6-2022

Hardware and Software Infrastructure for an Agricultural Sensor Network

Bennett Dorsey

Mariela Zuniga

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Bennett Dorsey, Mariela Zuniga

ENTITLED

Hardware and Software Infrastructure for an Agricultural Sensor Network

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

6/4/22

Thesis Advisor(s) (use separate line for each advisor) date

Shoba Krishnan

June 7th, 2022

Department Chair(s) (use separate line for each chair) date

Hardware and Software Infrastructure for an Agricultural Sensor Network

By

Bennett Dorsey, Mariela Zuniga

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Electrical Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Electrical Engineering

Santa Clara, California

2022

1 Abstract

With the rise of low-power processing and the availability of cheap solar, our project aims to create a low-power sensor network that can gather and report data with little oversight. The project will use a bluetooth mesh network to create a generalized solution where the network can start up regardless of the environment or number of nodes. Our tasks include designing algorithms that tackle redundancy in the face of node failure, self test and diagnostics, and power on and off strategies. The project aims to solve the problem of timing and communication when power is inconsistent and not shared. Our current task is working on developing the algorithms necessary for the network to function and communicate information properly.

2 Acknowledgements

We would like to acknowledge our advisor Dr. Wolfe for his help, support, and guidance on this project.

Contents

List of Figures

3 Chapter 1: Introduction and Motivation

3.1 Background and motivation

The development of smart farming technologies in order to automate the agricultural industry has been a relatively recent practice. Their real world applications could result in the increase of crop yields, the improvement of productivity, and the preservation of existing resources. Low power sensor network modules for agricultural applications currently require batteries, or wired connections in order to communicate for long periods of time. This requires a significant amount of infrastructure or replacement of sensors every few years. The current solution for this type of problem is to have a base station that communicates with a larger antenna to all of the nodes. Our project aims to be a cheaper and more sustainable solution without the need for this large base station which also removes the need to replace the sensors when the batteries run out.

Our project will use the new Bluetooth mesh specification for our network. This specification was only recently released in 2017. Figure 1 shows a graph model of a Bluetooth mesh network with the different types of nodes available for use. The relay mode relays messages received in order to extend the range of other devices. Friends and low-power nodes work together in order for the low power devices to save energy. Each device can act as multiple types of nodes; for example nodes can be both relay and friend nodes. A network structure of this kind can allow for us to prioritize longevity and resiliency. It is important for us to develop this remote sensing network in order to respond to disturbances in a meaningful way and in turn optimize our farming systems.

4 Chapter 2: Project Goals, and Objective

4.1 Problem Statement

Our project goal is to set up a resilient Bluetooth mesh network that is able to function under the constraints presented by an agricultural setting. The constraints we are working under are an intermittent power environment. In order to avoid having batteries which will limit the lifespan of our device, we are assuming that we only have solar, which means that our power could be cut quickly at any point. In order to develop the network of Bluetooth devices, we have created a demo application that includes: a self-diagnosis

