We can create example graphs with different edge weights in order to determine if our algorithm outputs the best possible path from each individual node. Once this is confirmed, we need to ensure that after each node provides both temperature and smoke data, that we are able to create a graph with an example floor plan and be able to generate an optimal path towards an exit. We plan to experiment with mimicking a node going offline and ensuring our algorithm is able to generate a new optimal path in the event that this occurs.

With these nodes, edges, and exit locations that help create our graph, we want to ensure that we are outputting correct and optimal paths to exit from each of these nodes. One risk that we could encounter is the latency between nodes. In the event that we use the base implementation of Dijkstra's or A*, it is possible that on larger buildings - and therefore larger graphs - we could run into long computation times, as every node is doing the pathfinding for the entire building. That being said, if we do the distributed version of Dijkstra's - Dijkstra-Scholten - we will be sending significantly more data over the network, and pathfinding would not be able to commence until nodes have received the shortest paths from their neighbors. Therefore, while computation would decrease from each node, the time spent waiting would be proportional to (the time it takes for communication between two nodes $\cdot$ the number of nodes in the shortest path).

7.3 Communication Between Nodes

In order to test the communication between nodes, we are planning on sending arbitrary messages to the transmitting nodes. In order to test that they are being sent properly, we will just ensure that the information that was transmitted is being displayed correctly on the receiving node. Initially, we can test this functionality through arbitrary messages. Later on, we will be testing this functionality with the temperature and smoke sensor readings and ensure they are received properly as well as tracking the changes as they are updated.

Furthermore, we would want to be able to test the offline node scenario too to ensure that we are able to properly determine which node has gone offline and which nodes continue to transmit data. One risk that we might encounter would be if we encounter dropped messages. We would want to analyze why this could be occurring and deal with potential causes like range. While in our implementation we are relying on the local WiFi network, we would also want to test for our Zigbee protocol between the two nodes that we will use to demonstrate this capability. As a result, we would want to perform communication testing relying on the local WiFi network for all of our nodes as well as relying on Zigbee between the two nodes that will utilize this feature.

7.4 Display directions on LEDs

To evaluate if we are able to correctly display directions on LEDs, we will start by having a program downloaded onto the microcontroller that specifies a direction. We expect the LEDs to match that directional command in the form of an arrow (north, east, south, or west). Once we can confirm this functionality, we will generate a test path made up of different directional arrows corresponding to indices of all of our nodes and ensure that once the path is outputted, we match the correct direction with the corresponding node. Finally, we will use the generated path from our pathfinding algorithm as our test input and ensure that the correct node matches up with the correct directional arrow. One risk that we might run into is that due to the spacing between nodes, there might not be sufficient information for the user to have in order to follow the path to find the next node. However, this is why we decided to include the display nodes as well to provide supplemental, in-depth instruction to ensure the user is aware of the path they must follow to efficiently and safely reach the exit.

7.5 Generate Path and Instructions on Display

To evaluate if we are able to correctly display directions on the display, we will start by having a program downloaded onto the microcontroller that specifies a set of instructions. We expect the display to present those instructions. Once we can confirm this functionality, we will generate a test path that will resemble a floor plan. We will ensure that we are able to depict it clearly on the display and that it is readable by the user. As the user will be depending on these instructions to determine where to go next, we are taking the testing very seriously because we don't want to confuse the user and lead them the wrong way. If we are not comfortable with the level of precision of the path, we will keep working until it is up to standard. We could even have third party members try the path that is displayed and ensure that they are able to follow the path to the exit without a lot of difficulty. One risk that we may encounter is the balance between written instructions and actually displaying a floor plan map with the nodes highlighted and the path to follow in bold. We believe the written instructions to find the next node will be very helpful but it is also extremely useful to have a map visual as well. We are going to test both methods and see which is clearer and has a higher success rate in terms of readability to get to an exit.

