Cicada

Design Document

Team 34

Cooper Hoy

Daji Qiao and Sarath Babu

Cooper Hoy, Marcus Yeung, Hrushi Bhatt, Sean Fjellman, Caleb Kugel, Jake Ebersole, Jack Rice

sdmay27-34@iastate.edu

https://sdmay27-34.sd.ece.iastate.edu/

Revised: V1 - 10/13/2026
Revised: V2 - 11/10/2026

Executive Summary

Field fires are a real and costly threat to farmers. A fire that starts in dry crop residue, a hot combine, or a stray spark can spread across acres before anyone notices, especially on large rural properties where no one is watching every field and cell coverage is spotty. By the time smoke is visible from the road, crops, equipment, and sometimes buildings are already at risk. Our project, Cicada, aims to catch these fires early and alert farmers while there is still time to respond.

Our key design requirements came from that problem. The system has to work in remote fields without Wi-Fi or reliable cellular service. Sensor nodes need to run on their own for long periods on solar and battery power. Detection has to be reliable enough that farmers trust the alerts, and the data has to reach a place where farmers can actually see it. The system also needs to be affordable and simple to set up, since it is meant for working farms rather than research sites.

To meet these requirements, we are building a network of low-power sensor nodes based on STM32 microcontrollers. Each node carries sensors that watch for signs of fire, starting with smoke and particulate detection, and later adding soil moisture and other environmental readings that help show when conditions are risky. Nodes communicate over LoRa, a long-range, low-power radio technology well suited to open farmland. Data travels to a LoRa gateway placed in the field, which forwards it to a remote server. From there, a dashboard displays sensor readings and alerts. The node firmware is being developed in Zephyr, a real-time operating system built for embedded devices.

We have made solid progress this semester. Nodes are successfully sending data to our base station, and we tested a node-to-node link across campus with low packet loss, which gives us confidence in LoRa's range. The gateway is forwarding data to our remote server, and a separate part of our team is building the dashboard that will display it. We have also ordered additional nodes to expand the network.

So far, our testing shows the core communication path works the way we need it to: data can travel long distances on low power and reach a farmer-facing interface. These results address the most basic user need, which is getting information out of a remote field reliably. We still need to validate detection accuracy and long-term power performance before we can say the full system meets every requirement.

Next, we plan to finish the node firmware, tune our fire detection thresholds, and test solar and battery performance. We will then expand from a simple link to a multi-node mesh, add more sensor types, and deploy the gateway in a real field to test range and reliability under actual farm conditions.

Learning Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the engineering standards that apply to this project that were considered.

Summary of Requirements

List all requirements as bullet points in brief.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- CPRE 2880: Embedded Systems I provided the embedded skills to work with our microcontrollers, learning how to read datasheets, how to program microcontrollers and their GPIO pins, integrating sensors to record various real-world data.
- CPRE 4370: Intro to Wireless Security offered experience with ARAnet, the rural wide area network on the Iowa State campus and expanding to other areas of the Ames/Boone area. Learned about networking, wireless metrics, wireless security, and how cellular networks work.
- COMS 3090: Software Development Practices combines technical knowledge with teamwork skills in a manner reminiscent of 4190. The pure software techniques of RESTful APIs and CI/CD pipelines will be incorporated into our dashboard design.

New Skills/Knowledge Acquired

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

Learning Zephyr, a real-time operating system.

Virtual Machine configuration.

Table of Contents

Development Standards & Practices Used ... 2
Summary of Requirements ... 2
Applicable Courses from Iowa State University Curriculum ... 2
New Skills/Knowledge Acquired ... 2
Table of Contents ... 3
List of figures/tables/symbols/definitions ... 4
1 Introduction ... 4
1.1 Problem Statement ... 4
1.2 Intended Users ... 5
1.3 User Needs ... 5
2 Requirements, Constraints, and Standards ... 5
2.1 Requirements & Constraints ... 5
2.2 Engineering Standards ... 6
3 Project Plan ... 8
3.1 Project Management and Tracking ... 8
3.2 Task Decomposition ... 9
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria ... 10
3.4 Project Timeline and Schedule ... 11
3.5 Risks and Risk Management/Mitigation ... 12
3.6 Personnel Effort Requirements ... 14
3.7 Other Resource Requirements ... 16
4 Design ... 17
4.1 Design Context ... 17
4.1.1 Broader Context ... 17
4.1.2 Prior Work/Solutions ... 18
4.1.3 Technical Complexity ... 18
4.2 Design Exploration ... 19
4.2.1 Design Decisions ... 19
4.2.2 Ideation ... 20
4.2.3 Decision-Making and Trade-Offs ... 21
4.3 Proposed Design ... 21
4.3.1 Overview ... 21
4.3.2 Detailed Design ... 22
4.3.3 Functionality ... 25
4.3.4 Areas of Concern and Development ... 26
4.4 Technology Considerations ... 26
4.5 Design Analysis ... 27
5 Testing ... 28
5.1 Unit Testing ... 28

5.2 Interface Testing..... 29
5.3 Integration Testing.....29
5.4 System Testing..... 29
5.5 Regression Testing..... 29
5.6 Acceptance Testing.....29
5.7 Security Testing (if applicable)..... 29
5.8 User Testing..... 29
5.9 Testing Results.....29
6 Implementation.....30
7 Ethics and Professional Responsibility.....30
7.1 Areas of Professional Responsibility/Codes of Ethics.....30
7.2 Four Principles..... 30
7.3 Virtues..... 31
8 Closing Material.....31
8.1 Conclusion.....31
8.2 References..... 31
8.3 Appendices..... 32
9 Team.....32
9.1 Team Members..... 32
9.2 Required Skill Sets for Your Project.....32
9.3 Skill Sets Covered by the Team..... 32
9.4 Project Management Style Adopted by the team.....32
9.5 Initial Project Management Roles..... 32
9.6 Team Contract.....32

List of figures/tables/symbols/definitions

1 Introduction

1.1 Problem Statement

The problem we are trying to solve is that most fires in a farm field go unnoticed until it is too late and heavy damage is sustained. We are trying to solve this problem by creating a network of nodes with sensors to detect field fires quickly and notify first responders to prevent as much damage as possible. Farmers have large amounts of land they care for and they can't always be there to monitor their fields if something happens, like a fire. These fires can start sporadically and it is often too late once the farmer realizes there is a fire. The farm fields are almost always not covered by Wi-Fi or a cellular network.

Our goal is to create nodes that farmers can deploy in their fields that will talk to each other using a mesh network and LoRa radio frequencies. LoRa is long-range networking and will provide connectivity between the nodes so they can transmit data to each other, eliminating the need to have very expensive broad-coverage cellular. The nodes will contain sensors used to detect signs of fire, such as smoke, temperature, humidity, soil moisture, and more. The nodes will be independent of each other so they can deploy far apart from each other, allowing the farmer to place them in the areas of his choice. If the nodes do detect a fire, an alarm will be raised and notify the farmer and first responders where the fire is, and provide them coordinates or a pinpoint using Google Maps.

1.2 Intended Users

There are multiple users for Cicada, as there are multiple people involved in a fire rescue situation. The primary user is the farmer; they are the person responsible for protecting the farm field. They will want their fields monitored and may even help place the nodes, repair them, or monitor their status. The secondary users are the first responders, firefighters. The tertiary users are public works employees, utility workers, and conservation or emergency managers who monitor and protect a large region such as a county.

1.3 User Needs

A farmer needs an easy to access system to check on his farm fields. Even if there isn't an active alert he needs to be able to check the app and monitor current conditions. They may be interacting with the device physically as well, during maintenance tasks such as replacing batteries, so the Cicada node must be easily accessible.

Firefighters main concern is quick, accurate information. When a fire alert occurs, they need to be notified as quickly as possible and be given exact coordinates of where the fire is located. Longitude and latitude coordinates can easily interface with any emergency system or even common systems like Google Maps.