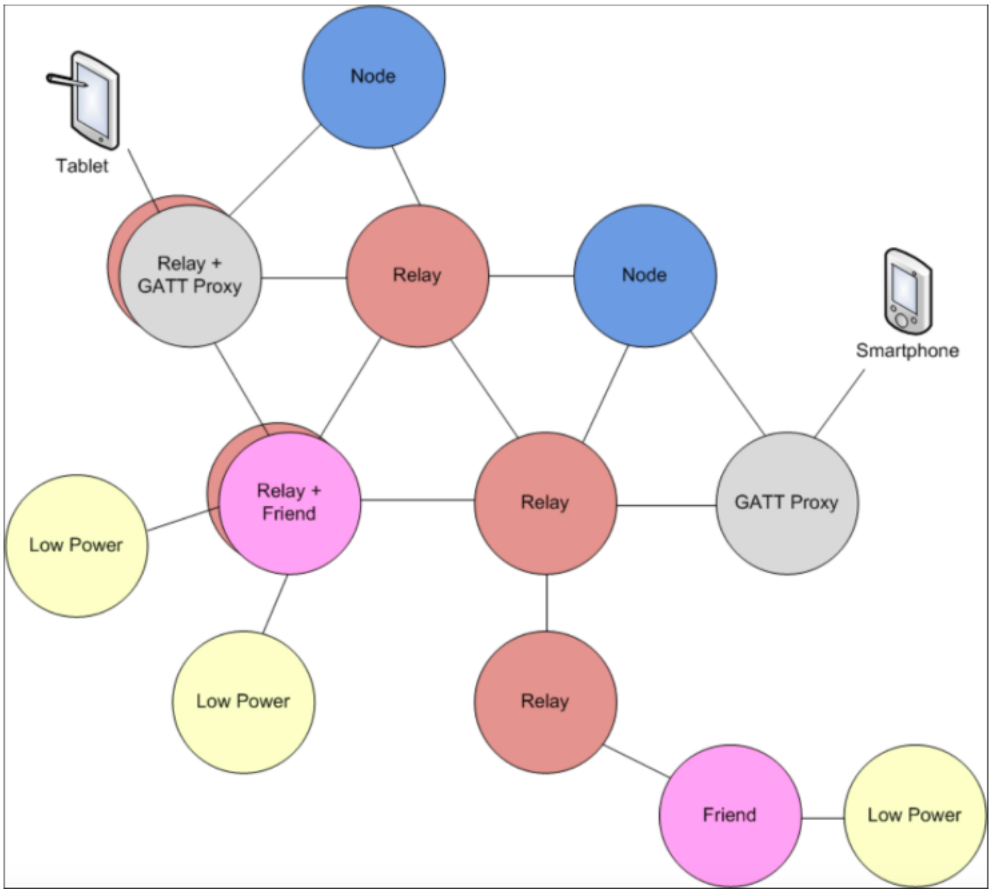

Figure 1: This image shows the types of nodes present and the graph model of a mesh network.

strategy which will monitor for evidence of any faults, management of redundant data, a power on and off strategy, and data aggregation and consolidation. The power on and off strategy is the way in which the nodes behave as they enter and exit connectivity. The nodes need to realign their timing and re-synchronize their data in addition to network specific data. We believe that this technology can have a positive impact on the agriculture industry. With the ongoing threat of climate change, our world's agricultural industry is in a very vulnerable state, and we hope to create a remote sensing network that helps us respond to disturbances in a meaningful way.

5 Chapter 3: Experiment Design

5.1 Project Plan

The plan for the development of our project was to first build the network so that it can function under ideal conditions. Ideal conditions for our project means every node is always online and is connected to other nodes in the network. With these conditions each node has the ability to share and collect data, and add nodes to the network.

5.2 Network Setup

5.2.1 Client Server Model Background

The challenge is defining our client, server and provisioner models. The client and server models are internal configurations that Bluetooth uses to define the states of each device. The Cypress boards we are using were designed primarily for a smart home setup, and the client/server examples given are primarily based around lights and switches. The idea is that the client application acts as a switch, while the server application acts as the light that it turns on. For our purposes we need every device to have both of these models so that they can both update other devices and be updated themselves. If each node is only a client, then it is only allowed to send data out, but would never be able to request updates, or any other information from the rest of the network. We need our nodes to have both the capability f both sending and requesting information. This will ensure that every node has the capability to send and receive data requests.

5.2.2 Tools

J-Link Debugger The J-Link Debugger uses JTAG probes to allow us to access hardware debugging on the development boards as well as flash multiple development boards at once.

Modus Toolbox This the development environment provided by Cypress. It allows us access to memory mapping that makes hardware debugging easier. It also includes many useful tools for serial debugging and Bluetooth mesh control.

Git This is used for version control as well as collaboration for the project.

BLE Mesh This is the protocol our network will be using. The specification is provided by Bluetooth Special Interest Group.

5.2.3 Provisioner Background

The other challenge we have in building our network is the provisioner model. Bluetooth uses a secure network and therefore requires private key-transactions in order to add a new node to the network. New nodes start by broadcasting an unprovisioned signal, until a nearby network responds to add it to the network. The complication comes when we want to have multiple provisioners. Adding a device to your smart home mesh network is easy, because your phone always acts as the provisioner, but if we want to add a new node to the network, having a single point of failure in one provisioner is not what we are looking for. The complication then becomes how does the network decide which device is provisioning the new one and how is the information of this new device shared in the network.

