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 through the use of LEDs for relative direction and an LCD display for in depth information. 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. This provides an innovative fire detection solution to inform occupants of real-time exit strategies which will give users the chance to avoid fires while they exit the building.

2 USE-CASE REQUIREMENTS

Use Case Requirements	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

Figure 1: Table of our case requirements paired with the reasoning behind them.

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

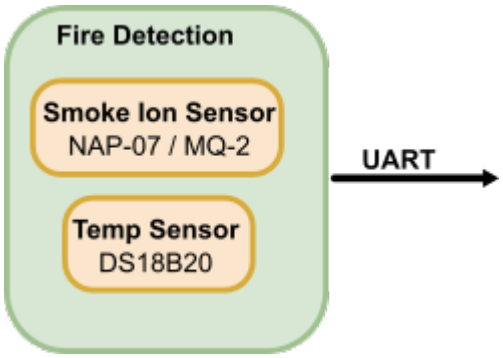


Figure 2: A snapshot of the section of our block diagram that handles sensor detection.

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 connected to 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

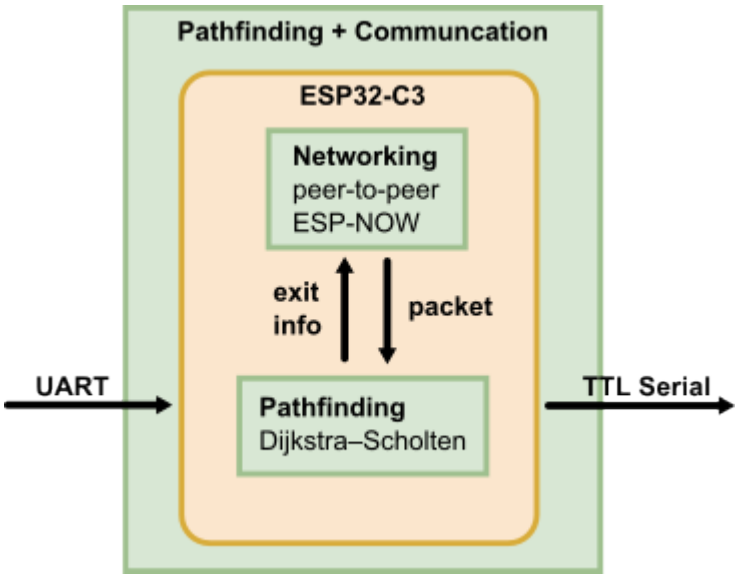


Figure 3: A snapshot of the section of our block diagram that receives sensor data from the current node, and sensor data from other distributed nodes, and with this information, plans the optimal path out.

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.

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 the environment that each node is in; the network communication we are using - ESP-NOW - has a preference for an open floor plan with few obstructions, similar to all wireless communication standards. In an open room, we can expect 200ft of communication without any dropped packets, and about 120ft without dropped packets in a normal building scenario.

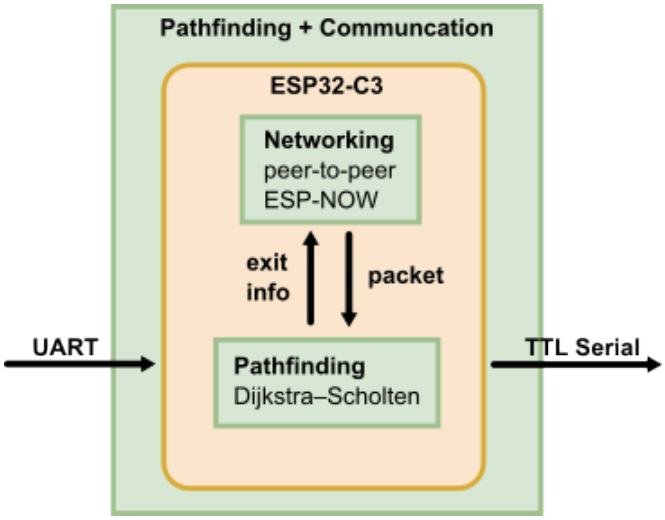


Figure 4: A snapshot of the pathfinding section that is distributed pathfinding and ESP-NOW peer-to-peer networking.

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 detection time, networking, pathplanning, 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

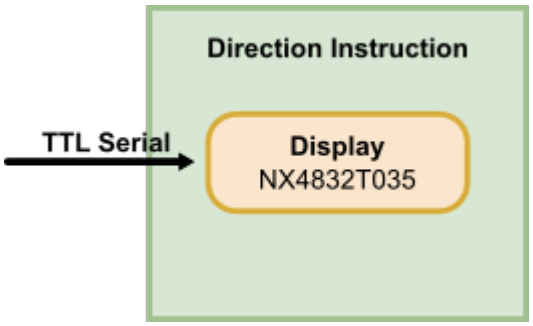


Figure 5: A snapshot of the section of our block diagram that handles directions out of the building using an LCD.

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 LED 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.

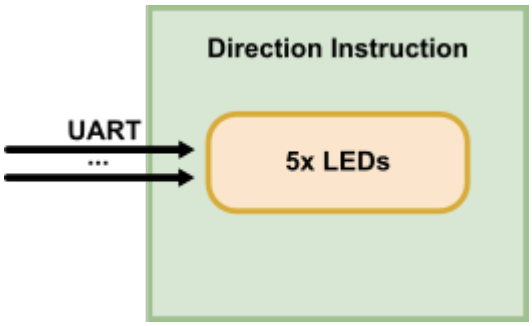


Figure 6: A snapshot of the section that shows the LED variation of Direction instructions.