Public works or emergency managers who monitor large areas need an easy way to monitor these systems. They need a clear dashboard view that will show current status as well as any alerts that occur.

2 Requirements, Constraints, and Standards

2.1 Requirements & Constraints

Functional Requirements: Cicada will observe sensor data and when certain thresholds are met, relay this data to the various users of Cicada. The fire potential is assumed based on observed sensor data such as smoke, dry soil, low humidity, etc. Each node must be able to detect wildfires near them such that the total coverage of an area bounded by nodes would exceed 50% of the surface area of the bounded region. Each user has different needs, so there will be different levels of access for the software platform. The monitoring dashboard must be clear so data can be easily

understood and signs of fire do not get missed. First responders should have quick notification and be provided with the coordinates of the potential fire. Engineers on the Cicada team should be able to view the sensor data and wireless metrics at ease, from the dashboard.

Environmental Requirements: Our device needs to be able to operate at all times of year, so it must be resistant to rain, snow, ice, and wind. If there is standing water due to flooding, the pole or tripod the Cicada node is mounted on should be able to withstand the flooding without intervention. We are building a custom housing, so it won't be possible for us to get a true IP waterproofing standard like a box off-the-shelf, but the housing should roughly represent IP65 ingress-protection. This means it will have minimal dust entry and be able to hold up against jets of water (not submersion). The nodes should be safe for the environment. If there is a device failure, it must not expose hazardous material into the environment. If an animal comes up to it thinking it's a feeder, it should have potential danger areas, such as the wiring, tucked away so that it doesn't harm the animal if they interact with it.

Resource Requirements: Each node should not exceed the cost that lets us build the full mesh within our $500 project budget, plus any extra funding we receive from our advisors and the department. To stay within that, we will use low cost, off the shelf parts that are easy to buy, replace and repair. We will also reuse hardware we already have, like our STM32 boards, LoRa shields and base station, wherever we can.

Power and Infrastructure Requirements: Each Cicada node must work independent of any existing power or cellular infrastructure. Most farm fields do not have sufficient cellular coverage, so the Cicada nodes must connect in an ad-hoc mesh configuration. They do eventually need to connect to a cellular conduit to transmit the data to the Iowa State network (we have access to ARAnet, a rural cellular network provided by Iowa State). Each node must be individually powered and not require constant charging, so they will need a solar panel and a battery to power the Cicada node.

Aesthetic and Size Requirements: When a person looks at the device, it should be apparent that this is a monitoring system and part of safety equipment. It should be highly visible to farmers so that it doesn't get run over or damaged, so high-visibility tape or something similar should be used. It must be close enough to the ground so it can measure the soil moisture, but at the same time, be tall enough to have line-of-sight to ensure LoRa networking functionality. About the size of a person should be sufficient. It must be placed in an area where it doesn't interfere with power lines or existing infrastructure.

2.2 Engineering Standards

Clearly established standards have many benefits for the engineering design process. Standards allow for consistent protocols so that different teams can optimize for safety and usability without needing to start from scratch on each new project. Instead, the explicitly stated standards from IEEE allow a degree of trust that products are engineered with quality in mind. These standards are very applicable to our project which is dealing with wildfires, which is a very critical phenomenon. Our users need to be able to rely on our technology providing accurate and timely data, because

emergency services could feasibly be making high-impact decisions based on the information our network provides.

The Cicada project will adhere to many specific IEEE standards, including:

1. IEEE 2700-2014; Standard for Sensor Performance Parameter Definitions
   This standard dictates how sensor data is to be calibrated, regulated, and interpreted. The paper notes how sensors are becoming increasingly more frequent and the world becomes more mechanized, but the process for testing sensor quality standards is being developed much more slowly. This standard analyzes many of the most common types of sensors and sets protocols on how the data is to be interpreted.
2. IEEE Std 802.11s-2011; Amendment 10: Mesh Networking
   This paper is an amendment to wireless communications standards specifically focusing on mesh networking. It intends to standardize the terminology related to mesh processes, which is relatively niche compared to other more common networking strategies. It is necessary to have standard networking protocols so that messages can be interpreted correctly.
3. IEEE 1063-1987; Standard for Software User Documentation
   Our third highlighted standard is software focused. Specifically, this document provides a structured outline for creating user documentation so that people who were not involved in programming a feature can thoroughly understand its capabilities. This standard aims to increase the quantity and quality of information about a software application so that users have all of the information they need to use it safely and effectively.

Each of these standards has a tangible impact on the project:

1. The *Standard for Sensor Performance Parameter Definitions* is very applicable to our project. The entire premise of early wildfire detection is dependent on receiving accurate sensor data. Each node will have a variety of sensor types, potentially including temperature, humidity, VOC, and soil quality sensors. Of these, the temperature and humidity sensors have individual sections in the standard explaining how these sensors are most effectively used in a real world project.
2. *Amendment 10: Mesh Networking* is clearly relevant to Cicada. The networking protocol is a major piece of our early design process. The nodes need to have a clearly established method of sending out data packets, so that neighboring nodes can confidently decode the information. Any loss of information could be critical in case of a wildfire, so the mesh network needs to be well-established. This document provides standards and diagrams for typical mesh networking structures, as seen below.

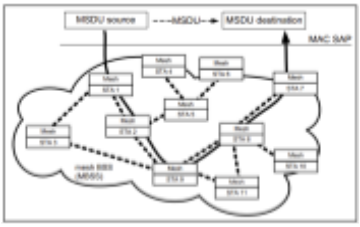


Figure 1: Mesh Network Overview

3. *Standard for Software User Documentation* is a very practical tool when it comes time to create documentation for our project. While a large amount of the project time will be spent testing our node sensors and the mesh communication, there are key components to the project which are entirely software-based and will need clear documentation. The React frontend will need an explanation for users on how to most effectively make use of the geographic interface we intend to build with a satellite imagery API.

We will emphasize reliability in our design. Ideally once the nodes are positioned out in the field they will last for a long lifetime with minimal downtime. This requires dependable sensors to read measurements which accurately reflect their local conditions. These sensor data are sent in packets through our LoRa mesh network. This network will follow the protocols outlined by the IEEE so that minimal package loss is achieved. We will test our sensors thoroughly before deploying them and advertising our dashboard data as legitimate. The nature of our disaster-detection project requires high accuracy and as few false positives as possible. Specifically, our team selected IEEE 1451 for sensor data sharing, IEEE 802.15.4w for low power, long range wireless networks, and IEEE 1625 for lithium-ion battery safety. Together, these standards help our nodes collect accurate data, send it reliably, and stay safe in the field without becoming a fire risk themselves.

3 Project Plan

3.1 Project Management and Tracking

We are using an Agile development framework. This ensures that we make steady progress every week, but also circle back to check progress and ensure we are still meeting our goals. This style project benefits from Agile because as we further develop Cicada, more ideas spring up that we want to incorporate or change. For example, we realized the node would need independent power and the power would drain quickly without being recharged often. So we added a solar panel to each node. Since we are using an Agile framework, we were able to touch base and meet as a team to discuss this with the advisors and get feedback, and then add it to our design. This project will have many more instances like that, so Agile helps us stay modular throughout the design process.

We are using Git issue boards to track progress for anything software related - the Dashboard, VM, and LoRa network code. For documentation, we are using our weekly reports, design documents, and the slideshow that we create weekly for our advisor meetings. All documents like this are stored in a shared Google Drive folder. All communication is handled via Discord, with the exception of email being used for advisor communication.

3.2 Task Decomposition

We divided Cicada into seven main tasks that line up with our subteams. Each task is split into smaller subtasks that can be finished within one or two sprints. Table 1 below lists the tasks, their subtasks, and the dependencies between.