5.2.4 Initial Network Build

To start solving this problem, we initially built our network without combining all of these models into one device and instead had them as separate devices as seen in Fig 7. This initial build had one provisioner, one client, and several servers it would update. Once we were confident, that the network could function with each of these working independently we could move on to combining the different models.

5.2.5 Combining Client/Server Models

The second implementation we had still only used a single provisioner, but we used both the client/server models on the same node, meaning each device works as both the client and the server. With this setup we needed to begin thinking about what kind of information we are sending. The first build can use a simple broadcast message and we know which device sent it, but now with multiple devices running identical firmware, we need to start sending different Metadata. In addition to the serial number of the device broadcasting, each message is coded with a timestamp. Fig 2 shows the graph model of this network build.

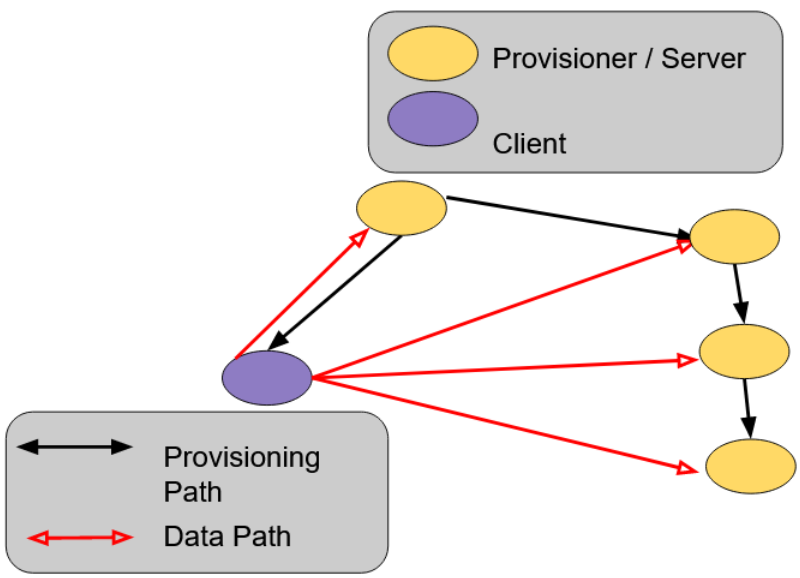

Figure 2: Graph of Initial Build with combining client and server

5.2.6 Multiple Provisioners

Our final test setup combined the provisioning model into the Client/Server model. All devices are now running the same firmware, can add new devices to the network, and both send and receive state changes. Fig 3 shows a graph model of this final network build. Verifying the success of this build was challenging because we had to force different devices to be provisioned by different ones. We did this by spreading them out farther across the room so the better provisioner would be chosen based on the close one. The algorithm to decide which device is going to be the provisioner is based on the RSSI(received signal strength indicator) metric collected when the provisioning process starts. The device with the strongest connection to the new device is chosen to exchange keys and add it to the network.