8 PROJECT MANAGEMENT

8.1 Schedule

Refer to the Gantt chart of our schedule attached at the end of the document. Currently, we are on schedule to meet our MVP requirement which involves the optimal path generation based on an inputted floor plan and display of easy-to-follow instructions to direct occupants out of the building based on the fire detection from our distributed node system by the due date. As we continue to make progress on our project, we continuously check and update our schedule to keep it as up to date as possible if certain tasks take longer or shorter than anticipated. We also take into consideration the turnaround time for the hardware we order based on their estimated delivery date wherein we work on other tasks while we wait to integrate new pieces. The schedule is shown in Fig. 9.

8.2 Team Member Responsibilities

Our team member responsibilities are divided into areas of interest and expertise. We have assigned a primary owner as wel as a secondary owner to each task so that we can keep each other accountable as well as bounce ideas off of one another to progress quickly. With that being said, Aidan Wagner is the primary lead on the software with regards to the pathfinding software and communication between the distributed nodes with Jason Ledon as the secondary lead. Neha Tarakad is the primary lead on the circuitry with regards to planning out the node structures and PCB for fabrication and collecting data from the sensors and passing them to the software for path calculation with Aidan Wagner as the secondary lead. Jason Ledon is the primary lead for the hardware-software integration and outputting the results from the pathfinding algorithm to the displays and LEDs with Neha Tarakad as the secondary lead. At the moment, our teams deadlines have not been extremely reliant on each other. This means that when one of our teammates has missed a deadline, the other members have not been strongly influenced. As a team, we all have experience with circuits, hardware, and software so we intend on helping each other out in the event that one of us gets stuck and are not making progress.

8.3 Bill of Materials and Budget

Refer to Bill Of Materials Table 2. Unfortunately, the hardware we needed was not in inventory from previous years so we had to purchase from scratch. However, we ensured that we would be finding our materials at reliable sources at the best possible prices to stay within budget. We have yet to order PCBs for fabrication but based on our attached budget we are allocating the remaining money for this purpose.

8.4 Risk Mitigation Plans

Throughout our design process, we have encountered design, resource, and shipping setbacks that would result in us pushing back some of our tasks more than we would've anticipated. We have already ordered the majority of our hardware that would allow us to make our nodes with the exception of the remaining temperature and smoke sensors as we wanted to test out a couple to ensure they work to our

Table 2: Bill of materials

Description	Model #	Manufacturer	Quantity	Ind. Cost	Tax/Shipping	Total
RISC V Developer Board	ESP32-C3	Adafruit	10	$9.95	$21.16	$120.66
Lithium Polymer Batteries	LP603449	Amazon	10	$5.30	$0.00	$53.00
Temperature Sensors	DS18B20	Adafruit	10	$3.95	$14.36	$22.26
Smoke Sensors	NAP-07 NIS-07 HIS-07	Ebay	10	$2.99	$4.40	$34.36
HMI Touch Display	NX4832T035	Itead	4	$39.90	$30.00	$189.60
LEDs	L513SRD-C	18-220 Lab	30	$0.00	$0.00	$0.00
Embedded RF modules	Digi XBee S2C DigiMesh 2.4	DigiKey	2	$33.10	$11.39	$77.59
						$497.47

liking and serve the purpose we intended. We have learned the importance of ordering from reliable distributors as to ensure datasheets as well as a timely delivery date. For example, our smoke sensors were purchased from ebay and they took almost a month to arrive and the datasheet corresponding to the sensor might require an additional chip which we didn't anticipate. While we tried to plan for these delays, we ran into resource issues such as limited stock or simply the hardware needed different functionality than advertised. We are also actively planning for setbacks due to complications in implementation. At the beginning stages of our project, we haven't run into major issues with regards to staying on schedule. However, we do want to anticipate difficulties that could arise when getting into the depths of the pathfinding algorithm based on life sensor data from each node, integration between hardware and software, and final displays based on the output of our pathfinding algorithm.

