MEMORANDUM

To:	Kimberly Lemieux
CC:	ECET Capstone Faculty, Camosun College
From:	Cameron Gillingham, Tella Osler, Aaron Huinink
Date:	October 25, 2024
Subject:	FLoRa Communications Progress Report

SUMMARY

Momentum is building here at FLoRa comms. Our prototype, Petal Radio v0.0, arrived from the manufacturer on October 22nd, and the AVAlink software and firmware development is progressing as expected. We are proud of our progress and can't wait to showcase our protype at the Capstone Symposium on December 13th.

OUR VISION

FLoRa aims to provide an affordable safety communications system outside of cell service to anyone enjoying BC's great outdoors. We are developing a user-friendly network based on *LoRa* radio that anyone with a smartphone can access. Just like the radio in your car, LoRa sends information over the airwaves. In the back country, most hard-core outdoor enthusiasts use an emergency satellite transponder like the Garmin InReach. These devices can run upwards of $800, plus monthly satellite subscription fees! These high costs mean the majority of recreators do not have one when they venture outside of cell service. We are designing an alternative communication network that runs on solar and battery power in remote locations, all for the fraction of the cost of a cell network upgrade.

Our *AVAlink* network is made up of *nodes*, like cell towers, spaced a few hundred to thousand metres apart. Each node has a digital radio we designed called the *Petal Radio*. When someone needs to send a message, they can go to a node and press a button to activate it. The Petal Radio provides a Wi-Fi network that the user connects to with their smartphone. Using a web browser, the user will connect to the AVAlink web app served by our device, and they can send messages to a local chat. This product targets established outdoor organizations who see a large volume of casual and seasoned visitors, such as ski resorts, provincial parks, or trail clubs. Installing our low-cost network adds a layer of redundancy to their safety communication systems that patrons can access in the event of an emergency.

We developed a list of project requirements that will fulfill the needs of our design, the most critical being designing and ordering the Petal v0.0 PCB. We completed this in early

October, and our PCB prototypes arrived on October 22nd. Completing this requirement early was essential to provide plenty of time for testing our hardware and for developing the AVAlink software and firmware, which is where we are now. Our hardware testing is underway (Appendix C – Test Plan), and we have designed an enclosure to hold the Petal during the symposium. Our current work on AVAlink requires designing the web application, developing the firmware, and establishing our LoRa network stack. We are on track to meet the December 13th deadline and are excited to demonstrate our hard work.

PETAL RADIO V0.0

Petal Radio v0.0 is the first PCB prototype for our project (Figure 1). We wanted a board that would work reliably so we could focus on developing firmware and software. It has redundant power supplies, multiple firmware interface options, and accessible inputs and outputs. The first PCBs arrived from JLCPCB on October 22nd, and we are in the process of building and testing their quality.

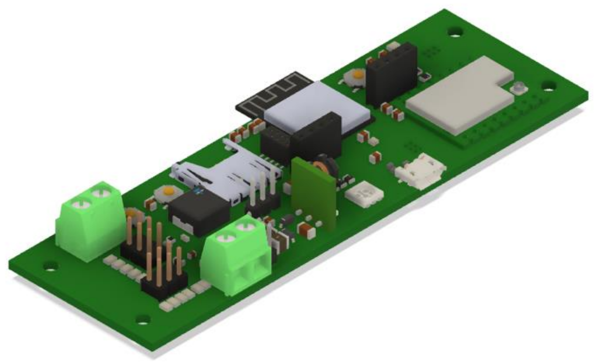


Figure 1: A 3D rendering of the Petal v0.0 PCB

Power Supply

The power supply provides the energy the board needs to operate. This is a critical component, so we included two power supplies on the Petal v0.0. One is a Texas Instruments TPS62933P power supply circuit we designed ourselves, and the other is a prefabricated TPSM84203EAB module. The former is the one we hope to use, and the latter is a backup in case the other doesn't work.

We want to use the TPS62933P because of its low cost and high efficiency, essential for battery-powered devices like the Petal. It also can disconnect the battery to protect it when it runs out of charge. Unlike a prefabricated module, we had to design the complex circuitry around the supply which has many areas where something could go wrong. That is why we also included the TPSM84203EAB. Implementing the module is incredibly easy, but it doesn't offer the same performance, so it exists only as a back up.

Firmware Development

We also included two methods for programming the board in case one doesn't work. The micro-USB port is the main way we hope to upload the board's firmware, but we also included a connection point for a USB-to-serial bridge. These two upload methods ensure we can easily develop and test the AVAlink firmware.

Inputs & Outputs

In case important connections were missed during the design phase, we've also made the unused microprocessor pins accessible at various connection points throughout the board.

ENCLOSURE

Our prototype enclosure will never be used in a permanent outdoor installation and is solely for testing purposes, so we designed a 3D-printed enclosure to hold the Petal during our Symposium demonstrations.