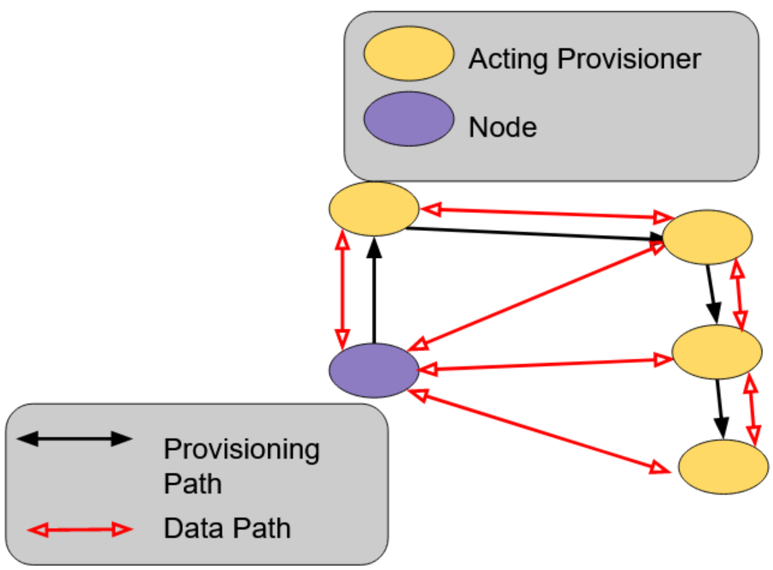

Figure 3: Graph showing Full Provisioner and Client/Server

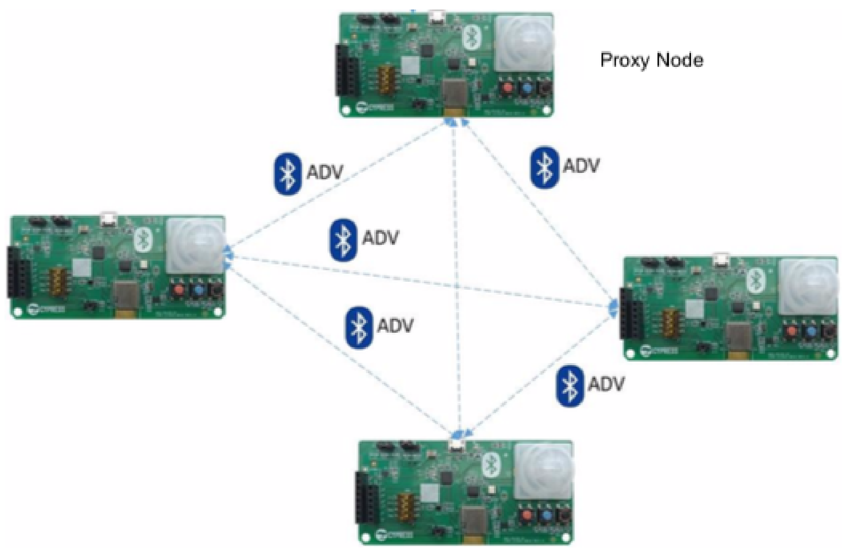

Figure 4: Diagram of Mesh Network

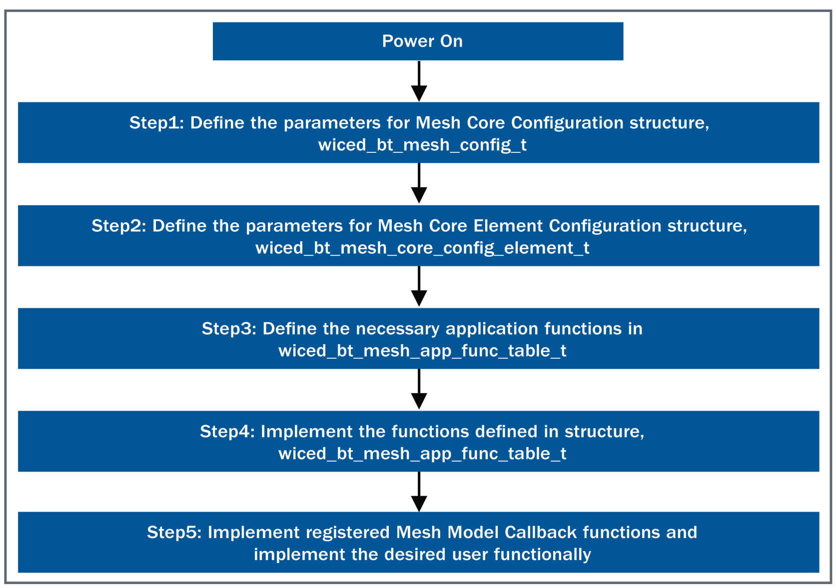

Figure 5: This image shows the steps to setting up a node in a Bluetooth mesh network