One of our main risks at the moment is the fact that we have two types of node structures (LEDs vs displays) which essentially where we run the risk of having conflicting specs for our varying hardware. To account for this, we have set up extensive testing time for specifically how we will get an optimal path to present arrows on the LED nodes and the in depth instruction and optimal path to populate the display. While this feature depends on the output from the pathfinding algorithm, we have set up time to simply learn to interface with both the displays and LEDs to minimize potential difficulties down the road.

We also wanted to ensure that we planned out major deliverables such as the proposal and design presentation, design report, ethics assignment, and the final presentation and report. In this way, we are not only accounting for tasks related to our project implementation but also planning to spend ample time preparing our supplementary materials.

9 RELATED WORK

We have been using some of the work done by team A3 - FreeSeats of ECE500 Fall 2021 as a reference, as they had a somewhat similar structure as us: they had a system of individual nodes that were communicating over ZigBee. There are some key differences: they had a central hub that they were talking to, whereas we are completely decentralized; they had a cloud-based solution, whereas we are doing all computation locally and aren't making a website. that being said, there are enough similarities that we have been able to draw inspiration from some of the design decisions and diagrams that they made.

Another team that we have been able to use for design inspiration is team our TA's, Kaashvi Sehgal. Her team worked with a lot of the same hardware that we are using: they used an ESP32 for the general brains of each node and temperature sensors. There are also some overall similarities as well; their team also had a distributed node system with power constraints, which used a wireless form of communication. The form of wireless communication that they used differs, but overall, there is a lot of similarity between our two projects. Because of that, we have been able to refer to their design documents to get some sense of what ours might resemble.

10 SUMMARY

With FireEscape we are working to create an efficient and safe way to help occupants evacuate a building in the event of a fire. With our distributed node system, we can provide real-time data of our fire detecting nodes and allow the system to communicate with one another and work together to output the safest, shortest path for the user to follow to guide them towards an exit.

10.1 Public Safety Considerations

Our project aims to prioritize the health and well being of occupants in a building. While currently there are fire drills and evacuation plans, we want to be able to provide effective instruction and ensure the safety of our users. We don't want to run the risk of occupants trying to leave a building and instead, putting themselves at a greater risk by walking towards greater hazards. While sometimes fire drills are not taken seriously, we want to ensure that if users have not seen and are kept up to date with their building's evacuation plan, that they are able to be provided with descriptive guidance on where to go to safely depart.

10.2 Lessons Learned

Our team has learned a lot of important lessons up until this point and we anticipate to keep learning throughout this process. We learned the importance of extensive planning when it comes to design choices and how it can take longer than we expect due to resource availability, shipping, and cost. We learned what goes into deciding what hardware to use for the functionality of our project and how to make those design choices as a team with regards to how we prioritize our needs. We also learned how important it is to get hardware from reliable sources so that the parts can arrive in a timely manner and provide good documentation.

Glossary of Acronyms

- LED – Light Emitting Diode
- LCD – Liquid-Crystal Display
- ESP32 – Microcontroller developed by Espressif Systems
- DFS – Depth First Search
- BFS – Breadth First Search
- PCB – Printed Circuit Board
- UART – Universal Asynchronous Receiver / Transmitter

References

Smith, Jason Poel. “Create Your Own Battery Backup Power Supplies - Projects.” All About Circuits, 22 Feb. 2016, https://www.allaboutcircuits.com/projects/battery-backup-power-supplies/.

K. Mani Chandy and J. Misra. 1982. Distributed computation on graphs: shortest path algorithms. Commun. ACM 25, 11 (Nov 1982), 833–837. https://doi.org/10.1145/358690.358717

Power Requirements, 2018, https://www.digi.com /resources/documentation/Digidocs/90002002/Content /Reference/r_specs_power_reqs.htm?TocPath= Technical+specifications%7C_____2.