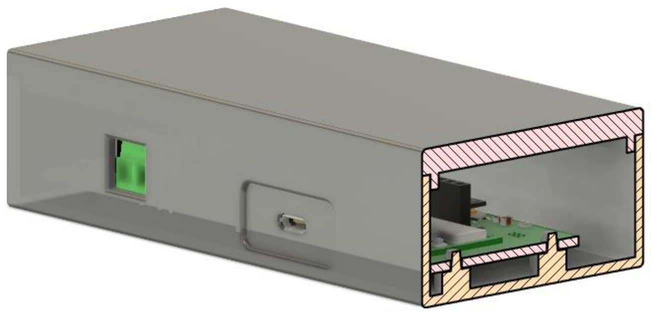


Figure 2: Our 3D printed enclosure designed in Autodesk Fusion

AVALINK

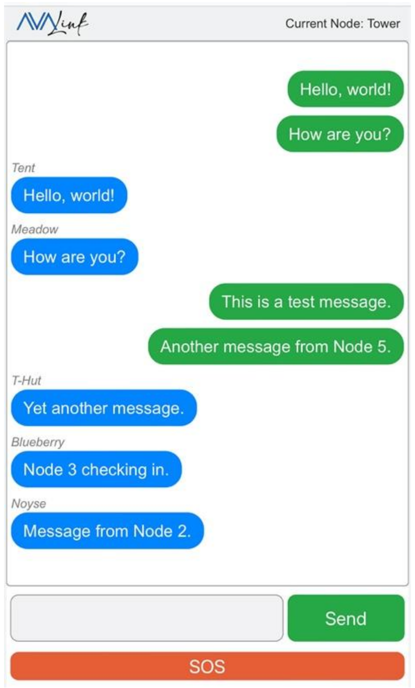
Our focus over the coming weeks will be developing the AVAlink suite of software and firmware that will bring the Petal Radio to life. We are developing a user-friendly web-based messaging application, a robust firmware for the Petal Radio, and a comprehensive network stack that integrates the web application with the rest of the LoRa network.

Web Application

Our web application is the interface between users and the rest of the network. After choosing a framework, we designed it to be as user-friendly and intuitive as possible by choosing colours, shapes, and a layout that resembles familiar smartphone messenger applications (Figure 3). Development has been completed besides the SOS button functionality.

Choosing a Framework

Initially, we chose the Blazor framework [2] to speed up development time, but it came at the cost of very large application files. We found these were too large to be reliably served by our microprocessor, so we pivoted to a more traditional approach. Writing the web application using only HTML, CSS, and vanilla JavaScript creates file sizes several orders of magnitude smaller than a comparable app written with Blazor. In our testing we were able to store these small static files on a SD card and reliably serve them to a user's browser.



Hello, world!
How are you?
Tent
Hello, world!
Meadow
How are you?
This is a test message.
Another message from Node 5.
T-Hut
Yet another message.
Blueberry
Node 3 checking in.
Noyse
Message from Node 2.

Figure 3: Current layout of AVAlink web application, showing a conversation between multiple nodes with different names.

Functionality

We've designed a public channel where all nodes can post and read messages. When a user connects to *http://avalink.local/* on our AVAlink Wi-Fi network, the app loads the layout, fetches the chat history from the server, and populates the chat window (Appendix A – Block Diagrams, Figure 7) The user can send messages to the rest of the network using the chat box. We will implement the SOS button functionality over the next few weeks.

Layout

The sent and received messages are displayed on the screen in chronological order. As shown in Figure 3, messages posted by the node the user is connected to are on the right of the screen in a green bubble, and messages received are on the left in blue bubbles and stamped with the node they came from.

Firmware

The AVAlink firmware will provide the low-level hardware control users require to access our network. After deciding between C++ and Rust for our firmware programming language, we designed detailed flowcharts describing the firmware's operation (Appendix A – Block Diagrams), and now we are implementing these processes in code.

Data will arrive from both the LoRa network and the web server at any time, so these two interfaces will need to operate independently of each other and share time on the MPU. We divided the LoRa and web server processes into two separate *tasks* – a technical term for processes that can run independently of each other. These tasks will be managed by a *task scheduler* – a program that allocates the processing power, preventing tasks from conflicting with each other. This task scheduler will be built using a framework called *FreeRTOS*, a very popular choice in small-processor software development.

An Adventure in Rust

Originally, we hoped to use the Rust programming language to develop our firmware because of its memory safety guarantees, but shortly after beginning development we pivoted to C++. We started our firmware development by writing a small web server in both languages and compared the development experience. C++ gave us much faster development time because there were more open-source libraries for our specific applications in C++. We decided the additional memory safety and easy tooling provided by Rust wasn't worth the added time and effort to develop some necessary libraries from scratch.

Web Server

The web server task can serve the web app quickly and reliably using the files on the Petal's SD card. Messages can be sent and received between different users connected to the same Petal node. Next, we will implement the ability to store and fetch message history from the SD card and to send and receive messages from the LoRa network.

LoRa Network

We've designed our LoRa P2P layer (see Layer 2 – LoRa 'Peer to Peer' (P2P) below). Next, we will implement it in the LoRa task.

FreeRTOS Task Scheduler

The task scheduler will prevent the webserver task from getting in the way of the LoRa task, and vice-versa. It will also provide methods for network messages to be passed between them. We have designed a flow chart describing how the task scheduler will work (Appendix A – Block Diagrams, Figure 8), and development begins this week.

LoRa Network Stack

A robust stack is essential for communications in any network. The network stack determines the how, when, and where for sending and receiving data. A stack is made up of *layers*, or protocols that determine how, when, and where data is sent and received. As shown in Figure 4, data in the network is managed by passing it from layer to layer, with received data moving *up* the stack (from lower layers to higher layers), and sent data moving *down* the stack (from higher layers to lower layers). AVAlink firmware will handle Layer 2 and Layer 3, and it will also handle passing the data between layers.

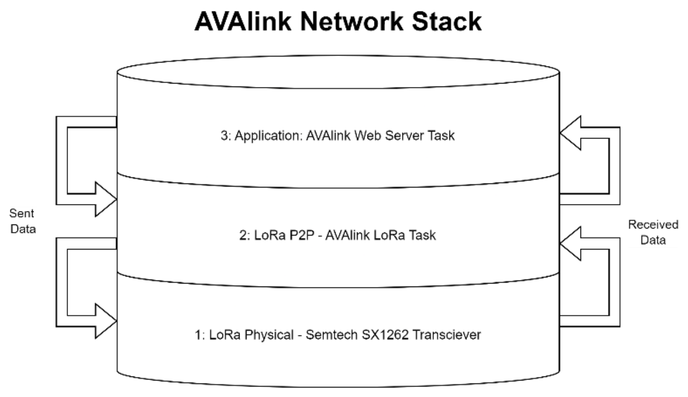

AVAlink Network Stack
3: Application: AVAlink Web Server Task
2: LoRa P2P - AVAlink LoRa Task
1: LoRa Physical - Semtech SX1262 Transciever
Sent Data
Received Data

Figure 4: The layers of the AVAlink network stack.