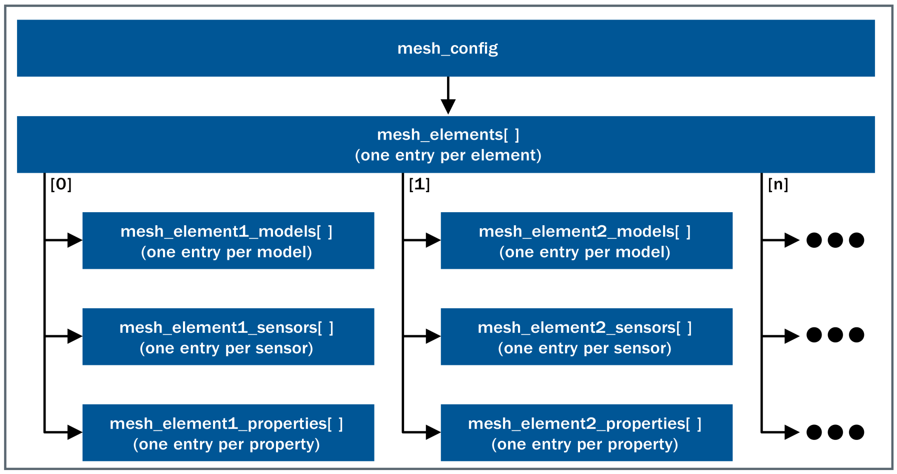

Figure 6: This image shows the hierarchy inside the configuration of each node

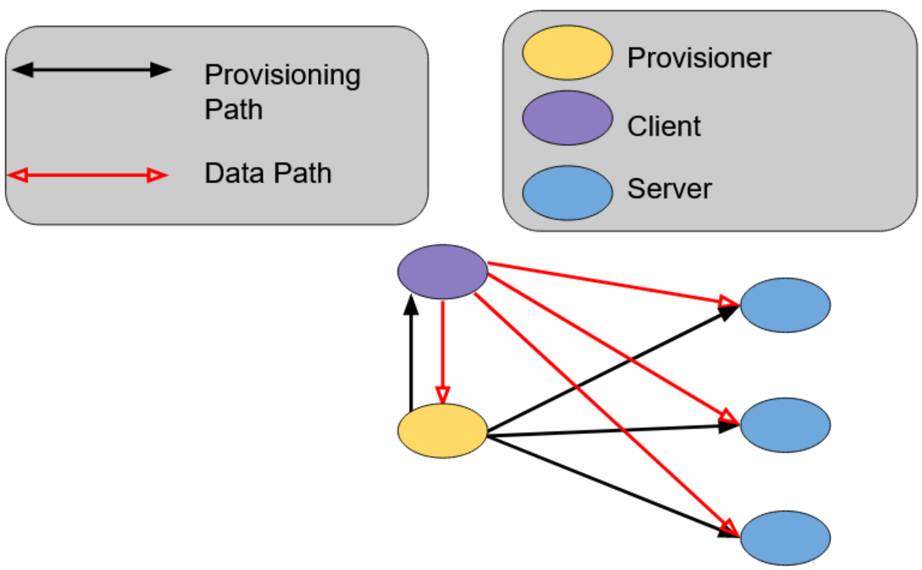

Figure 7: Graph of Initial Build with Only a single Provisioner, Client and Server

5.2.7 Messaging Protocol

The next part of our implementation is the structure of how our messages will be sent. Each message contains all of the sensor data that the node recorded. The messages also contain the metadata about date, time, and node id in order to properly store it. The difficulty in our message sending algorithms is making sure each node's list of all the data is consistent across all nodes. The master list that is aggregated and exported from the network should be consistent at each node. Maintaining this list consistency is a challenge, however with each message information tracked, updating Nodes to the current state is not difficult.