Last Minute Engineers. “Insight into ESP32 Sleep Modes amp; Their Power Consumption.” Last Minute Engineers, Last Minute Engineers, 1 Feb. 2023, https://lastminuteengineers.com/esp32-sleep-modes-power-consumption# ˜ text=ESP32%20Active%20Mode,-Normal%20mode%20isamp text=Since%20everything %20is%20always%20active,and%20Bluetooth%20are%20used%20simultaneously.

Mahoney, Shawn. “Fire Alarm Notification Delay from Sprinkler Waterflow: NFPA: NFPA.” National Fire Prevention Association, https://www.nfpa.org/News-and-Research/Publications-and-media/Blogs-Landing-Page/NFPA-Today/Blog-Posts/2022/06/03/Fire-Alarm-Notification-Delay-from-Sprinkler-Waterflow: :text=1%2C%20NFPA%2072%20permits%20up %20appliances%20within%20the%20building.

“NX4832T035 – Nextion 3.5’ Basic Series Hmi Touch Display.” ITEAD STUDIO OFFICIAL, 6 Sept. 2022, https://itead.cc/product/nx4832t035-nextion-3-5-basic-series-hmi-touch-display/.

“034.” Pennsylvania Code, http://www.pacodeandbulletin.gov/Di =%2Fsecure%2Fpacode%2Fdata%2F034%2Fchapter50%2Fs50.53.htm

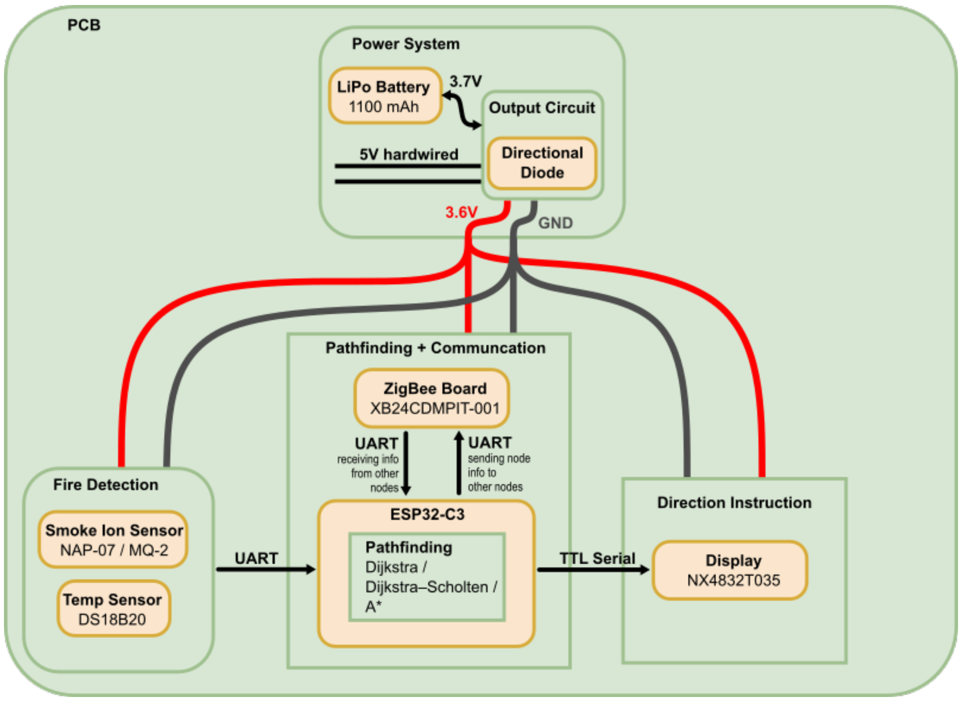


Figure 6: A full-page version of the same system block diagram as depicted earlier.