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:

- detection time + networking time + path planning time + the time to display instructions 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

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.

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

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.

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

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

$$O(V + E) \quad (3)$$

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

$$O(V + \lg V) \quad (4)$$

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: notably, it will allow us to significantly increase the range of our system because nodes only need to communicate with their neighbors and not every node on the system. 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. 1) 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 one of the better networking options 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.

5.4.3 ESP-NOW

Similar to ZigBee, ESP-NOW is also a peer-to-peer network communication protocol, with all of the same benefits that ZigBee has. It even has the added benefit of having a larger range, being built directly into ESP32's and ESP8266's, is lower power consumption because there is no need to power an additional board, and has less tech debt because there is no need to maintain any additional libraries. ESP-NOW has all the benefits of ZigBee, plus a few more, with the only downside being that it is exclusively locked down to Espressif boards.

5.5 PCB Fabrication

In regards to printing our PCBs we considered two possible options that centered around making the PCBs ourselves or sending the design to a professional company who specializes in PCB fabrication. While being pressed for time was huge factor in our decision, we also wanted to learn how to use new tools and valued our interest in discovering how PCBs are made.

5.5.1 PCB Lab in Techspark - Voltera

This option was introduced as a method that would allow us to get a quick turnaround time as we would be learning how to use the machine that drills and prints our circuit boards. However, initially, there was no resource that we could contact to learn how to use the machine but luckily we were able to learn from our lab technician. Throughout the first time making the PCB from scratch, we realized how time consuming the process was in that we had to ensure the Voltera was calibrated properly in terms of alignment to the through-holes but also in regards to ensuring just the right amount of conductive ink was flowing from the probe. The initial PCB took close to two hours to complete. Despite immediately having access to this PCB, we decided it might not have been worth spending this long especially considering that in the end we would need seven total nodes. We decided to prioritize efficiency which is why we ended up ordering our final round of PCBs through JLCPCB. This way we we could send the PCB design off, and work on other components of our project while they were in production.

5.5.2 Shipping through JLCPCB

This method proved to be the most efficient and relatively cheap. We did have to make two orders as the first had issues with our circuit, but the boards that were finalized and working as intended were extremely easy to solder our components onto. With the Voltera in Techspark, when we tried to solder our components, the conductive ink actually ended up coming off despite being cured to the board. We never ran into this issue with the printed boards from JLC which proved to be the best option. While it is unfortunate that once we catch an error we cannot immediately fix it in the design once it is sent to production, it is worth it in the sense that it comes relatively quickly and we can focus our efforts elsewhere until its arrival and be able to

quickly assemble the components onto the board without hassle.

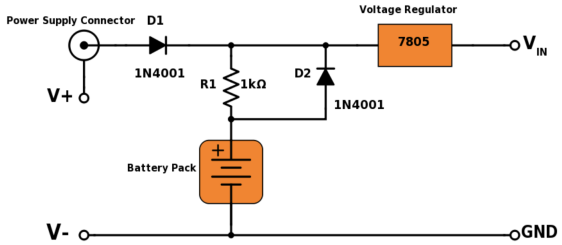
6 SYSTEM IMPLEMENTATION

Our system implementation can be broken into descriptions for the 4 following subsystems: Circuit design, sensor interfaces, communication software, and pathfinding software. We also added a combination of these 4 which is shown in our PCB integration section. section 3 rather than redundant. You can refer back to earlier figures in section 3 using Fig. 2, Fig. 3 and Fig. 5.

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 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 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 15 seconds, our system will switch into 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 execution.

We have settled on using ESP-NOW as our form of communication between the nodes in our system. As explained in the Trade-offs section, this means we don't have to rely on WiFi which has a plethora of downsides: it relies on a router which is a central point of failure, and requires all ESP32s to know the SSID and password of the network, which is troublesome to maintain for sysadmins. It also means we don't have to rely on ZigBee, which would be an added expansion board, and therefore added cost to the price of an individual node. There is little to no downside to using ESP-NOW in the context of this project.

6.4 Pathfinding Software

We have two distinct implementations for pathfinding.

In the first implementation, we are just implementing a standard Dijkstra's implmenataion where all of the nodes will continuously execute their pathfinding algorithm in order to determine the safest route to the exit. 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 path-planning 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 second implementation is a version of distributed pathfinding called Dijkstra-Scholten algorithm. 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(K. Mani).

We see the distributed pathfinding to be more viable in large-scale buildings where nodes are not in range of every other node in the system, as it will allow for near-infinite range as nodes only need to be able to communicate with their neighbors. We see the base Dijkstra's implementation being more useful in small buildings, such as residential houses if the project were to expand into that space of the market.

6.5 Display Output

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.

6.6 PCB Integration

Below we have the final PCB designs of our nodes as well as a photo of one of our complete LED nodes.