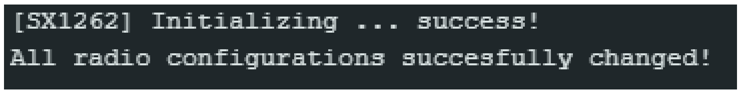
Layer 1 – LoRa Physical

The foundation of LoRa communication is the physical layer, which are the radio waves that carry our data between Petal nodes. The Semtech SX1262 radio transceiver on the Petal Radio handles this layer for us, so we do not need to implement it in AVAlink.

Layer 2 – LoRa 'Peer to Peer' (P2P)

The second layer will be handled by our MPU, so we have designed the protocol for this layer (Appendix A – Block Diagrams, Figure 10). This layer is responsible for reading the messages from the LoRa transceiver, sending them to the web server or storing them for later, and then repeating the message to the next node. This layer will also make sure that messages are transmitted reliably through the network by reducing the likelihood of *collisions*, which is when two nodes try to talk to each other (or to a third node) at the same time.

To prepare for the arrival of our prototyping boards we have successfully tested configuring (Figure 5) and sending messages between two SX1262 transceivers. We will be implementing our Layer 2 protocol into the LoRa task code in the coming weeks.



```
[SX1262] Initializing ... success!
All radio configurations succesfully changed!
```

Figure 5: A successful radio configuration test.

Layer 3 – Application

The web server task will handle the data in Layer 3. It can currently handle multiple client connections within this layer, and we will be interfacing it with the LoRa P2P layer in the coming weeks.

CONCLUSION

The AVAlink network will provide an affordable backcountry communication option accessible by everyone. The Petal Radio v0.0 has arrived from the manufacturer, and we have begun assembling and testing it. Our software and firmware development continues as scheduled bringing our network to life. FLoRa Communications is excited to continue working with you to make the local outdoors a safer place. We invite you to join us when we unveil the AVAlink network on December 13th at the 2024 Camosun College Capstone Symposium!

REFERENCES

[1] “PCB Prototype & PCB Fabrication Manufacturer - JLCPCB.” Accessed: Oct. 16, 2024. [Online]. Available: https://jlcpcb.com/
[2] “Blazor | Build client web apps with C# | .NET.” Accessed: Oct. 18, 2024. [Online]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor
[3] “What is web socket and how it is different from the HTTP?,” GeeksforGeeks. Accessed: Oct. 19, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/
[4] “ESP32-S3 Wi-Fi & BLE 5 SoC | Espressif Systems.” Accessed: Oct. 18, 2024. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-s3
[5] “What is FreeRTOS? - FreeRTOS™.” Accessed: Oct. 18, 2024. [Online]. Available: https://freertos.org/Why-FreeRTOS/What-is-FreeRTOS
[6] J. Gromeš, *jgromes/RadioLib*. (Oct. 19, 2024). C++. Accessed: Oct. 18, 2024. [Online]. Available: https://github.com/jgromes/RadioLib
[7] “RadioLib: PhysicalLayer Class Reference.” Accessed: Oct. 18, 2024. [Online]. Available: https://jgromes.github.io/RadioLib/class_physical_layer.html#af9a7e739e39705a72ffa8b63ec09bb15
[8] “Introduction | Meshtastic.” Accessed: Oct. 18, 2024. [Online]. Available: https://meshtastic.org/docs/introduction/