Table 1: Core Tasks

Task	Subtasks	Depends on
T1: Research and Requirements	Define user needs and requirements, research existing solutions, select the microcontroller, radio, and sensors	None
T2: LoRa Mesh Network	Point to point link, range and packet loss testing, packet structure with ACK, HELLO, DATA, ALERT, and ROUTE packets, neighbor tables and relay, self healing routing	T1
T3: Sensor Integration and Power	Zephyr firmware and unit tests, SEN66 and NEO 6M drivers, power profiling and polling rate, battery and charging circuit, sensor calibration	T1
T4: Node Enclosure and Solar	3D printed weatherproof housing, ventilated sensor section, pole mount, fixed solar panel mount	T3 (Component List)
T5: Gateway and Backend	Conduit to VM data path, Spring Boot API, PostgreSQL database, automated build and deploy pipeline	T2 (Packet Structure)
T6: Dashboard	React frontend, Google Maps node view, live readings, fire alerts with coordinates	T5
T7: Integration and Field Testing	Three node prototype, scale to about ten nodes, outdoor field testing, acceptance and user testing	T2 - T6

The main dependency is the packet structure. It defines what the firmware sends, what the gateway forwards, and what the database stores, so T3, T5, and T6 all build on it. The sensor firmware was written and unit tested before the hardware arrived, but final validation depends on the SEN66 and NEO 6M parts we ordered. The enclosure is sized around the final component list from T3, and the integrated prototype in T7 needs every other task working.

Following our sprints, the requirements are completed in this order.

1. Early fall: Requirements, hardware selection, a working point to point link, and the gateway to server data path.
2. Late fall: Final sensors, power design, enclosure, and a working three node prototype.
3. Early spring: Calibrated sensors, measured power, a mesh network, and dashboard alerts.

4. Late spring: About ten nodes in a real field, acceptance testing, and final delivery.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Each task has a milestone with a target we can measure. We review progress in our weekly advisor meetings and track tasks on our GitLab issue boards. Table 2 lists the milestones. Targets start simple and are tightened in later sprints once we have measured data.

Table 2: Milestones and Evaluation Criteria

Task	Milestone	Metric and Target	Target Date
T2	Point to point link	Packet delivery ratio of at least 90 percent at 0.2 miles on campus. Met at 94.3 percent	Oct 2026
T3	Polling rate selected	Battery life modeled for each polling rate. Selected 15 minute polling with a 3000 mAh battery	Oct 2026
T5	Data path	Uplinks pulled from the Conduit to the VM and stored in PostgreSQL	Nov 2026
T3	Sensor bring up	All three nodes return SEN66 and GPS readings, with 96 of 96 readings valid over a 24 hour test	Dec 2026
T2	Three node relay	Packets relayed through a middle node to the gateway with at least 90 percent delivery	Dec 2026
T7	Three node prototype	Three nodes report to the dashboard map for 24 hours with at least 95 percent of readings received	Dec 2026
T3	Measured power	Measured current within 20 percent of our power model	Feb 2027
T4	Enclosure	No water inside after a hose spray test, and sensor readings within 10 percent of an open air sensor	Feb 2027
T3	Detection	Controlled smoke detected in at least 8 of 10 trials, refined to 9 of 10, with less than one false alarm per node per week	Mar 2027
T2	Self healing mesh	With one node turned off, the other nodes reroute and still reach the gateway within two polling cycles	Mar 2027

Table 2: Milestones and Evaluation Criteria			
T6	Alerts	Alert with GPS coordinates shown to the farmer within 60 seconds of an alarm packet	Mar 2027
T4	Solar	Battery charge holds steady or rises over 7 days outdoors with the fixed solar panel	Apr 2027
T7	Field deployment	About ten nodes run for 7 days in a field, covering more than 50 percent of the bounded area	Apr 2027

3.4 Project Timeline and Schedule

We work in two week sprints with weekly advisor meetings across the fall 2026 and spring 2027 semesters. Figure 2 shows the Gantt chart for the tasks in Table 1, with deliverables marked as diamonds. Table 3 summarizes the schedule by phase.

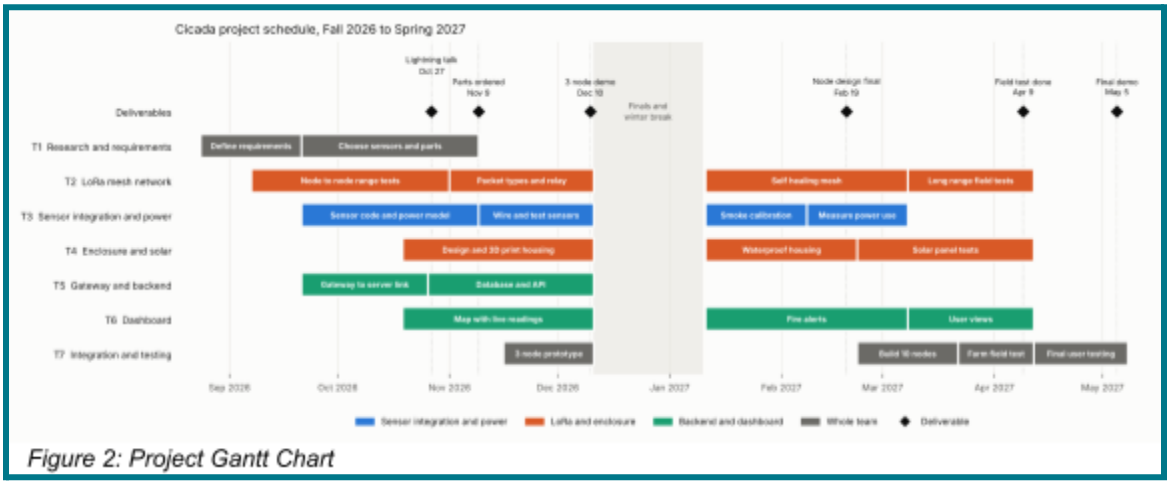


Figure 2: Project Gantt Chart

Table 3: Project Schedule by Phase:			
Phase	**Dates**	**Focus**	**Deliverable**
1. Research	Aug 24 - Sep 20	Scoping with advisors, hardware setup, first node to node link	Requirements and sensor shortlist
2. Subsystem	Sep 21 - Oct 31	Zephyr firmware, packet	Campus range test

Table 3: Project Schedule by Phase:			
Development		structure, power profiling, VM and database, range tests	and lightning talk
3. Design Finalization	Nov 1 - Nov 29	Final sensors, power circuit, enclosure prints, neighbor tables, dashboard map, parts order	Design Document V2 and parts ordered
4. Three node prototype	Nov 30 - Dec 18	Hardware bring up, three node relay, readings on the dashboard	Three node prototype demo
5. Validation	Jan 11 - Feb 21	Sensor calibration, power measurement, self healing mesh, alerts, weatherproofing	Validated node design
6. Scale and field test	Feb 22 - Apr 11	Build out to about ten nodes, outdoor field and solar testing	Field test results
7. Final Delivery	Apr 12 - May 7	Full deployment, acceptance and user testing, final documentation	Final demo

The fall semester builds and tests each subsystem on its own and ends with a working three node prototype, which matches the scope our advisors recommended for this semester. The spring semester turns that prototype into a field ready system with about 10 nodes. The main schedule risk is part lead time. Our ETG order has about a two week lead time, so we wrote and tested the sensor firmware ahead of time to start hardware bring up as soon as the parts arrive.

3.5 Risks and Risk Management/Mitigation

This project is multifaceted in the sense that many separate components will need to interface cohesively for the design to work. Thus, there are risks associated with any piece of the system breaking. Some pieces are more integrated than others, or are more difficult to fix in case of an error. Below is a list of the tasks and the estimation (on a scale of 0 to 1) on the corresponding risk. Our time will be heavily focused on the riskiest, most critical components of the system.