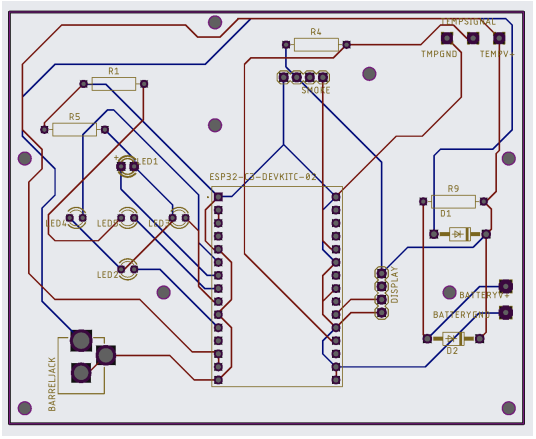


Figure 7: A snapshot of the PCB design for each node.

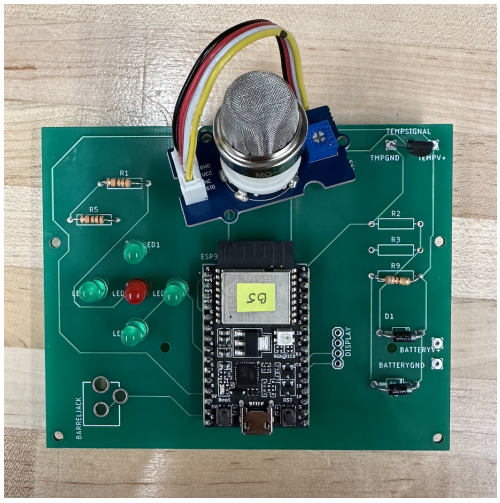


Figure 8: A snapshot of one of our custom PCB boards depicting our LED node.

Here we can see that we have a single design for both types of nodes. This means that for our LCD nodes, we only take advantage of the display and disregard the LED components on the board. Similarly for our LED nodes, we only take advantage of the LED components and disregard the connection to the display. We found that this was a good solution in terms of being able to succinctly order our PCBs. In addition, the display is to be mounted on the four front facing through-holes which would end up covering the LED components, microcontroller, and backup battery circuit, ensuring maximal space for the user to focus on the provided instructions. The PCB integration incorporates every subsystem with the communication and path-finding uploaded to the microcontroller, and the backup battery circuit, temperature and smoke sensors, and power soldered to the board.

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 about 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 Results

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 135º 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. Because of the ease of use of our temperature sensor, this sensor became the trigger that we used to test our entire system as a whole. Through all of the testing that we performed, we have never seen our temperature sensor fail. This means that our temperature sensor has a 100% success rate for all of the testing that we have done. It is difficult to find a benchmark that we were hoping to reach because temperature sensors are not very common in fire detectors, but we feel that this testing shows that our system will react well in a situation that involves a room or hallway heating up due to a fire. We realize that temperature detection is often not enough to detect a fire. To detect a fire with temperature requires a flame that is near the temperature sensor, and the fire may take some time to spread to our detection node. To remedy this, we have installed smoke sensors into our nodes that provide an additional metric to detect fires with. Smoke detecting is a common way of detecting fires, so we wanted to ensure that our version of this method was reliable.

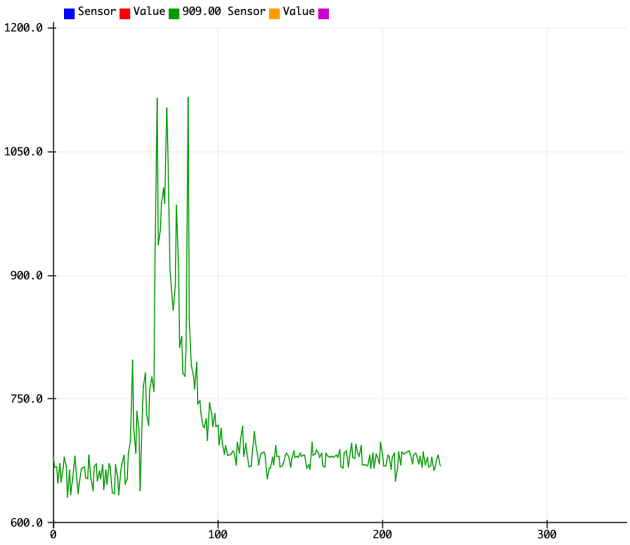


Figure 9: A graph of smoke readings on MQ-02 sensor vs time.

In this graph we can see that at the spike was when the sensor was brought towards the smoke that was coming from a burning candle. Using this graph, we were able to print the values at which the smoke was detected by the sensor and reached a threshold value of 950. Because our smoke sensors have to be calibrated individually, we wanted to ensure that the smoke sensors that we are using are in sync in that they are able to output fire detection at the same thresholds when smoke is present. To test this, we set up 3 of our smoke sensors and tested the readings when exposed to a burning candle. We wanted to ensure that the smoke readings would exceed our threshold for these trials. We tested each of these three sensors three times, and for each trial, the reading exceeded our threshold value and returned back below the threshold once the fire was removed. For this testing, our smoke detection had

a detection accuracy of 100%.

7.2 Path Finding Algorithm Results

Evaluating the correctness of the path-finding algorithm relies primarily on test cases for the algorithm itself. To test this, we ran against known unit-tests, checking that the output of our implementation of the Dijkstra code matched what we expected it to be. After having worked out any memory issues and issues keeping the process alive in an infinite while loop, we had no discrepancies between our output and the output we were expecting from the known unit tests.