APPENDIX A – BLOCK DIAGRAMS

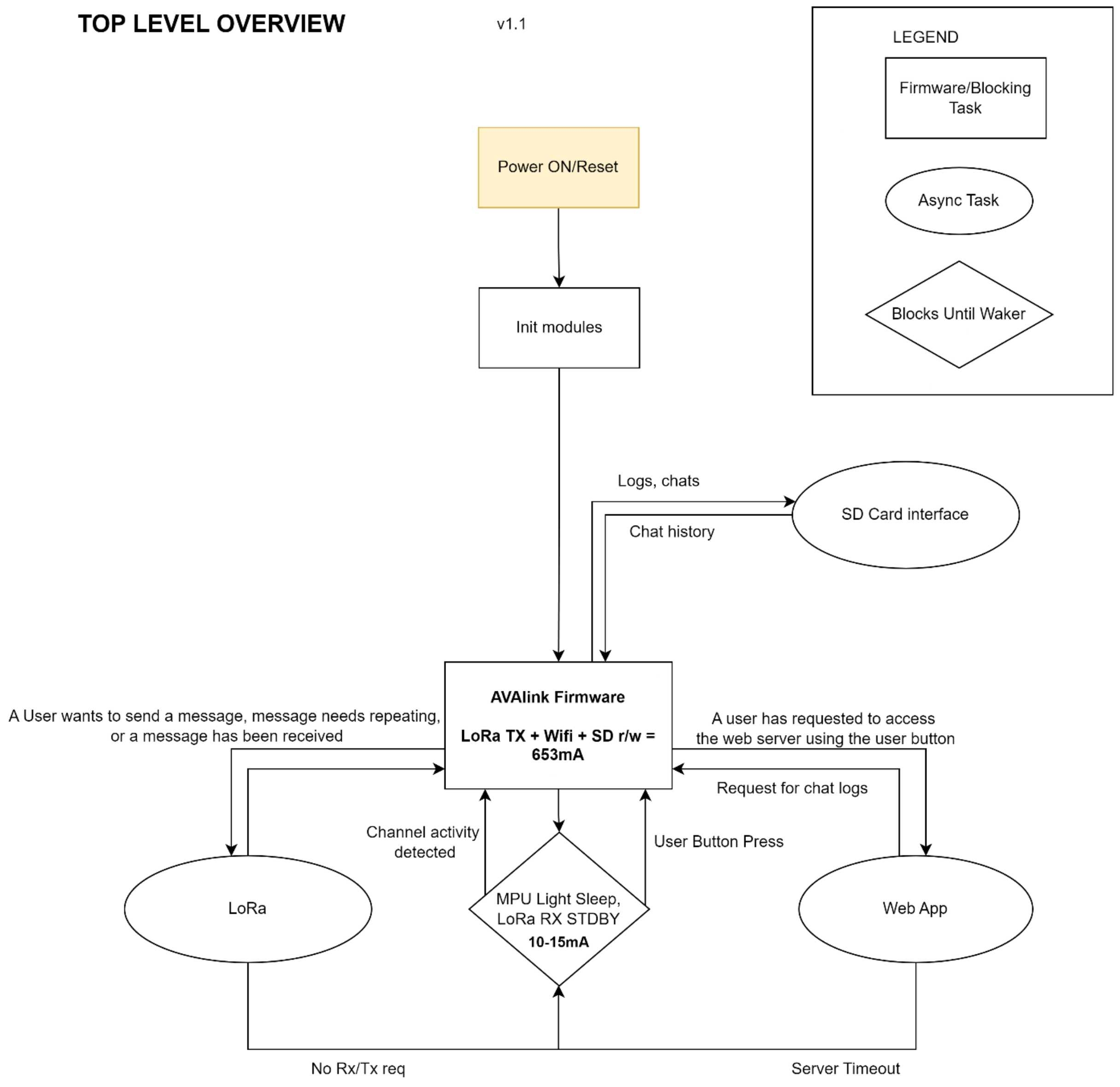
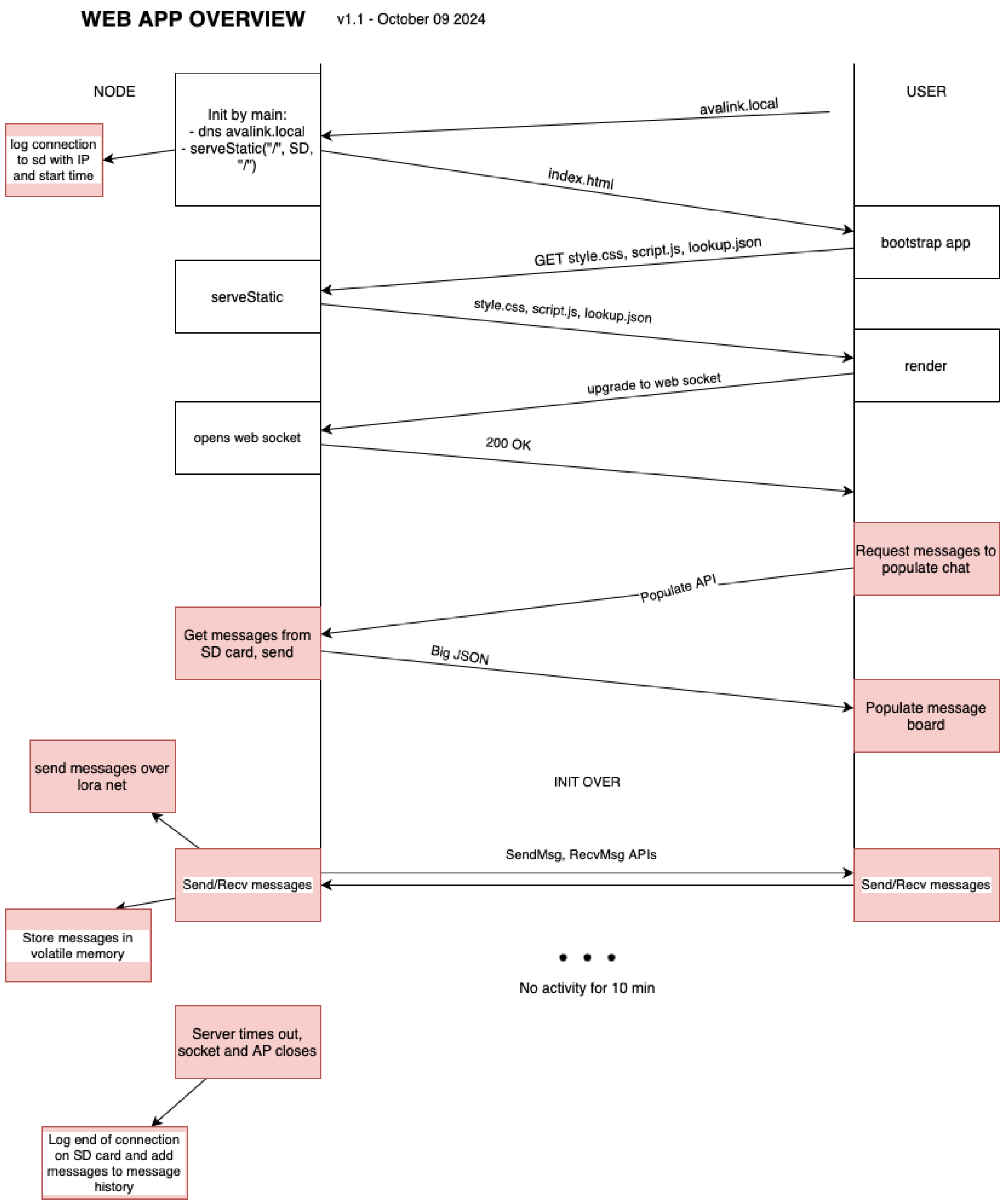

TOP LEVEL OVERVIEW
v1.1
LEGEND
Firmware/Blocking Task
Async Task
Blocks Until Waker
Power ON/Reset
Init modules
Logs, chats
Chat history
SD Card interface
AVAlink Firmware
LoRa TX + Wifi + SD r/w = 653mA
A User wants to send a message, message needs repeating, or a message has been received
A user has requested to access the web server using the user button
Request for chat logs
Channel activity detected
User Button Press
LoRa
MPU Light Sleep, LoRa RX STDBY 10-15mA
Web App
No Rx/Tx req
Server Timeout

Figure 6: AVAlink firmware top level overview.