5.3 BLE Mesh Use Case and Issues

Many of the challenges we faced in implementing the different network setups was in dealing with the Bluetooth mesh protocol. As mentioned earlier, many of the desired use cases of Bluetooth mesh was for use in a smart home setting. The smart home network is very different from the agriculture sensor network we are designing. The smart home works fine with a single provisioning node, because every new device can just be added and controlled by your phone or computer. Fig 8 shows the layout of a typical smart home network. Similarly, the devices do not need to share a ton of information or know very much about the network. The smart switch in your kitchen only needs to know how to turn on the lights, and the lights only need to turn on and off. In this environment only the provisioner, or the phone, knows information about each device. This single point acts similar to a router in that sense and defeats the purpose of the mesh network for our purposes. We need to be able to pull information about all nodes from any node. Every device needs to have this functionality. Bluetooth does not recommend that you enable multiple provisioners or begin combining all of their models into one device, and we needed to be careful not to cause any problems. Even once all the models were combined, we still needed to synchronize the information.

Figure 8: Layout of a smart home mesh network

5.4 Test Environment and Fault Injection

We tested the network with simulated failures. Certain nodes were preset to go "offline" for set periods of time where they would not transmit or react to data. It was helpful for us if they still receive and record the data to make bug checking and verification easier, and we can look at the full log files from any device. In addition to the ”offline” state we also forced several different network topolgies in software. Node A cannot talk to Node C for examples so that we do not need to spread out the devices long distances in physical space to get new topologies.

To test our network with these different states, we are intentionally causing faults into how the network would prefer to operate. We have already shown that the network can function under ideal conditions, but we now need to show functionality under the constraints of the agricultural setting with low power, and longer distances.

5.5 Test 1: Power Strategy

The first test we worked on was the power strategy. This includes what the node is doing when receiving a low power input signal that it is shutting down soon or what it does immediately on startup. On receiving the low power signal, the node will first stop broadcasting and receiving messages, and then save all of it's local sensor data to memory for long term storage. This is to keep its data and all other important information stored while the RAM is saved on shutdown.

When the device powers on, it broadcasts a message asking for what it missed while offline. It does this by sending the id of the last message it received so the other nodes know what points it missed. From the received messages, the newly online node might send out sensor data or do nothing depending on what messages were received. This re-synchronizes the node with the network. The network has a drawback of frequently having a significant time delay in gathering some information out of the network. We are working with the assumption that this sensor data is not time critical to the minute and that this is not an issue.

Our success metric for this test was the rest of the network receiving the back online message, and then the "offline" node synchronizing itself with the rest of the network. To Verify this we looked at the log files on the other boards in the network to verify that they received the message. Fig 9 Shows an portion of the log file from the network, showing adding the node to the network, the heartbeat messages showing that is it online, and then the back online message and request for an update.

```
19:16:59.067 Set Local Device addr:0x0001 net_key_idx:0000 iv_idx:0 key_refresh:0 iv_updata:0

19:17:14.044 Provision Connect Provisioner:1 identify_duration:1 use_gatt:1
19:17:14.044 Provision status:2 Device UUID: 50 72 6d dd 2a 4b 6b c4 2f 9a 5d b2 ac d2 e6 e0
19:17:21.254 Provision: Device Capabilities from Server:1 Num elements:1 algo:1 pub key type:0 oob type:0 oob size:0 oob_action:0 input oob size:0
19:17:21.255 Provision status:3 Device UUID: 50 72 6d dd 2a 4b 6b c4 2f 9a 5d b2 ac d2 e6 e0
19:17:21.256 Provision Start Server:1 pub_key_type:0 auth_method:0 auth_action:0 auth_size:0
19:17:22.031 Provision: end from Server:1 addr:0002 result:0 dev_key:d0 ba 7e 40 72 9c f7 cc b8 8c 2c 39 36 1d 05 b8
19:17:22.034 Provision status:6 Device UUID: 50 72 6d dd 2a 4b 6b c4 2f 9a 5d b2 ac d2 e6 e0
19:17:22.092 Provisioning disconecting Link Status:0008 provisioner:1
19:17:22.093 Proxy Connect type:0
19:17:32.589 Node Connecting Link Status provisioner:0 addr:0002 connected:1 over_gatt:1
19:17:32.594 Set Dev Key addr:0002 dev_key:d0 ba 7e 40 72 9c f7 cc b8 8c 2c 39 36 1d 05 b8
19:17:32.704 Composition Data: from:2 page:0 CID:131 PID:3005 VID:2 CRPL:8 Features:0
19:17:32.705 Provision status:4 Device UUID: 50 72 6d dd 2a 4b 6b c4 2f 9a 5d b2 ac d2 e6 e0
19:17:32.727 Net transmit set addr:0002 count:3 interval:100
19:17:32.818 Net Transmit Status from:2 count:3 interval:100
19:17:32.962 Model App Bind addr:0x0002 element_addr:0002 company_id:ffff model_id:0002 app_key_idx:041d
19:17:33.055 Model App Bind Status from:2 status:0 Element addr:0002 Model ID:2 ApptKey Index:41d
19:17:33.067 Model App Bind addr:0x0002 element_addr:0002 company_id:ffff model_id:1005 app_key_idx:041d
19:17:33.176 Model App Bind Status from:2 status:0 Element addr:0002 Model ID:1005 ApptKey Index:41d
19:17:33.188 Model App Bind addr:0x0002 element_addr:0002 company_id:ffff model_id:1003 app_key_idx:041d
19:17:33.298 Model App Bind Status from:2 status:0 Element addr:0002 Model ID:1003 ApptKey Index:41d


19:19:13.328 Device Back Online, addr: 0x0002, Requesting Messages since: 19:18:07.089
```