T1 Research and Requirements

- Compatibility - 0.5 Medium
- Oversight of research - 0.2 Low

T2 LoRa Mesh Network

- Connection / Packet loss - 0.4 Medium
- Weather / Terrain - 0.4 Medium

- Self healing failure - 0.7 High

T3 Sensor Integration and Power

- Thorough testing - 0.4 Medium
- Node Power - 0.6 Medium
- Sensor Calibration 0.2 Low

T4 Node Enclosure and Solar

- Weather proofing - 0.7 High
- Ventilated Sensors - 0.4 Medium
- Stability - 0.4 Medium

T5 Gateway and Backend

- Path to VM from Conduit - 0.4 Medium
- Database Management - 0.1 Low

T6 Dashboard

- Updating Live readings - 0.6 Medium
- User experience - 0.2 Low

T7 Integration and Field Testing

- Scalability - 0.8 High
- Field testing - 0.4 Medium
- User Needs - 0.6 Medium

Risk Mitigation Plan

Self-Healing Failure: This feature is in place in case one of the nodes fails; however, we must also account for the possibility that our core algorithm fails. The goal is to make it as reliable as possible, so it is not a worry. One way to do this is to conduct failure testing, where we intentionally drop a node from the mesh. If we notice it is unreliable, we can attempt other algorithms. This is high risk because it is a core part of our project with many variables.

Node Power: One of the most detailed aspects of the project, power management covers not only how our node runs but also how long it lasts. Many variables are involved. However, we may get a working model; it might not be optimal. We can experiment with things such as battery size, Solar panel, Using Idling, and power testing to figure out what will work best to reach our goals.

Weatherproofing: No matter how you design a product, preparing for all types of conditions is daunting. However, our plan keeps these conditions in mind; it may take a few attempts to ensure we can handle what is thrown at the node while still getting reliable data. The biggest issues are battery overheating/freezing, the Solar Panel acting like a wing in the wind, and exposed sensors. For the battery, we can make the walls thicker, use a protective filament, and add a battery cover. Solar panels can take inspiration from solar farms to lock the panel into place. Lastly, we will test the optimal amount of open-air design to protect the sensor.

Updating Live Readings: This is a key part of our project where everything comes together. We need to make sure the readings are not only as fast and accurate as possible, but also displayed in a way that is easy for the user to understand. Because there are so many steps between what the

nodes detect and what the user sees, delays can occur, or readings may fail to appear. We can continue to send heartbeat-type requests to ensure we get not only live but accurate data.

Scalability: Our project has many components and many dependent variables that can be hard to see. Because of this, we need to ensure we have a strong understanding of our objectives. So, as we scale, troubleshooting will not become much of an issue. It's not about preventing problems; it's about making the ones that appear easier to manage by having the right preparation and direction for the project.

User Needs: The absolute most important part of any project is the "why". As we develop our project, we must keep in mind who it is for. The reason the risk is high with this one is that, even if you understand the situation, refining the needs can be challenging; for example, making sure we have a user experience that is easy to follow. Our job is to take all this data and package it in a way that is easy to understand. One way we can do this is by comparing what similar websites do and not reinventing the wheel. Another option is to have farmers test our GUI in the field to learn what they like / not. We need to do this at each step, not just at the end of the project. This helps ensure the user stays at the center of development throughout.

3.6 Personnel Effort Requirements

Table 4: Fall 2026 Task Hours Estimate

T1	Research and Requirements	320
T2	LoRa Mesh Network	121
T3	Sensor Integration and Power	158
T4	Node Enclosure and Solar	96
T5	Gateway and Backend	106
T6	Dashboard	102
T7	Integration and Field Testing	229
Total		1,132 (161 hours a person, 10 per week)

3.7 Other Resource Requirements

Table 5: Estimated Cost per Node

Name	Quantity	Total Price	Purpose
Sensirion SEN66	1	$62.03	Sensor for detection
NEO-6M	1	$5.49	GPS sensor for location
Samsung INR18650-35E	1	$9.99	Battery per node (Outputs 4.2V at full charge and ~3.0V when almost empty)
TP4056 charger with protection	3	$6.59	Charges the battery and cuts it off if it's over-charged, over-drained or shorted
3.3 V buck-boost converter (TPS63020)	5	$13.99	Ensures the SEN66 receives the correct voltage range from the battery. SEN66 needs roughly 3.15V - 3.6V for function
5 V boost converter (MT3608)	10	$8.87	Ensures the NEO-6M receives the correct voltage range from the battery. NEO-6M needs roughly 5V for function.
SMA Extenders	3	$11.85	Connects antenna to outer side of node housing
Asa Filament	4	$56.69	Weather Proofing for Node Housing
Gromit Kit	1	$7.48	Weather Proofing for wiring and climate
SOT-23-6 to DIP adapter	5	$6.25	The load switch chip is tiny; these give it pins you can wire up
TPS22917DBVR Load Switch	10	$3.96	Lets the STM32 fully cut power to the GPS and SEN66 between uses; 2 per node, plus spares
18650 battery holder with wiring	8	$6.69	Holds the cell and gives us wires to connect
Total		$199.88	

Sensor Hardware
Node Design Components
Anything Else

4 Design

4.1 Design Context

4.1.1 Broader Context

The Cicada project is initially designed for the context of field fire detection in an agricultural setting. However, the data collected by the mesh work has benefits that extend far beyond the farm. The modular nature of the nodes allows the technology to be easily adapted for other contexts, such as remote wilderness or populated urban areas. In particular, Cicada contributes to the following societal needs:

1) Public Health, Safety, and Welfare

The public safety implications of a fire detection system are evident. The earlier a fire is identified, the easier it will be for first responders to intervene. Since the LoRa-based communication does rely on existing network infrastructure, it is well-suited for sparsely populated rural areas where fires could otherwise spread far before anybody notices. Quicker detection also allows for the public to evacuate if necessary. In emergency situations every minute is critical, and Cicada aims to accelerate fire detection.

2) Cultural and Social

The primary societal group Cicada is designed for are the farmers whose fields will house the nodes. It can be assumed that the typical farmer does not have the expertise with electronics necessary to make repairs. Therefore, the nodes will be designed to have a long lifetime to ensure they can exist in outdoor conditions for years without any physical maintenance. Additionally, we will consider adding soil sensors and other probes convenient for farmers. The LoRa communication allows a range large enough such that the nodes can be placed around a field so that they do not impede large farm equipment.

3) Environmental

Despite the Cicada nodes being exposed to the elements, each node will be sufficiently self-contained so that there is no impact on the environment. The electronics will be housed in weather-safe plastic, with the air sensor being the only exposed component. Thus, any potentially dangerous or fragile wiring will be hidden away from curious wildlife.

4) Economic

The technology required for LoRa communication and reliable sensors for multiple nodes is not cheap. However, our team is making an effort to minimize cost at every opportunity. The cost for the Cicada system remains small compared to the potential loss farmers face when at risk of an undetected field fire.

4.1.2 Prior Work/Solutions

Since fire detection is so critical for public safety, there are many existing technologies seeking to tackle the issue. Cicada has drawn much inspiration from these previous projects by implementing the strategies that seem most effective. However, the exact capabilities of Cicada are distinct from all other products due to its modularity and farmland-friendly design.

Perhaps the most similar existing project is Dryad, a startup originating in Germany [1]. The core strategy of Dryad matches the Cicada approach; both have nodes which measure for fires using third-party sensors, and broadcast the data through a LoRa mesh network. A significant difference between the approaches is that Cicada is built for agricultural contexts, whereas Dryad is based in forests. While Dryad nodes can be reliably affixed to trees, we may not have that luxury in an open field. Cicada nodes will instead have a casing specifically for sitting on top of poles.

We also draw inspiration from AranyaLink, which has the benefit of being an open-source fire detection system [2]. The public-available code allows our LoRa team to understand the intricate detail of how nodes can communicate with each other. In particular, the AranyaLink packet structure (right) demonstrates where node IDs are necessary in the networking flow. Cicada will modify this approach as necessary. For example, since LoRa has relatively low bandwidth we will reduce the packet fields to use as few bits as possible.

```
typedef struct __attribute__((packed)) AranyaPacket {
    uint32_t      packetID;
    int           originNodeID;
    float         temperature;
    float         humidity;
    uint8_t       flameDetected;
    double        latitude;
    double        longitude;
    unsigned long timestampMillis;
    int           verifyingNodes[5];
    int           verificationCount;
} AranyaPacket;
```

Figure 3: AranyaLink Packet Structure

Fire detection is not a mission limited to startups and personal projects. Large tech companies also contribute their capabilities to the fight against fires. Google uses satellite imagery and their mapping software to track near-real-time boundaries of wildfires [3]. This is relevant because Cicada makes use of the Google Maps API to provide a visual representation of the nodes' locations to the user.