WEB APP OVERVIEW
v1.1 - October 09 2024
NODE
USER
log connection to sd with IP and start time
Init by main: - dns avalink.local - serveStatic("/", SD, "/")
avalink.local
index.html
bootstrap app
GET style.css, script.js, lookup.json
serveStatic
style.css, script.js, lookup.json
render
upgrade to web socket
opens web socket
200 OK
Request messages to populate chat
Populate API
Get messages from SD card, send
Big JSON
Populate message board
send messages over lora net
INIT OVER
SendMsg, RecvMsg APIs
Send/Recv messages
Send/Recv messages
Store messages in volatile memory
No activity for 10 min
Server times out, socket and AP closes
Log end of connection on SD card and add messages to message history
Figure 7: Web app deployment sequence

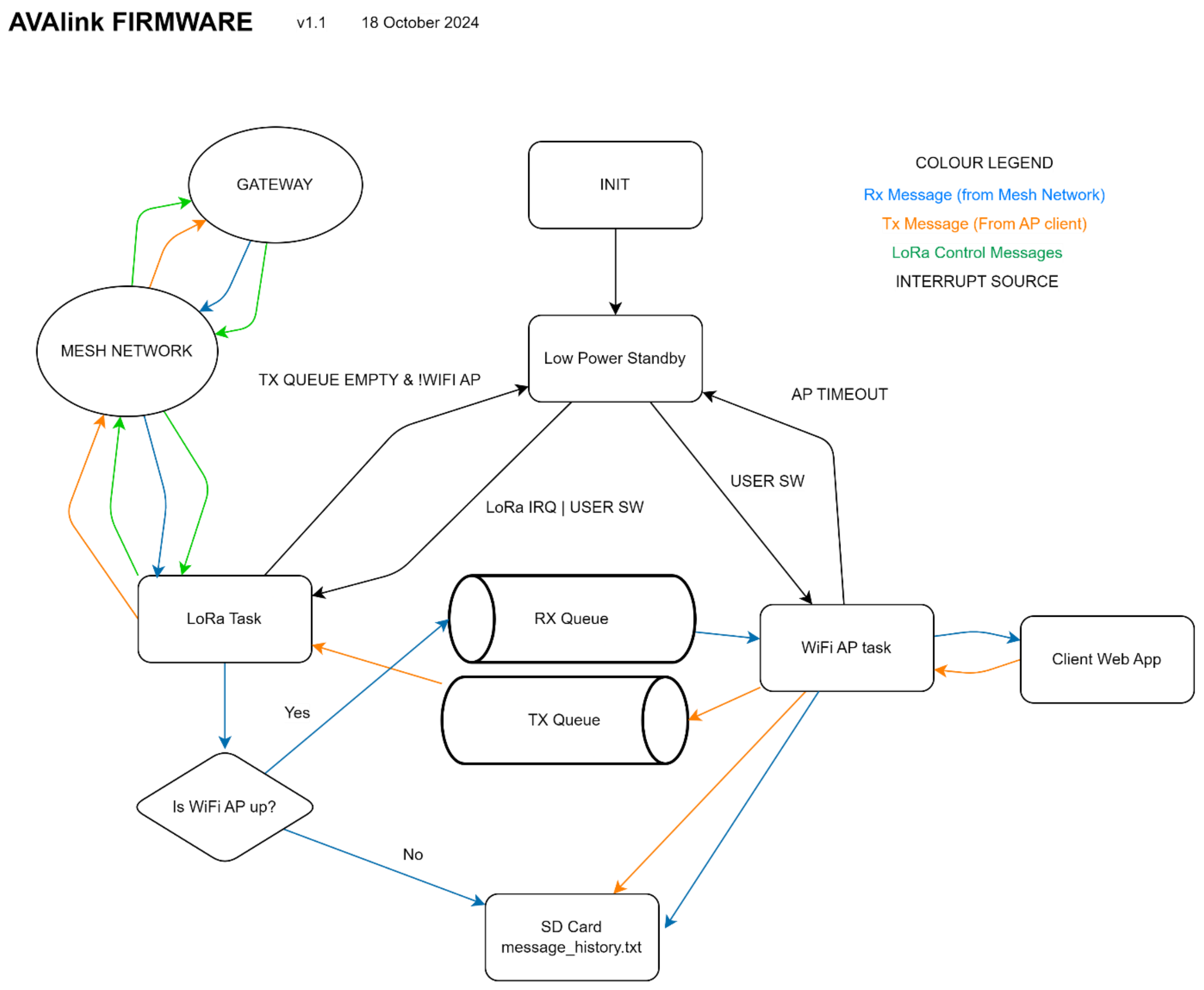


Figure 8: Firmware decision matrix

LORA OVERVIEW

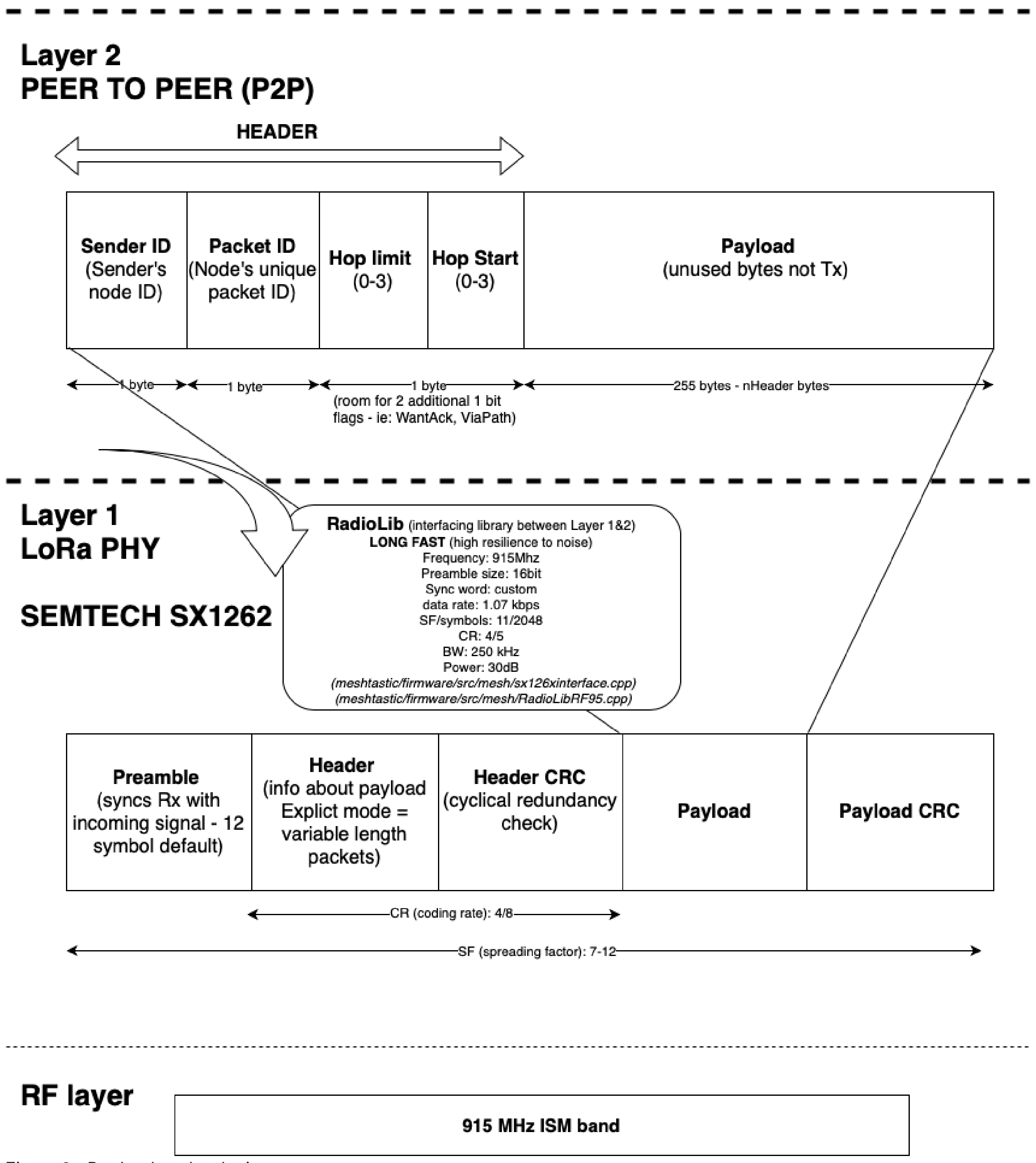

Layer 2
PEER TO PEER (P2P)
HEADER
Sender ID (Sender's node ID)
Packet ID (Node's unique packet ID)
Hop limit (0-3)
Hop Start (0-3)
Payload (unused bytes not Tx)
1 byte
1 byte
1 byte
(room for 2 additional 1 bit flags - ie: WantAck, ViaPath)
255 bytes - nHeader bytes
Layer 1
LoRa PHY
SEMTECH SX1262
RadioLib (interfacing library between Layer 1&2)
LONG FAST (high resilience to noise)
Frequency: 915Mhz
Preamble size: 16bit
Sync word: custom
data rate: 1.07 kbps
SF/symbols: 11/2048
CR: 4/5
BW: 250 kHz
Power: 30dB
(meshtastic/firmware/src/mesh/sx126xinterface.cpp)
(meshtastic/firmware/src/mesh/RadioLibRF95.cpp)
Preamble (syncs Rx with incoming signal - 12 symbol default)
Header (info about payload Explict mode = variable length packets)
Header CRC (cyclical redundancy check)
Payload
Payload CRC
CR (coding rate): 4/8
SF (spreading factor): 7-12
RF layer
915 MHz ISM band