Due to having two versions of pathfinding implemented, we also plan to run our code with both the Distributed pathfinding and the base Dijkstra's pathfinding - which we had verified as being correct already - and log any time there was a difference between the optimal safe output. At the time of writing this, the distributed pathfinding Dijkstra-Scholten algorithm implementation is not running without error, and therefore cant be accurately compared to base Dijkstras. That being said, we do expect for there to be a 1-2 log cycle delay on a small graph which we attribute to the latency of packets on the ESP-NOW networks. That being said, we expect to receive the same results between the two pathfinding versions after an appropriate settlement period.

7.3 Communication Between Nodes Results

As mentioned in the design tradeoffs section, we completely switched over to using ESP-NOW, rather than WiFi or ZigBee for a variety of reasons. We tested the communication protocol in a variety of ways, but we were primarily focusing on the distance we could reach between nodes before we began experiencing packet drop. To test this, we had a stationary ESP that was sending arbitrary packets to a mobile ESP that we receiving these packets. We would systematically move the node further from the stationary node, hold for a set of packet sends, and record the number of packets dropped within this time period. After this, we would move back an consistent distance, and repeat the process, noting the number of packet drops at each distance between nodes. After doing so, we came us with the following graph:

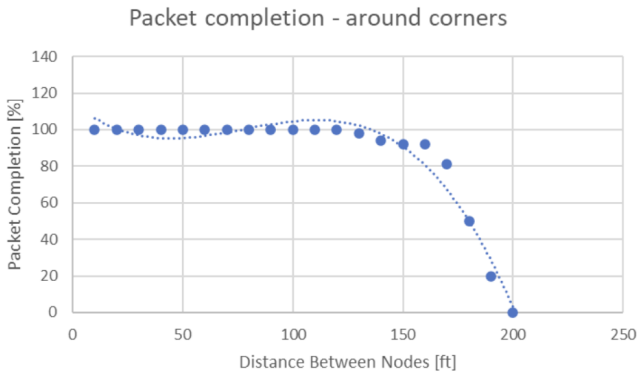


Figure 10: A graph of the packet drop over varying ranges.

We came up with the result that within a dense, office-building like floorplan, we could consistently expect to receive information without any of the packets dropping up until about 120ft. In a more open floorplan, such as a long hallway, we could consistently receive packets without dropping at 200ft, and in an outdoor scenario, we were able to reach around 1000ft.

7.4 Display directions on LEDs Results

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 Results

When testing this subsystem of the project, we were interested in looking into both the readability of the determined safe path, as well as the reliability of the display to accurately show the entire path.

We were originally only using a line to connect nodes on the floorplan, but that was only 1 pixel wide, and therefore there were notable issues actually determining when the

color changed and the path had updated. We made the decision to thicken the line so that the color distinction was more clear, which significantly improved readability in the team's opinion. During the TechSpark demo, once users had the color distinction between different nodes and lines explained to them, they responded positively and seemed to be able to confident about being able to follow the path out of the demo model; this is a qualitative test, but the positive feedback from real users of the system was encouraging to us.

As mentioned, we were also interested in the reliability of the display to output a correct path based on the input path it received. During our testing, we noticed that there were very infrequent instances where the display would miss showing one of the legs of the output path. During all of our testing and the TechSpark demo, we have only viewed this small issue 5 times, in the estimated 2400 display refreshes that we have personally witnessed. This gives us a success rate of 99.86% which we deem to be more than satisfactory, despite not having a hard use-case requirement defined for this subset of the project.

7.6 Power Testing results

After assembling a node on our PCB, we were able to begin testing the amount of current that one node is drawing. From our design requirements, we want to be able to operate our system for 24 hours in sleep mode and 5 minutes in active mode. We made the decision to do our power testing on a node with a display, as this has the most power draw. After testing our system, we can see that, when in active mode, our system draws 420mA. Additionally, our batteries have a capacity of 2000mAh. From this, we can use the following equation to determine the time that our node will be able to run.

$$\frac{2000\text{mAh}}{420\text{mA}} = 4.76 \text{ hours}$$

This meets our active mode requirement. In our deep sleep mode, we were drawing 270mA. Repeating the equation:

$$\frac{2000\text{mAh}}{270\text{mA}} = 7.4 \text{ hours}$$

This metric did not meet our design requirement. After measuring the current going into our smoke sensor, we can see that the smoke sensor is drawing 200mA. This was not accounted for originally in our calculations, because the smoke sensor was decided on later in our design. However, we suggest two remedies for this. First, it would be feasible to increase the battery size.

$$24\text{hours} = \frac{\text{Capacity}}{270\text{mAh}}$$

$$\text{Capacity} = 6480\text{mAh}$$

However, this is quite a large capacity. A better solution may be to incorporate an electric switch that can be turned on and off by the microcontroller to shut the smoke sensor off while the ESP32 is sleeping.

8 PROJECT MANAGEMENT

8.1 Schedule

Refer to the Gantt chart shown in Fig. 14 of our schedule attached at the end of the document. Due to our participation in the Techspark demo, we pushed up a lot of our deadlines to ensure that we could present our project in its best possible state. That being said, we have been on progress to finish our project and ensure that we update our schedule when there are delays. We also ensured to 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 to ensure efficiency.