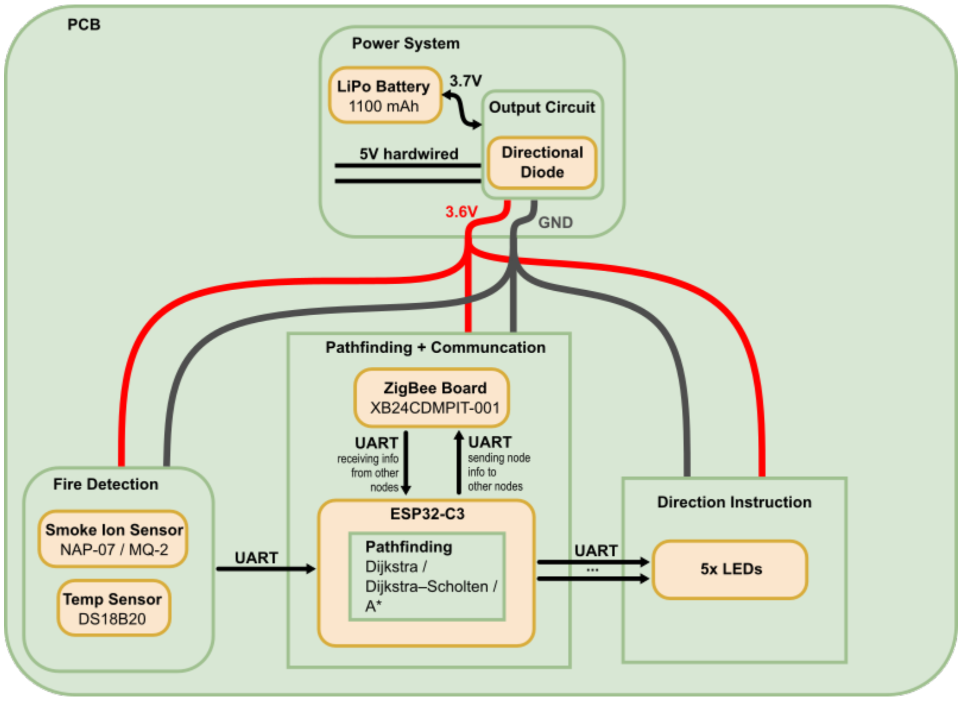


Figure 7: A full-page version LED variant of the block diagram.

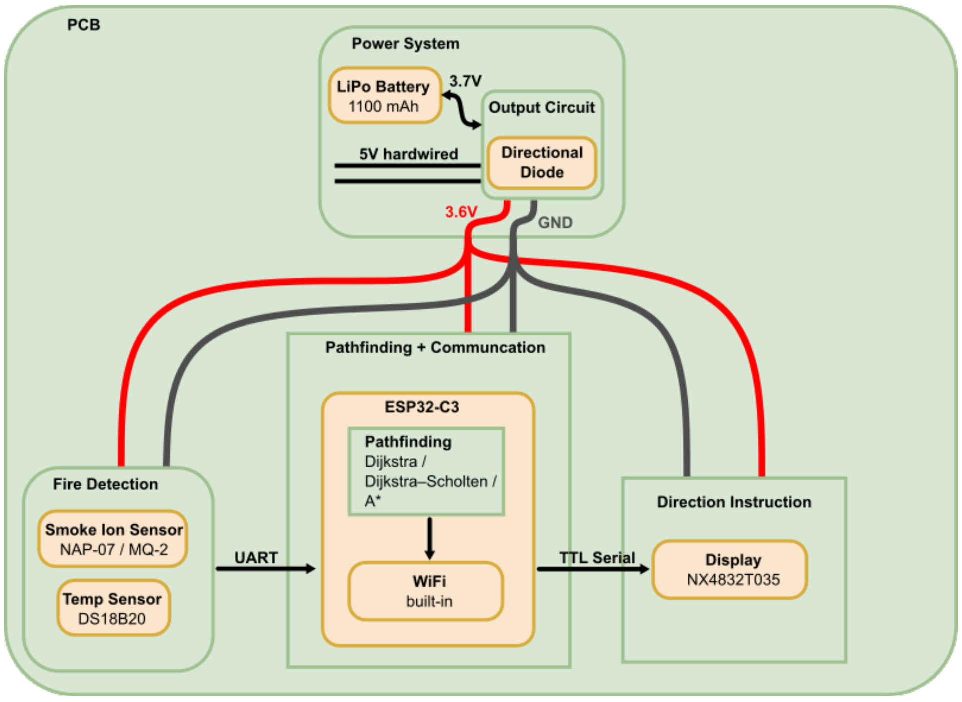


Figure 8: A full-page version WiFi variant of the block diagram.