Figure 9 - Packet header design

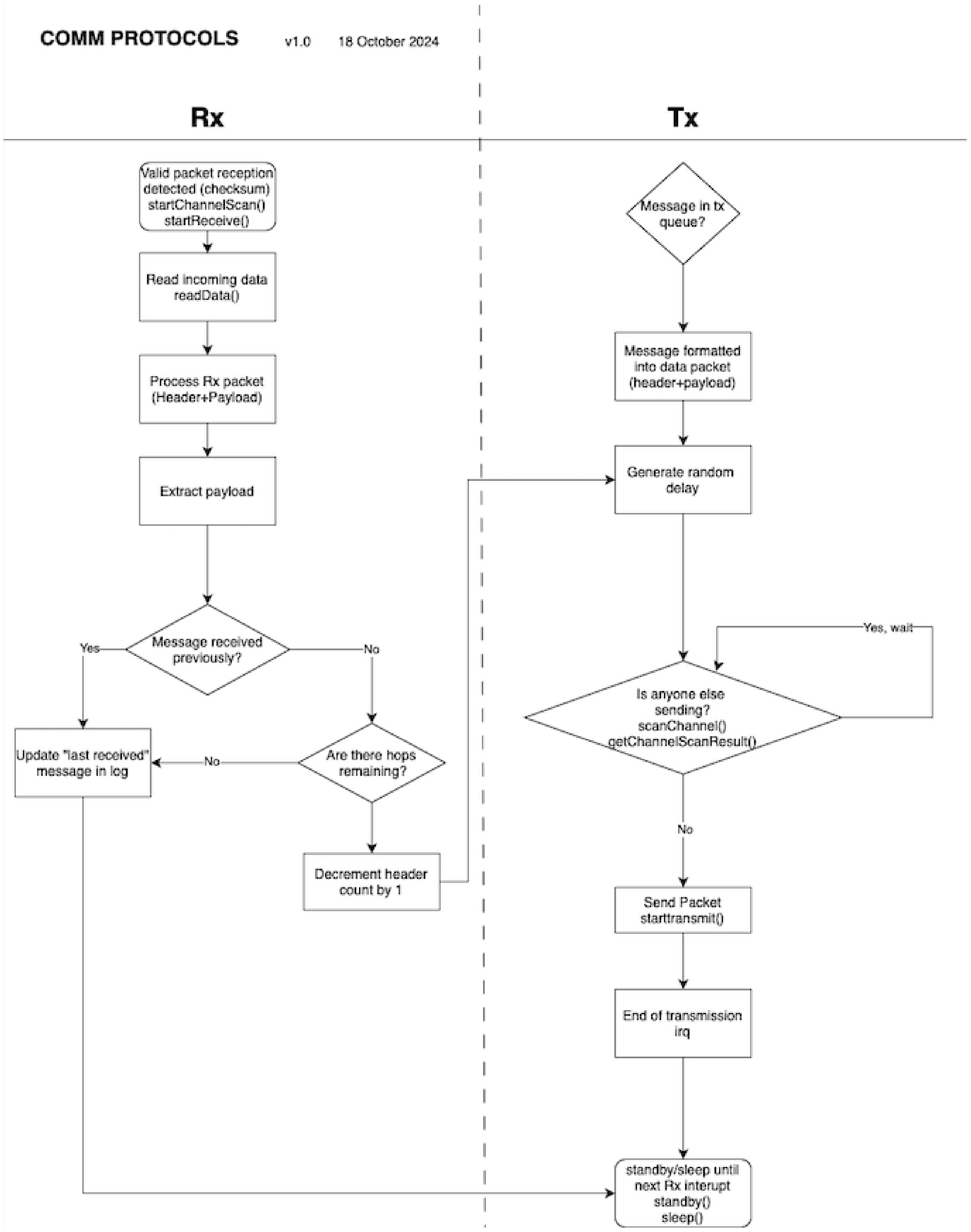

COMM PROTOCOLS
v1.0
18 October 2024
Rx
Tx
Valid packet reception detected (checksum) startChannelScan() startReceive()
Read incoming data readData()
Process Rx packet (Header+Payload)
Extract payload
Message received previously?
Yes
No
Update "last received" message in log
Are there hops remaining?
No
Decrement header count by 1
Message in tx queue?
Message formatted into data packet (header+payload)
Generate random delay
Is anyone else sending? scanChannel() getChannelScanResult()
Yes, wait
No
Send Packet starttransmit()
End of transmission irq
standby/sleep until next Rx interupt standby() sleep()

Figure 10: LoRa P2P layer protocol.