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. Initially, our teams deadlines were not 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 we got closer to our interim demo, we had to do a lot more integration and most of our meetings involved at least two teammates to ensure that we were combining sub components properly. However, as a team, we all have experience with circuits, hardware, and software so we were able to help each other out in the event that one of us was stuck and were not making progress.

8.3 Bill of Materials and Budget

Refer to Bill Of Materials Table 1. This is our cumulative list of hardware that was necessary for our project. Unfortunately, some 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. Some of our resources came from

the 18-220 lab but the majority we had to order online. In addition, unfortunately, we had to account for the first round of PCBs that we sent out to be fabricated had some trace issues that we resolved in the second round of orders. We plan to use the remaining budget on mounting supplies to ensure that we can connect our smoke sensor, battery holders, and display to the PCB so it is one cumulative node.

8.4 Risk Management

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 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. In the end, we ended up ordering a more reliable, commonly-used smoke sensor with a well documented datasheet to ensure that we were not wasting time making the internal workings of the sensors just to get it to work. 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.

One of our biggest risks throughout our project was the PCB fabrication. Our initial plan was to make our PCBs in the PCB lab in Techspark using the Voltera machine to drill and print. We had to get in contact with Quinn Hagerty, the lab technician, who learned how to use the machine and passed that knowledge onto us. We didn't account for its imprecision and issues due to human error in calibration of the conductive ink. As a result, we had to scrap this idea based on the fact that the conductive ink was smudging and connecting components that were not supposed to have connections. We then switched to ordering our PCBs through JLC where we were able to upload our design and receive them in four days at a relatively low cost. Unfortunately, this first round of PCBs had some issues with the traces and we had to recreate a new design quickly as we were pressed for time. We meticulously made those changes and shipped out a new set of PCBs through the same website. Again, we were worried that they were not going to come in time so we decided to give the Voltera another go and make the PCB ourselves. While the process was a lot smoother, when it came time to soldering the conductive ink was coming off despite being cured to the board. Luckily, the new set of PCBs came earlier than expected and they worked as intended. Throughout this process, we always wanted to ensure that we were making progress and actively seeking out ways to get past road blockers and risks.

In regards to software, we ensured that we always had backup plans in terms of the algorithms that we wanted to use for the pathfinding. While we started with Dijkstra's we wanted to be able to see if we can implement distributed path finding as well. Most recently, it has been our biggest challenge incorporating test cases and debugging. While we continue to work on this, we always wanted to make sure we have a backup plan for being able to ensure that we can provide the occupant with the optimal path.

Throughout our project process we wanted to ensure that we planned out major deliverables such as the proposal and design presentation, design report, ethics assignment, and the final presentation, video, and report. In this way, we were able to account 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 Ethical Issues

With FireEscape, at a high level we are trying to eliminate the risk leading occupants towards hazards when evacuating a burning building. If this project became an actual product or widely adopted, the ideal user/customer would just be any occupant of a building, anyone entering a building, anyone spending time in a building, etc. The most vulnerable to failure would also be the users in the event that the product fails. This is because failure for our project would mean that the occupants are unable to evacuate or are directed towards the fire instead of escaping it which would put harm towards the people in the buildings. As a whole, our product would not provide discrimination or injustice in a global, economical, or environmental matter as it is centered around the safety of the public. The cost

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Ind. Cost	Tax/Shipping	Total
RISC V Developer Board	ESP32-C3	Adafruit	7	$9.95	$21.16	$120.66
Rechargeable Batteries	Amazon Basics	Amazon	28	$1.88	$0.00	$52.64
Temperature Sensors	DS18B20	Adafruit	7	$3.95	$14.36	$42.01
Smoke Sensors	MQ-02	DigiKey	7	$7.60	$16.25	$69.45
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
Battery Holders	36-2465-ND	DigiKey	7	$3.20	$8.76	$31.16
Diodes	1N4004	18-220 Lab	14	$0.00	$0.00	$0.00
PCBs (failed)	–	JLCPCB	10	$0.50	$20.69	$25.69
PCBs	–	JLCPCB	10	$0.50	$20.69	$25.69
220 ohm Resistors	220QBK-ND	18-220 Lab	7	$0.00	$0.00	$0.00
150 ohm Resistors	150QBK-ND	18-220 Lab	7	$0.00	$0.00	$0.00
DC Power Jack	PJ-102AH	PCB Room	7	$0.00	$0.00	$0.00
						$551.90

of the system would be covered by the building makers in which they can implement based on resources and availability.

After an in-depth analysis of different ethical considerations, we have summarized them into key categories below.