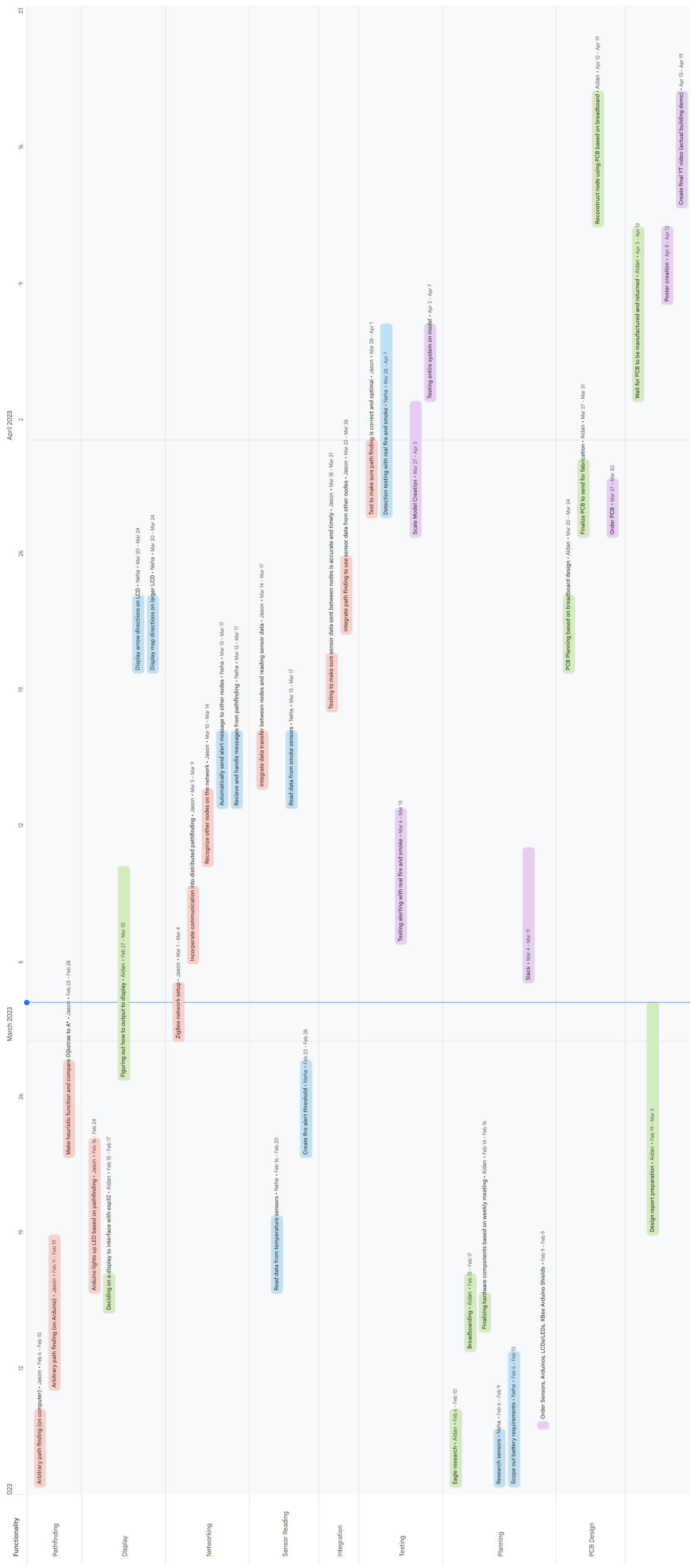


Figure 9: Gantt Chart