Figure 9: Log Showing Offline test

5.6 Test 2: Aggregation

Once our network can function and aggregate data with all of the nodes online and relaying properly, the network needs to function under the constraints we have set forth for an agricultural environment. This means that we expect each node to have variable up time. The challenge here is to make sure that the network can recover the data and know which information needs to be re-relayed if certain relay nodes were down. Each node needs to store it's own data and potentially other nodes' data in order to more reliably aggregate the data.

To test this aggregation setting we had each node generate random data, and store the time stamps, and then attempt to gather all of the information from one specific node. Our tests succeeded and we were able to gather information from all up nodes and recover lost information when the "offline" nodes came back up. A

6 Chapter 4: Results and Discussion

6.1 Test Results

From our various tests we were able to accomplish adequate data aggregation through our managed flood operation. When a device receives a message, it has tags telling it whether it is supposed to process the message, or just rebroadcast it, so if the message is meant for node G, node B will not do anything, but send the message again. To avoid endless loops of the same message, each message has a number of 'hops' that are allowed. This allowed us to mitigate duplicate messages.

We were also able to have multiple provisioners in our network, and each is able to add other nodes. As for our power strategy tests, we concluded that our network is able to rebuild itself as more nodes come online.

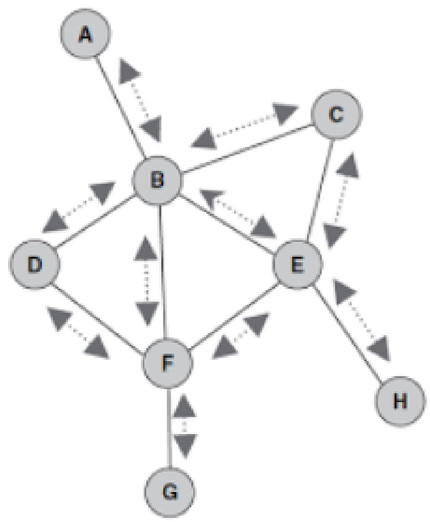

Figure 10: Managed Flood diagram

6.2 Network Performance

Through our aggregation tests as well as our power strategy tests, we were able to have our network receive and transmit message packets that contained the Cypress node id, the sensor data, and the time stamp. Additionally, through our power strategy tests, we noted that power variability is not critical as the network can function as long as there is an established path data can travel through. This verifies that the network can potentially be reduced to a fraction of its size so long as we have a path for the data.

6.3 Bill of Materials

Figure 11 below shows the budget for our project.