10.1 Public health, safety, welfare

Regarding public health, we would have to ensure that our design is up to the standards of the fire code, a bar that we know is properly maintained and addressed. If our nodes already fire detect, then in a way it replaces the need for smoke detectors in buildings but also provides a way to exit the building instead of just evacuation maps on the backs of doors. While our product is more directed towards safety, we are still interconnected with health in that we want to make sure our services are not putting the users at risk. We want to make sure the levels for flames, smoke, and heat match those of current fire code standards to ensure that occupants aren't being exposed to worse conditions than they otherwise would have. Regarding public safety, we believe that our project is the most concerned with this consideration area as our product would aim to prioritize the health and well being of occupants in a building in the event that a fire is present. We would want to provide effective instruction and ensure the safety of our users. We do not want them to escape towards hazards which would put themselves at a greater risk. Especially when fire drills are not taken seriously, we want to ensure that the users can see a viable, safe path and are kept up to date with the optimal evacuation plan to safely depart. Regarding public welfare, our project as a whole would be scalable to any building in which a floor plan can be inputted. In this way the product can be implemented in most buildings for use by the general public. While the product would have to be implemented by the decision of the buildings, it wouldn't't directly impact the incomes, insurance, housing, etc. of the general public for use of these systems. It is a choice made by the building decision makers whether or not to implement our fire escape system or not.

10.2 Accessibility

Regarding accessibility, our system is based entirely on visual cues which does not bode well for visually impaired users. As a result, a possible extension to address this concern is to implement verbal directions in addition to the visual cues to ensure that we are not directing our product favoring certain groups of people over others deepening social discrimination. In a similar vein, while our project leads occupants toward exit routes which most likely involve stairwells and hallways, we need to consider accessibility in the form of people who rely on wheelchairs. Another possible extension to address this concern is to have a button on the node to indicate wheelchair usage which would cause the pathfinding algorithm to favor larger hallways and ramps over narrow hallways and staircases. Finally, we also considered the possibility of colorblindness in occupants as our LED nodes utilize a red center LED for positioning and green directional LEDs as well as the path on the display highlighting the optimal route to take in green, while leaving the original path in its input color (most likely black). In this situation, a possible extension would be to have the middle LED blink so that it can be distinguishable from the four other relational LEDs so a user can determine the right direction as well as an animated path out or dashed lines for the optimal route when it is displayed on the LCD node.

10.3 Malicious Intent

Regarding malicious intent, we believe that the worst-case scenario for our product would be if someone were to maliciously manipulate the signals being sent by our nodes, with the intent of harming or confusing the users of our system. We are HMAC'ing all of the packets being sent so that no one can imitate our data; our data can easily be read,

but attackers could not send their own data to our network without knowing our secret key. For example, if a node has gone down entirely, or is reporting a fire nearby, if we weren't HMAC'ing our packets, someone could send a fake signal to the remaining nodes stating that all is clear and safe. This would result in the pathfinding algorithm directing people toward this hazard, rather than around or away from it. This, however, is not a possibility within our current system, as we disregard all packets that fail the HMAC check immediately upon receipt. Someone could, however, completely shut off the system which would leave the occupants vulnerable to a fire if they were to jam the frequency that ESP-NOW uses to communicate. Our system is relying whole-heartedly on the validity of the data that each node is receiving is true and hasn't been tampered with; therefore, it is essential that we get HMAC'ing correct, as our system would not work in the event of a malicious attacker. Getting this wrong would mean we lose the ability to create a safe path out of the building. The people who get harmed in this situation would be the users of our system, who trusted us to get the pathfinding right and direct them out of a dangerous situation, and not towards one. This issue could cause harm and even death to the occupants of the building. We think anyone with some sort of impairment to their movement would be particularly susceptible to harm in this scenario – a wheelchair or crutch user, for example. We say this because it would take this group of people slightly longer to reach a potential hazard if they were directed toward one, and therefore would have less time to find another way out of the building upon realizing they were led in the wrong direction. As a result of all the possible issues listed above, that is why the HMAC'ing and crypto part of our code is essential to our project. In the situation with a hacker, occupants who are new to the building may be most at risk. This is because they would not be as familiar with the floor plan and may be less able to make it out of the building safely without directions. We realize that for our project, we are accountable for the reliability of our software, regardless of tampering, and we are equally responsible for the path output being optimal and correct.

11 SUMMARY

With FireEscape have created an efficient and safe way to help occupants evacuate a building in the event of a fire. With our distributed node system, we 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. We have successfully met all of our use case and design requirements that we introduced in our design.

11.1 Future Work

Ideally, we would be able to implement FireEscape in large scale buildings where there would naturally be multiple paths to exit. This would mean introducing more nodes than we created for our demo and scaling upwards to place them in key areas like long hallways, corners, junctions, stairwells, etc. We have thought in depth about how the number of nodes would scale up for our report as demonstrated earlier, but it would be exciting to see this in action. During our Techspark demo, we received a lot of comments about what to do when there are fires detected at every single node and there is no safe path out. After our debate of what algorithm to use between Djikstra's and A*, we decided that a "less hot" path towards a hazard is still unsafe and it would be best to wait for medical and fire professionals. If implemented in the future, we envision a button that would be able to immediately get you in contact with a qualified professional to alert them of your presence in the burning building and receive immediate assistance and advice. Overall, we could brainstorm multiple additional extensions to our project but we are proud of how far we have come.

11.2 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.

11.3 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. We also all learned how to use new tools especially in regards to the Voltera machine where we were able to print versions of our own PCBs from scratch using drill bits for the through-holes and conductive ink for the traces. Special thank you to Quinn for spending a lot of time and effort helping us out!

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,notification %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/Display/pacode?file =%2Fsecure%2Fpacode%2Fdata%2F034%2Fchapter50%2Fs50.53.htmlamp;d=reduce.