4.1.3 Technical Complexity

The complexity of the Cicada project is the result of the large quantity of subsystems necessary to measure, broadcast, receive, parse, store, and display sensor data. Our team consists of computer, cybersecurity, and software engineers. Each of these curriculums provides many contributions to the design. Early in our design process we divided into subgroups to focus on one crucial topic: Sensors, LoRa, and Dashboard. To be successful, these teams require expertise in the following skills:

- Sensors Team
    - Power Consumption: for determining the batteries and voltage converters necessary to power the desired sensors.
    - Digital Simulation: for evaluating the trade-offs between sensor polling rates and battery longevity.
    - Zephyr RTOS: for managing the microcontroller and configure for communicating with external hardware.

- LoRa Team
    - Self-Healing Mesh Networks: for algorithms to most efficiently transmit data even while nodes lose functionality.
    - Packet Structuring: for transmitting sensor data and returning acknowledgements in as few bits as possible over a low-bandwidth LoRa network.
- Dashboard Team
    - Database Design: for constructing a system to reliably store historical sensor readings for each node in a relational structure.
    - RESTful APIs: for converting raw JSON data into meaningful objects which can be stored in the database.
    - Web Development: for creating a responsive user interface for users to intuitively access their desired data and receive alerts.

Each of the teams must communicate with each other to ensure consistent design and to not leave vulnerabilities in the data pipeline.

4.2 Design Exploration

4.2.1 Design Decisions

Throughout the development of Cicada, our team has made several important design decisions regarding the communication network, sensor selection, power system, and software architecture. These decisions were influenced by the project's requirements for long-range communication, low power consumption, and reliable fire detection in fields.

1) Lora Mesh Network

One of our primary design decisions was to use LoRa communication between our sensor nodes rather than relying on traditional Wi-Fi or cellular connections. Since the fields often have limited network infrastructure, we need a communication method capable of transmitting data over long distances while consuming minimal power. We selected a LoRa-based mesh architecture because it would allow sensor nodes to relay information through neighboring nodes before reaching the base station.

2) Sensor Selection

Another important decision was selecting the sensors used to detect potential field fires. Our research considered multiple environmental measurements, including temperature, humidity, and volatile organic compounds (VOCs). We prioritized sensors capable of measuring multiple environmental conditions to reduce the number of individual components required per node. Our team will use the Sensirion SEN66 sensor because it supports several measurements relevant to fire detection.

3) Solar-Powered Sensor Nodes

Our team decided to use stationary solar panels with rechargeable batteries to power our sensor nodes. Initially, we developed a solar-tracking prototype that used light sensors and motors to

adjust the panel's orientation toward the sun. However, after considering wind resistance, additional power consumption, and cost, we decided to move forward with stationary panels. This approach reduces mechanical complexity, lowers costs, and improves reliability, making it more practical for deployment in agricultural fields.

4) Software Architecture and Database Selection

For our software system, we selected a React frontend, Spring Boot backend, and PostgreSQL relational database hosted on a virtual machine. This architecture separates the user interface, application logic, and data storage into individual components, allowing each part to be developed and tested independently.

We chose PostgreSQL after comparing relational and non-relational database options. PostgreSQL was our final choice because it supports structured relationships between data, provides persistent storage for historical sensor data, and integrates with Spring Boot through JPA/Hibernate.

These software decisions provide a modular foundation that can be expanded as additional sensor types, nodes, and dashboard features are newly introduced.

4.2.2 Ideation

Our choice of smoke sensor affects how quickly and reliably the system can detect smoke, how many nodes we can deploy within budget, and how often customers will need to service those nodes. We considered sensors that measure airborne particles, sensors that respond to changes in gas concentrations, and environmental sensors that could help interpret those readings.

Advisor-provided VOC sensor and **Sensirion SGP40.**

Our advisors provided a VOC sensor for initial evaluation, and we also considered the SGP40. Changes in VOC levels could provide a signal of combustion, but other sources can also raise VOC levels. However we discovered that it is intended for indoor air quality only, not outdoor.

STMicroelectronics LPS22HB and HTS221

The LPS22HB measures barometric pressure, and the HTS221 measures relative humidity and temperature. These measurements could provide context for interpreting sensor readings under different weather conditions. Neither device measures airborne smoke particles.

MQ-4

This heated gas sensor primarily detects methane. Its heater can draw up to 1 W, and the manufacturer specifies at least 48 hours of preheating under its standard test conditions. Its heater demand and methane-focused response made it a poor fit for a battery-powered node intended to detect airborne smoke.

Analog Devices ADPD188BI

This optical module uses two wavelengths to detect light scattered by airborne particles. It offered a way to sense smoke directly, but we would still need to integrate a smoke chamber and evaluate the module's response to smoke, dust, and other aerosols in our enclosure.

Sensirion SEN55 and SEN66.

Both modules measure particulate matter, relative humidity, temperature, and VOC and NO_x indices. The SEN66 also measures CO_2. An integrated module gives us several measurements through one interface, with PM2.5 providing a starting point for detecting smoke. Outdoor dust can also raise PM2.5 readings, so we need to evaluate how well those readings distinguish smoke from other particle sources.

Renesas solution

We considered this option for its potential to reduce power consumption. We had limited evidence about its performance in our application, so its suitability depended on measuring both its power draw and its response to smoke.

4.2.3 Decision-Making and Trade-Offs

A lower efficacy in our smoke detector means we may need more nodes per area. More nodes means both more costs to cover a given area of land and more nodes to maintain which puts a burden on our customers. The benefits however are that more nodes per area relaxes the requirements on our networking implementation because each node needs to transmit over less distance with more nodes in range.

A higher cost per node means that the product either costs more for our end customer or they can cover less field area. Each node therefore needs to be far more effective than a cheaper one if we are willing to spend the extra money.

A higher power consumption means that our nodes need to be serviced more frequently which puts a time and monetary burden on our customers creating a worse experience for them. A more power hungry smoke detector may need to be polled less often which limits our time to alert.

For our prototype we decided to go with the sensirion 66. It seems to be the most tested solution which aides in our hopes of having a highly effective network. It also makes monetary sense due to the fact that many of the sensors are on the same board in case we want each of our nodes to be able to utilize more information in the future.

4.3 Proposed Design

4.3.1 Overview

Project Cicada is a network of small, solar-powered sensor nodes that watch farm fields for fire. Each node checks the air for smoke and sends a short radio message (a "chirp") to a receiver at the edge of the field. The receiver passes the data to a server at Iowa State. A web application shows every node on a map and sends a warning if any node picks up smoke.

We built it for places where fire is a real risk but there's no Wi-Fi or reliable cell service, such as Iowa farm fields during harvest. The goal is to catch a fire in its first few minutes, before it spreads, instead of waiting for someone to notice the smoke.

The system has four parts:

1) Sensor nodes**:** each node is a pole with a weatherproof case on top. Inside is an air sensor, a small computer, a radio, a GPS, and a battery charged by a solar panel.
2) Radio link: LoRa, a long-range radio that can send small messages several kilometers using very little power.
3) Gateway: one receiver per site that collects messages from every node and sends them over the internet.
4) Server and dashboard: stores the readings, decides when something looks like fire, and alerts the farmer and first responders.