Item Name	Cost per Item	Quantity	Description
CYBT-213043-MESH-02 development boards	$127	2	Each board will act as a node for our sensing network, adding more nodes allows us to expand the range of our experiments.
8.08.00 J-LINK BASE	$476	1	This hardware debugger helps us step through and edit lines of code, which is essential for the development of our network's communication protocol.
Coin Cell Batteries	$12	1	We will need batteries for the sake of testing our nodes without connecting to our laptops
Serial Monitor Application	$40	1	Serial Monitor App for a MAC computer to read trace messages
Total Cost	$655		

Figure 11: Budget Breakdown Table

7 Chapter 5: Professional Issues and Constraints

7.1 Ethical Analysis

When it comes to the implementation of a system like ours into the real world, we have to evaluate the potential ethical ramifications. The main positive impact our technology could have is improving productivity in an agricultural field. We recognize that a more streamlined approach to the agricultural industry could result in better labor conditions for migrant workers when it comes to mitigating disturbances. Additionally, due to the fact that the majority of our project was working on developing algorithms for our demo application, our risk assessment was very small. Hence, our ethical evaluation of our project is at a net positive.

7.2 Science, Technology, and Society

All new and upcoming technologies may be well intentioned but could have unexpected societal ramifications. When it comes to our project, the unexpected societal impact we foresaw is the threat of security breaches. This is not something we had considered in the development in our network, and we recognize the devastating consequences if there ever was to be a security breach on our network connected to our very valuable food production industry.

7.3 Civic Engagement

We foresee that some aspects of our project, if implemented in the field, would need to be approved by the United States Department of Agriculture. Most likely, they will be evaluating the accessibility of this technology, its effectiveness, and also the potential for it to be standardized. We would be advocating for our project in a way that emphasizes the versatility of our communication protocol. The fact that any type of sensor could be used with our devices opens many opportunities within this industry.

7.4 Sustainability

The ongoing threat of climate change puts our world's agricultural industry in a very vulnerable state, so implementing smart farming technologies like ours allows us to detect disturbances early and mitigate them as soon as possible. With our communication protocol, any application in the agricultural field can be made possible such as motion detection, soil temperature, or soil moisture. We believe our technology could help preserve this valuable industry in the face of environmental uncertainty.

8 Chapter 6: Conclusions and Future Work

8.1 Future Work

The next steps for the project would be further fleshing out the messaging and network controls. Currently the network only has the ability to aggregate data and ask if nodes are online. The messages being sent when aggregating are very basic as they only contain the node id, sensor data, and a time stamp. Expanding the criteria for each message and adding more controls to the network would allow it to be used for a wider array of applications. Additionally, further organization of the incoming data would allow easier use of the information once it is aggregated. Beyond expanding the network controls itself, working on a physical implementation that has both physical sensors and a solar power control system would allow the network to be tested in a real world agricultural environment.

8.2 Summary and Conclusions

The problem we solved is the network setup and communication system for an 8-10 node sensor network in order to make it resilient for low power, and applicable for various types of measurements (i.e. soil temperature, soil moisture, humidity). We primarily worked on the software networking component of this system. We built out the network using the Bluetooth mesh protocol, and worked beyond Cypress's desired configurations to build our entirely independent mesh network. We tested our network by simulating different failure scenarios we envisioned would happen in the field. Our end goal was to build the network to be resilient under all of the constraints provided by an agricultural setting, and our network was able to complete these goals and function well with some drawbacks.

9 Bibliography

References

[1] My Newt, Bluetooth Mesh, http://mynewt.apache.org/latest/network/mesh/index.html, Date Accessed: May 27th, 2021

[2] Bluetooth SIG, Bluetooth Mesh Specification, www.bluetooth.com/specifications/adopted-specification

[3] Cypress Bluetooth Mesh https://www.cypress.com/products/airoc-bluetooth-low-energy-mesh

[4] Getting Started with Bluetooth Mesh https://www.infineon.com/dgdl/Infineon-AN227069-Getting-Started-with-Bluetooth-Mesh-ApplicationNotes-v02-00-EN.pdf?fileId=8ac78c8c7cdc391c017d0d36a46266b3

[5] Cypress Bluetooth Mesh https://www.cypress.com/products/airoc-bluetooth-low-energy-mesh

[6] Bluetooth SDK https://infineon.github.io/btsdk-docs/BT-SDK/index.html