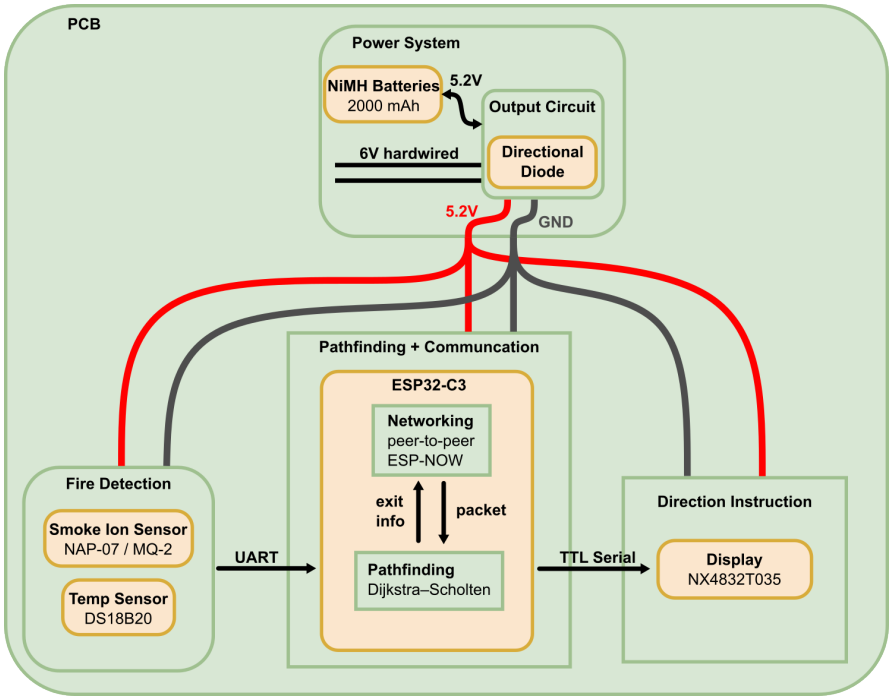


Figure 11: A full-page version of the same system block diagram as depicted earlier.

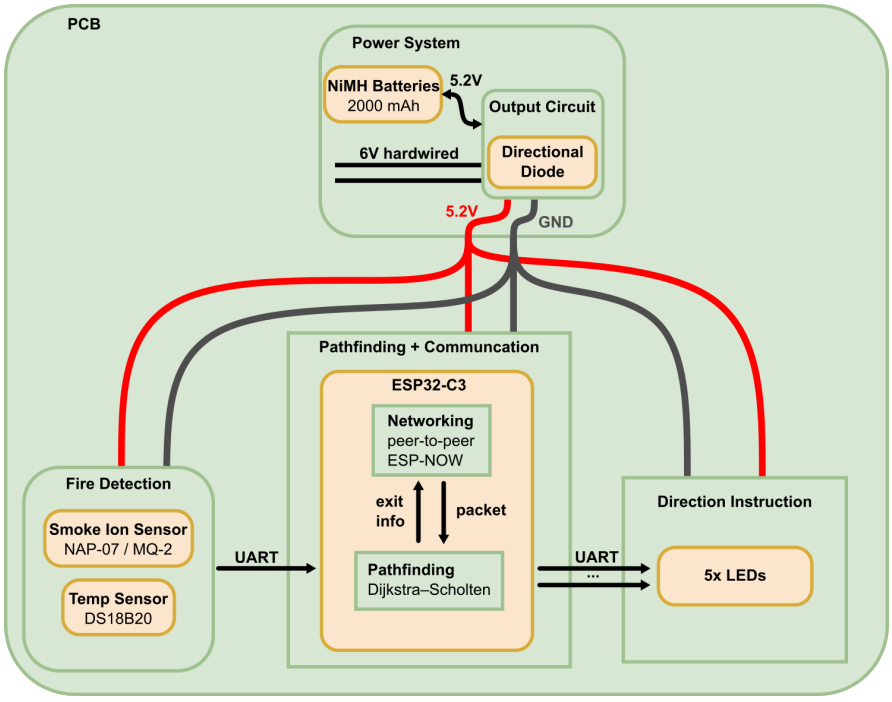


Figure 12: A full-page version LED variant of the block diagram.