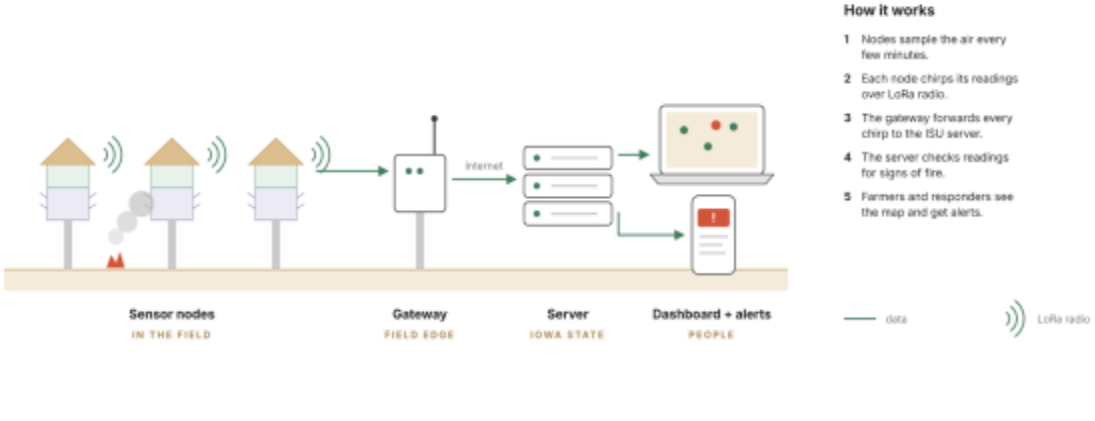


Figure 4: System Overview

4.3.2 Detailed Design

Cicada uses a star network. Each node talks directly to one gateway, and the gateway sends everything to a server on the ISU network. In semester 2 we plan to add relaying, so a node that's out of range can pass its messages through a neighboring node.

Data moves one way: node → gateway → server → dashboard. Each step uses a standard interface (LoRaWAN, MQTT, SSH, and a log file). That lets each sub-team build and test its own part separately.

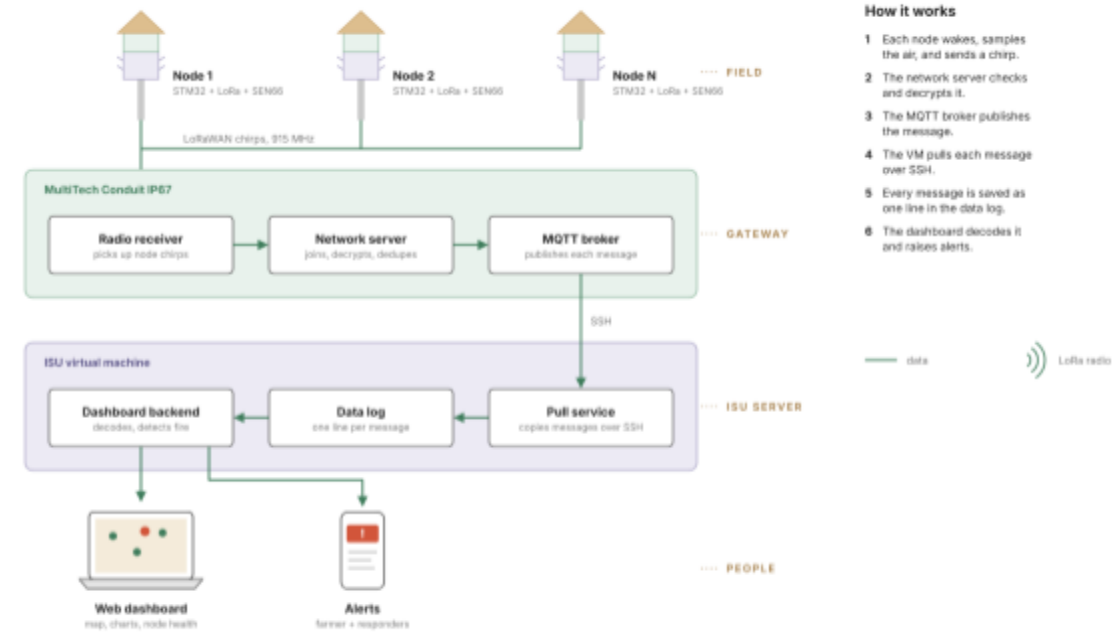


Figure 5: Dataflow Overview

Sensor node

Each node runs on an STM32 Discovery board (B-L475E-IOT01A) with a Semtech SX1272 LoRa shield stacked on top. The shield talks to the board over SPI and connects to a 915 MHz antenna. Two sensors connect to the board:

- Sensirion SEN66 (I2C): measures particulate matter (PM1 to PM10), VOCs, NOx, CO_2, temperature, and humidity. Smoke shows up as a sharp jump in PM2.5 and VOCs. We check several readings together so the node can tell smoke apart from harvest dust.
- NEO-6M GPS (UART): records where the node is, so every alert includes exact coordinates.

The firmware is written in C on Zephyr RTOS. The node wakes up, takes a reading, compares it to fire thresholds, sends a short message, and goes back to sleep. If readings stay high for several samples in a row, the node raises an alert and starts reporting more often.

Power

A 5–6 V solar panel charges a Samsung 35E 18650 battery through a TP4056 charger. On the bench, the charger can run from USB-C instead. An MT3608 boost converter raises the battery voltage to 5 V for the STM32 board and the GPS. A separate 3.3 V regulator powers the SEN66.

To save power, the GPS and the SEN66 each sit behind a TPS22918 load switch. The STM32 turns each switch on (pins D7 and D6) only while it reads that sensor, then turns it off again. This matters most for the SEN66, which uses the most power because of its built-in fan.

Enclosure

The case is a 3D-printed stack of three sections that screw together with O-ring seals. It's about 8 in wide and 12.5 in tall.

- Power cap (top): a sealed cone that holds the battery, charger, and boost converter.
- Electronics bay (middle): a sealed section for the STM32, LoRa shield, GPS, regulator, and load switches.
- Sensor bay (bottom): a vented section for the SEN66. Angled gills shade the sensor and let air through, and a stainless mesh keeps insects out.

Wires pass between sections through sealed rubber grommets. The whole stack slides onto a 1-inch metal conduit pole.

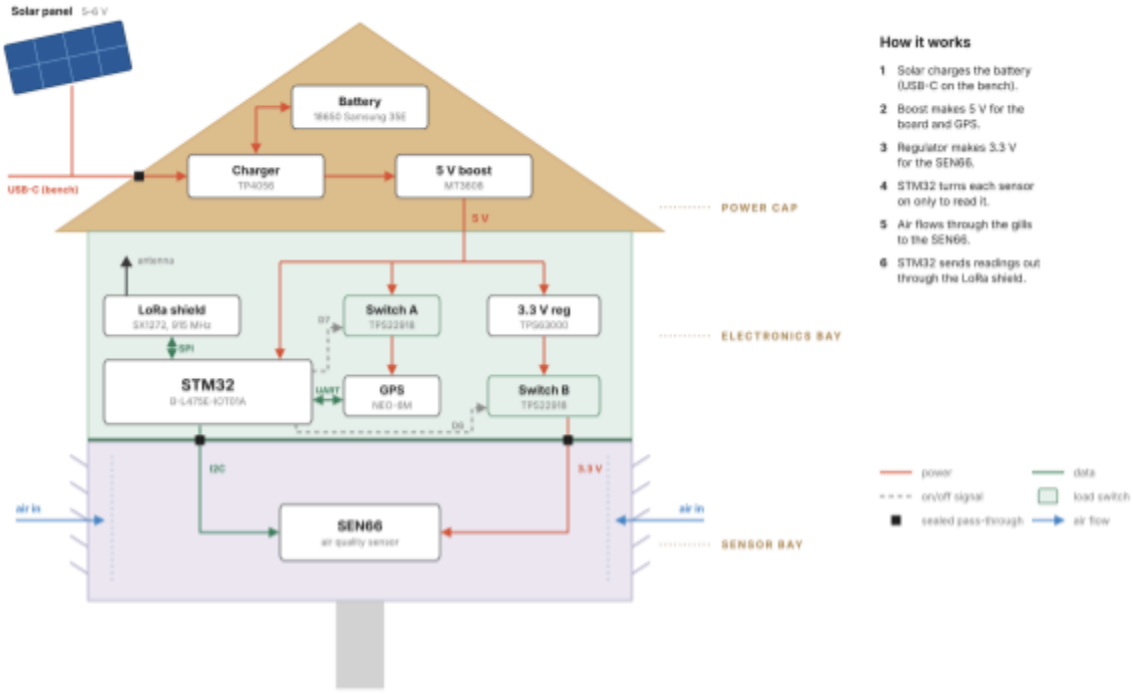


Figure 6: Node Power Diagram

Gateway, Server, and Dashboard

A MultiTech Conduit IP67 gateway receives every message, runs the LoRaWAN network server, and publishes each message to a local MQTT broker. A virtual machine on the ISU network connects to the gateway over SSH, pulls every message, and saves it to a log file. The dashboard reads that file as it updates, decodes the readings, and shows each node on a map with its latest data, battery level, and signal strength. When a node reports smoke, the server double-checks the alert before notifying anyone.

Security

LoRaWAN encrypts every radio message with AES-128 keys unique to each device. The link from the gateway to the server uses SSH keys, and the server is on the ISU network. In semester 2 we plan to add end-to-end message authentication so alerts can't be faked.

4.3.3 Functionality

From the user's side, Cicada is meant to be set up once and then left alone.

Setup: The farmer puts up a pole, slides the node onto it, aims the solar panel south, and twists the cap closed to turn it on. Within a few minutes the node finds its GPS location, joins the network, and shows up on the dashboard map.

Day to day: Nodes report every few minutes on their own. The farmer can check air conditions anytime, and the dashboard flags any node that stops reporting or runs low on battery.

When there's a fire:

1) A combine sparks a fire in dry crop residue.
2) Smoke reaches the nearest node, and its PM2.5 and VOC readings spike.
3) The readings stay high for several samples, so the node raises an alert and starts reporting every minute.
4) The alert goes through the gateway and reaches the server within seconds.
5) The server confirms it, either because the alert keeps coming or because a nearby node also picks up smoke.
6) The farmer and first responders get a warning with the node's GPS location. Nearby nodes that pick up smoke show which way the fire is moving.
7) Once the air clears, the node goes back to normal and someone closes the alert on the dashboard.

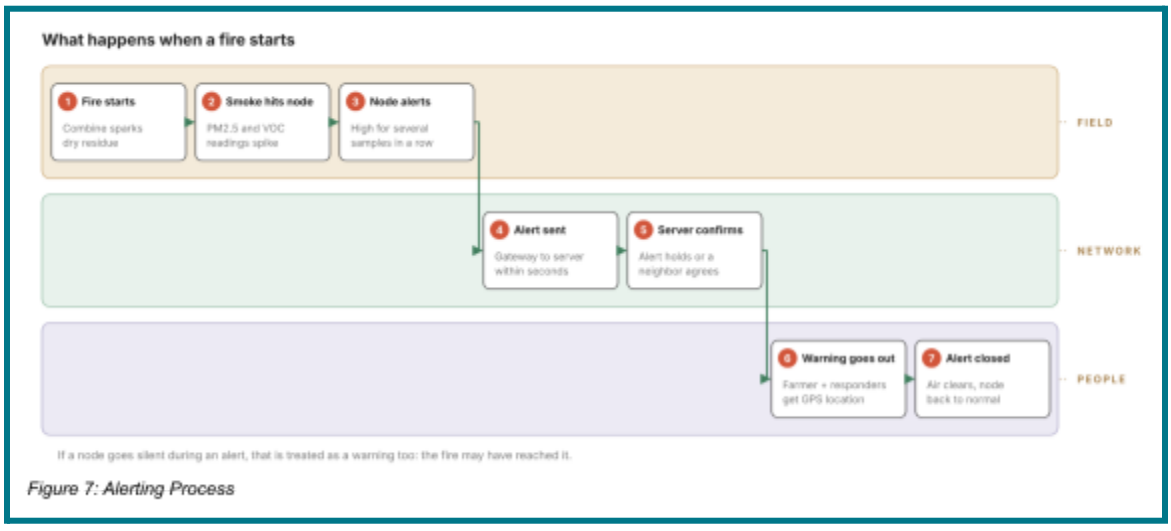


Figure 7: Alerting Process

If a node stops reporting in the middle of an alert, we treat that as a warning sign too, since the fire may have reached it.

Maintenance: The battery sits under the cone cap, so swapping it just means unscrewing the cap. The insect mesh slides out for cleaning.

4.3.4 Areas of Concern and Development

The design covers our main requirements: it works without Wi-Fi or cell service, runs on solar, reports its location, and uses affordable, off-the-shelf parts. Using our development boards, we've sent data from a board through the gateway to our server, and a range test between two campus buildings showed low packet loss. Parts for the full nodes are on order, so no node has been built or deployed in the field yet. What we still need to prove is that the system reliably detects real fires, lasts a full year on solar, and keeps a good signal in tall crops.

Next steps:

1) Assemble the first full node once parts arrive.
2) Finish the node firmware.
3) Measure power use and set the reporting interval from the results.
4) Print, assemble, and water-test the first enclosure.
5) Run controlled smoke tests on the bench, then collect baseline air data at Curtiss Farm.
6) Make the data pipeline reliable, with a persistent connection and log rotation.
7) Deploy several nodes and the gateway at Curtiss Farm for field testing.

4.4 Technology Considerations

Our project uses several technologies across hardware, networking, and software. Each technology was selected based on its reliability, cost, power efficiency, and compatibility with the overall system.

LoRa Communication

We chose LoRa because it supports long-range communication while consuming relatively little power. This makes it suitable for agricultural fields where Wi-Fi and cellular coverage may be limited. However, LoRa has lower data transfer rates compared to other wireless technologies, and signal reliability can be affected by interference and obstacles. We considered other communication methods, including Wi-Fi and cellular, but LoRa better fits our requirements for low-power, long-distance communication.

STM32 and Zephyr RTOS

Our sensor nodes use STM32 microcontrollers running Zephyr RTOS. STM32 provides the processing capabilities needed to collect sensor readings and communicate over LoRa while maintaining low power consumption. Zephyr provides built-in drivers and tools for managing hardware components, reducing the need to develop everything from scratch. However, working with an RTOS introduces additional complexity when integrating sensors and managing power consumption.

Backend and Database

We chose Spring Boot for our backend and PostgreSQL for data storage. Spring Boot provides a structured framework for developing RESTful APIs and integrates with JPA/Hibernate for database operations. PostgreSQL supports relationships between nodes and their historical sensor readings. We considered other databases, including MySQL and MariaDB, but selected PostgreSQL based on its relational capabilities and compatibility with our backend. One consideration is managing database schema changes safely as the project evolves, particularly when preserving previously collected sensor data.

Frontend and Mapping

Our dashboard uses React with the Google Maps API to display sensor locations and environmental readings. React allows us to organize the interface into reusable components, while Google Maps provides satellite imagery that helps farmers identify where sensors are located in their fields. One trade-off is that Google Maps requires internet connectivity and may introduce usage costs. However, its existing mapping features reduce development effort compared to building a custom mapping system.

4.5 Design Analysis

So far, our team has made significant progress in developing and testing the individual components of Cicada. While the complete system has not been fully integrated, our initial testing has shown that the proposed design is feasible, particularly in terms of LoRa communication, backend development, and data storage.

For the LoRa network, we successfully established communication between STM32 nodes and the MultiTech Conduit gateway. We conducted several range tests to evaluate signal strength, packet loss, and reliability in different environmental conditions. One test across campus (Beardshear Hall to Curtiss Hall) achieved a 94.3% packet delivery rate over approximately 0.2 miles. However, testing in wooded areas showed that obstacles and terrain would significantly make the signals weaker. We have also developed the packet structure and started implementing acknowledgement, retransmission, neighbor discover, and packet forwarding functionality. Further testing is needed to verify reliable three-node communication and eventually support self-healing mesh networking.