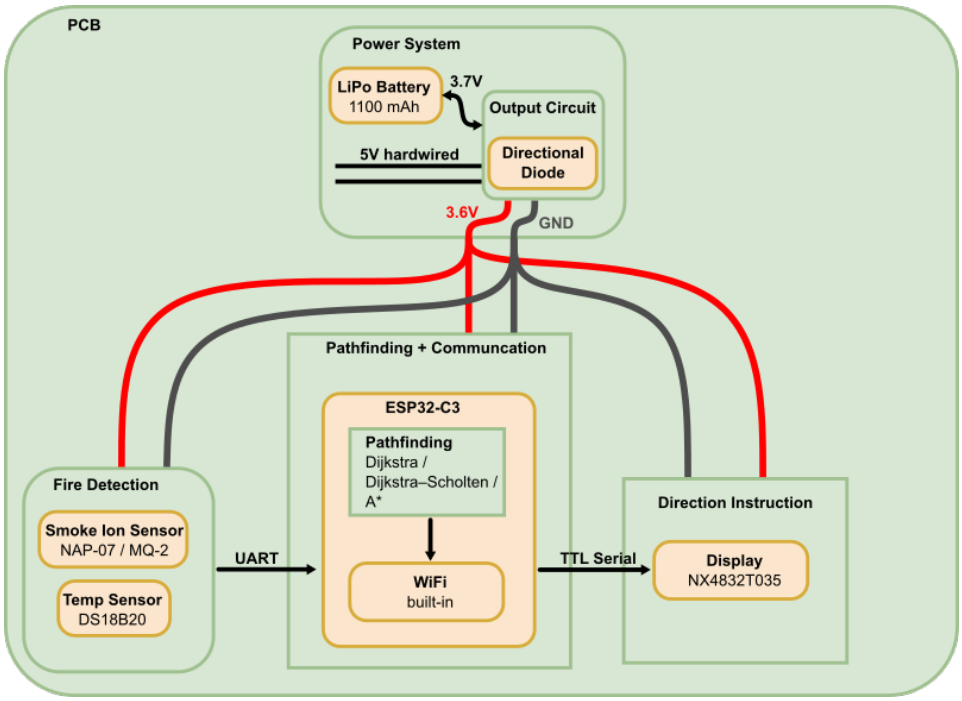


Figure 13: A full-page version WiFi variant of the block diagram.

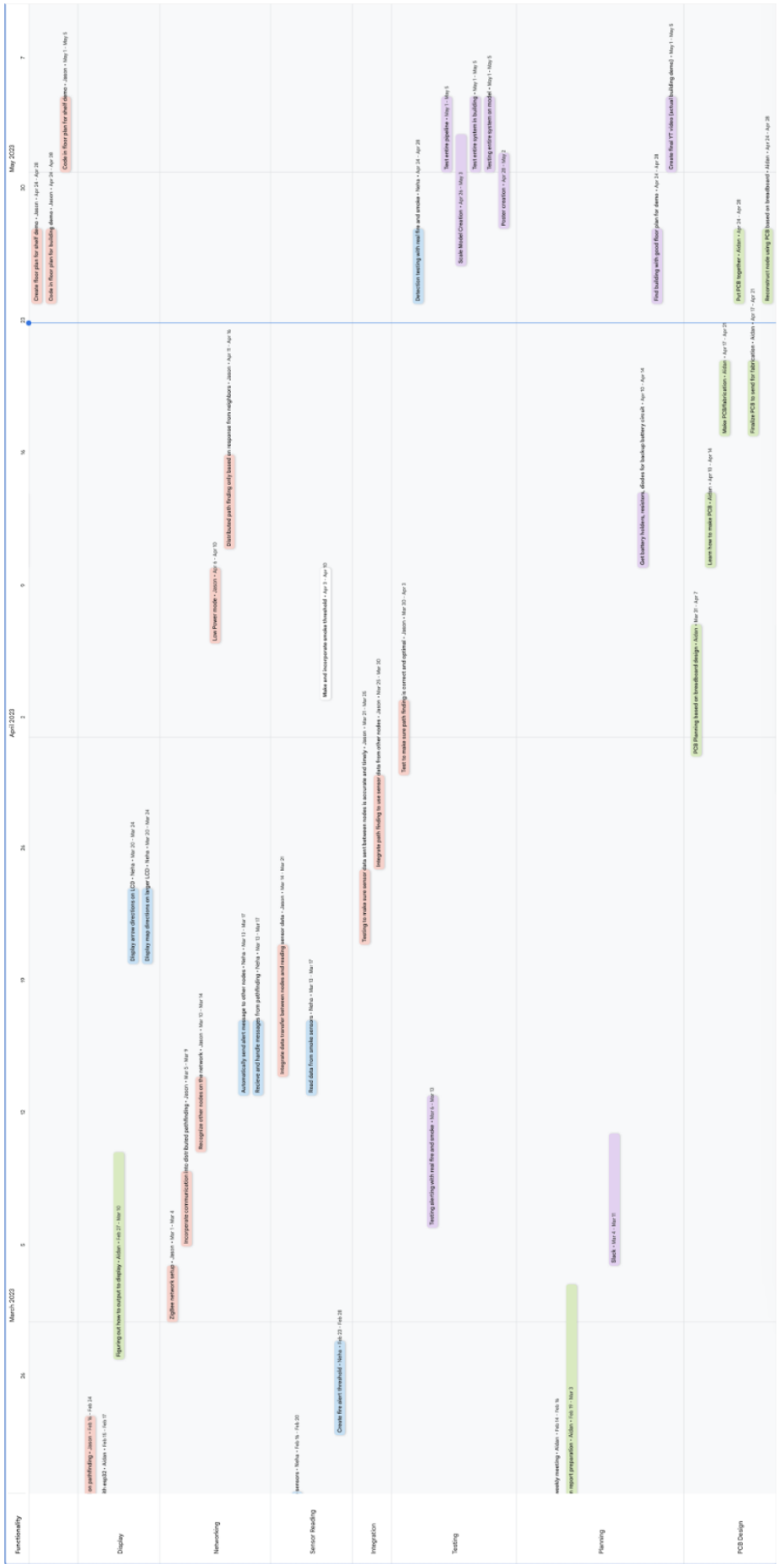


Figure 14: Gantt Chart