For the sensor and power systems, we developed initial firmware using Zephyr RTOS and began implementing unit tests. We also created simulations to estimate battery consumption based on sensor warm-up times, measurement intervals, and sleep cycles. Our research showed that the SEN66 sensor's warm-up period consumes a significant amount of energy, making the polling interval an important factor in battery life. Based on these findings, we are working on optimizing power consumption through adjustable polling intervals and load switches that can turn sensors off when they are not needed. We have also developed and tested initial 3D-printed enclosure prototypes, although weatherproofing and final hardware integration still need to be validated.

For the software system, we successfully configured our virtual machine, deployed a Spring Boot backend, and established a PostgreSQL database. We implemented HTTP endpoints that can

process sensor uplink data and store it as database records. We also configured a GitLab CI/CD pipeline to automate the building, testing, and deployment of backend changes. Additionally, we tested JPA/Hibernate's ddl-auto=update functionality to understand how changes to our Java entities affect existing database records. These tests showed that adding or removing entity fields did not delete historical data, although renaming fields and other structural changes may require manual database migrations. On the frontend, we have created a React project and successfully tested connections to the backend and Google Maps API, but the complete interactive dashboard is still under development.

Overall, our results suggest that the major components of Cicada can work as intended. Most of the challenges we have encountered involve integration, power optimization, and physical design rather than fundamental problems with our proposed architecture. Our next steps are to complete the three-node prototype, integrate the SEN66 and GPS sensors, finalize the stationary solar power system, and continue developing the dashboard. We also plan to conduct controlled smoke tests, measure actual power consumption, and perform outdoor field testing to evaluate detection accuracy, communication reliability, and weather resistance. These tests will help determine whether the complete system meets our requirements before expanding to a larger network.

5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, give an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure do not break? Is it driven by requirements? Tools?

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional, are being met? How would you involve your client in the acceptance testing?

5.7 Security Testing (if applicable)

How will you demonstrate that the security requirements, if any, are being met? What data and/or resources are being protected? What security testing principles, methods, and/or tools may be applied?

5.8 User Testing

Describe how the team will include real users to interact with the solution/product. Your user testing plan should evaluate your solution/product's usability and overall user experience by gathering user feedback through observations, surveys, etc. User feedback allows the team to identify areas for improvement and ensure the product meets user needs.

5.9 Testing Results

What are the results of your testing thus far? Include any numerical, graphical, or qualitative testing results here. How do they demonstrate compliance with the requirements or addressing user needs? Use a summary narrative to discuss what you've learned and what next steps need to be taken.

6 Implementation

Describe any (preliminary) implementations of your design thus far. Support any general, descriptive text with relevant images. If your project has inseparable activities between design and implementation, you can list them either in the Design section or this section.

7 Ethics and Professional Responsibility

Use this section to describe your considerations of engineering ethics and professional responsibility. Most importantly how are you defining engineering ethics and professional responsibility in the context of your project and what steps are you taken to ensure ethical and responsible conduct. Each section references one type of ethical/professional responsibility considerations. You may also use this introductory section to note any overarching ethical philosophy among your team.

7.1 Areas of Professional Responsibility/Codes of Ethics

This discussion is with respect to the paper by J. McCormack and colleagues titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

Pick one of IEEE, ACM, or SE code of ethics (all linked in class slides). Create a table, like Table 1 in the McCormack et al. (2012, pg. 418)) paper, with the following columns representing: Area of Responsibility (from the paper), Definition (in your own words), Relevant Item from Code of Ethics (from the Code of Ethics you selected, and description of how your team has interacted with that area of professional responsibility or adhered to that code during your project thus far.

In text below the table, describe one area in which your team is performing well. Describe what your team is doing and how that signifies strong performance. Also describe one area in which your team needs to improve. Describe what your team is currently doing and what it should do in the future to improve.

7.2 Four Principles

Create a table with rows for each broader context area (see Section 4.1.1) and columns for each of the four principles (beneficence, nonmalificence, respect for autonomy, and justice; see Beauchamp, 2007). Within the table, identify at least one way each of the four principles applies to each of the broader context areas. Some principle-broader context connections might be more prominent than others, but you should be able to identify something for each table cell. Note: Your design may end up negative or neutral in some cell. For example, your product might perform poorly in environment-nonmaleficence because it utilizes natural resources without a positive/mitigating effect.

Below the table, note one broader context-principle pair that is important to your project. Briefly describe the benefit in that area you are working towards and how you will ensure it. Also note one broader context-principle pair in which your project/end design is or will be lacking. Describe either (a) how this negative is overcome by other positives in other areas of the project/design or (b) what your team must do to improve in this area.

7.3 Virtues

List and define at least three virtues that are important to your team. Describe what you will do or have done as a team to support these virtues among all team members.

Each team member should also answer the following:

- Identify one virtue you have demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - How have you demonstrated it?
- Identify one virtue that is important to you that you have not demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - What might you do to demonstrate that virtue?

8 Closing Material

8.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals. What constrained you from achieving these goals (if something did)? What could be done differently in a future design/implementation iteration to achieve these goals?

8.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE). See link:

https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf

[1] B. Jones, "Fighting Wildfires with LoRaWAN," dryad.net, Aug. 22, 2022. [Online]. Available: https://www.dryad.net/post/fighting-wildfires-with-lorawan. [Accessed Nov. 10, 2026].

[2] The Uncertified Engineer, "AranyaLink — Open-Source Forest Fire Detection Mesh Network," instructables.com, Jun. 7, 2026. [Online]. Available: https://www.instructables.com/AranyaLink-Open-Source-Forest-Fire-Detection-Mesh-/. [Accessed Nov. 10, 2026].

[3] Z. Ben-Haim and O. Nevo, "Real-time tracking of wildfire boundaries using satellite imagery," research.google, Feb. 3, 2023. [Online]. Available: https://research.google/blog/real-time-tracking-of-wildfire-boundaries-using-satellite-imagery/. [Accessed Nov. 10, 2026].

8.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics/layout, PCB testing issues, software bugs, etc.

9 Team

Complete each section as completely and concisely as possible. We strongly recommend using tables or bulleted lists when applicable.

9.1 Team Members

9.2 Required Skill Sets for Your Project

If feasible – tie them to the requirements.

9.3 Skill Sets Covered by the Team

For each skill, state who covered it.

9.4 Project Management Style Adopted by the team

Typically, waterfall, agile, or hybrid for project management.

9.5 Initial Project Management Roles

Enumerate which team member plays what role.

9.6 Team Contract

Team Members:

1) _____Cooper Hoy_____ 2) _____Marcus Yeung_____

3) _____Hrushi Bhatt_____ 4) _____Jack Rice_____

5) _____Sean Fjellman_____ 6) _____Caleb Kugel_____

7) _____Jake Ebersole_____

Team Procedures

1) Day, time, and location (face-to-face or virtual) for regular team meetings:
2) Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
3) Decision-making policy (e.g., consensus, majority vote):
4) Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
3. Expected level of communication with other team members:
4. Expected level of commitment to team decisions and tasks:

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
2. Strategies for supporting and guiding the work of all team members:
3. Strategies for recognizing the contributions of all team members:

Collaboration

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
2. Strategies for encouraging and support contributions and ideas from all team members:
3. Procedures for identifying and resolving collaboration or contribution issues (e.g., how will a team member inform the team that the team environment is impeding their opportunity or ability to contribute?)

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
2. Strategies for planning and assigning individual and team work:
3. Strategies for keeping on task:

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
2. What will your team do if the infractions continue?

****

a. I participated in formulating the standards, roles, and procedures as stated in this contract.
b. I understand that I am obligated to abide by these terms and conditions.

c. I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) _____ DATE _____

2) _____ DATE _____

3) _____ DATE _____

4) _____ DATE _____

5) _____ DATE _____

6) _____ DATE _____

7) _____ DATE _____

8) _____ DATE _____