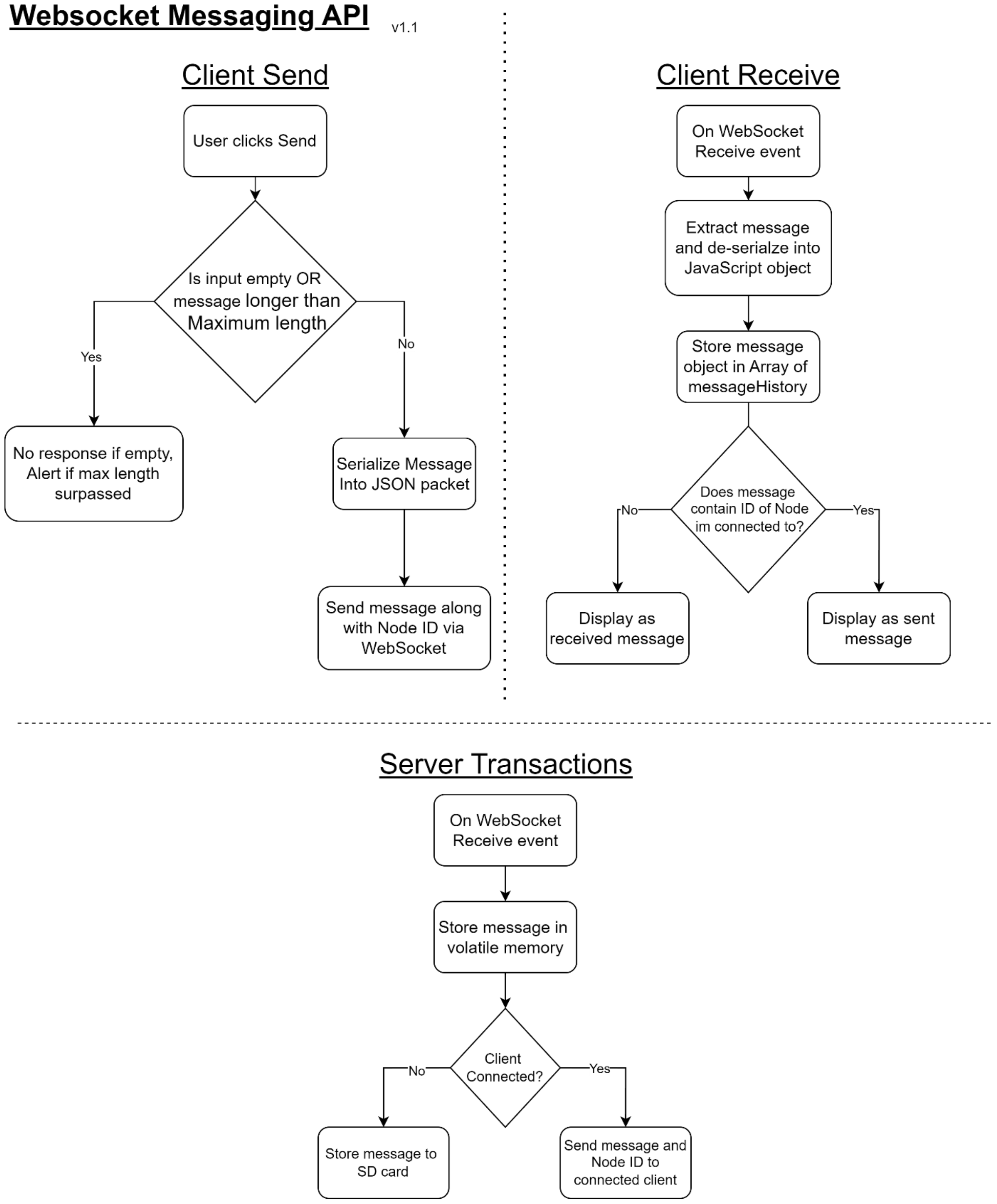

Websocket Messaging API
v1.1
Client Send
User clicks Send
Is input empty OR message longer than Maximum length
Yes
No
No response if empty, Alert if max length surpassed
Serialize Message Into JSON packet
Send message along with Node ID via WebSocket
Client Receive
On WebSocket Receive event
Extract message and de-serialze into JavaScript object
Store message object in Array of messageHistory
Does message contain ID of Node im connected to?
No
Yes
Display as received message
Display as sent message
Server Transactions
On WebSocket Receive event
Store message in volatile memory
Client Connected?
No
Yes
Store message to SD card
Send message and Node ID to connected client

Figure 11: Application layer protocols.

APPENDIX B – TABLES

Table 1- Capstone requirements

Reference Number	Requirement	Test	Pass/Fail Criteria
1-SW	R1 – The user interface MUST be accessible through a modern web browser without the need for a separate application.	T1.0 – Connect a smartphone with a modern browser to the node and scan the web server QR code. T1.1 – Connect a PC to the node and navigate to the web server page.	C1.0 – UI is rendered and readable on a mobile device. C1.1 – UI is rendered and readable on a PC.
2-SW	R1 – Users MUST be able to send LoRa packets using the web interface to a public forum-like chat that can be viewed from another node. F1 – Users can send chats between specific nodes that are not publicly visible to all nodes.	T1 – Send a message using the UI to another node	C1 – The sent message is shown on all other devices' UIs. FC1 – The sent message is only viewable on the intended receiver's UI.
3-SW	R1 – Chats on UI must have a username or device identifier and a timestamp of when the message was sent.	T1 – Send chat messages from multiple devices over the UI	C1 – The UI renders chat history with usernames in chronological order according to message timestamp
4-SW	R1 – The repeater node MUST monitor battery voltage and disconnect when dropping below the low voltage threshold. The repeater node	T1 – Supply voltage to the node with a variable power supply and document which voltages result in disconnect and reconnect. The node	C1.0 – The reported battery voltage is accurate within 3%. C1.1 – The load disconnects when the battery voltage drops below the low

	low-voltage disconnect MUST implement hysteresis to prevent power cycling. F1 – The repeater SHOULD indicate to the rest of the mesh network that it is powering down.	voltage monitor will be compared to that of the power supply and measured with an external meter.	voltage threshold and turns back on when the battery charges above threshold voltage. The implemented hysteresis prevents power cycling of the device. FC1 – Before the low-voltage disconnect, the device transmits an alert that it is powering down.
5-SW	R1 – The software MUST implement a collision avoidance or multiple access protocol that deals with the hidden-node problem	T1 – Transmit a LoRa packet from two devices to a single receiver at the same time without a connection between the two senders to coordinate between them	C1.0 – Neither message is lost, or corrupted. C1.1 – Messages are displayed in the UI in the correct order based on their timestamp
6-HW	R1 – MUST Design and order a PCB	T1 – Before each revision is submitted for manufacturing, it will be subject to an internal review by the group and an external review by the Capstone Committee.	C1.0 – The PCB design passes an internal review process and review from the Capstone Committee. C1.1 – Each revision passes our hardware testing suite.
7-HW	R1 – MUST have a bespoke enclosure. F1 – SHOULD be protected against rain and moisture ingress	T1.0 – The enclosure will be inspected by professors. T1.1 – Third-party parts like cable glands are IPX4 certified.	C1 – All materials have IPX4 or greater certification from reputable lab FC1 – No water ingress is present after the IPX4 test.

		FT1 – The enclosure is subjected to the standard IPX4 test	
8-HW	R1 – Voltage regulator MUST effectively provide the required 3.3V to the hardware for a range of typical battery voltages.	T1 – Sweep the input voltage across the range defined by the regulator datasheet and measure the voltage regulator output while it is loaded with the expected full load current the PCB requires.	C1.1 – The hardware receives a stable 3.3V +/- 0.1V out across the range of test voltages while under the expected load. C1.2 – The regulator operates as expected according to the manufacturer's datasheet.
9-HW	R1 – MUST provide recommendations for sizing batteries and solar panels based on expected insolation.	T1 – Use recommendations to size solar and battery power for a mock installation at Camosun College using insolation data for that location.	C1 - Recommended panel wattages and battery Ah meet or exceed node requirements as calculated by our power audit (datasheet specifications, duty cycle, solar insolation modeling)
10-HW	R1 – Antennas MUST be well matched to the driving hardware	T1 – SWR/Impedance testing of antenna and source using VNA (may require tuning to meet these requirements)	C1 – Source impedance is matched to antenna so that VSWR < 2 and return loss < -10 dB
11-HW	R1 – Prototype nodes MUST incorporate an accessible user button at access points for users to initiate the web server.	T1 – Check that the Wi-Fi access point is powered down. Press the user button to initiate the Wi-Fi access point.	C1 – The Wi-Fi access point is available after pressing the user button. C2 – The Wi-Fi access point is

	R2 – The Wi-Fi access point MUST time-out after 5 minutes of inactivity to save power.	T2 – Leave the access point for 5 minutes without activity.	disabled after 5 minutes of inactivity.
12- SW	R1 – The firmware has multiple modes: Passive: The MPU is sleeping and listening for new messages. It will act as a repeater. Message Available: The MPU is in passive mode, but it has saved new messages for the next user to view Active: The MPU is powered up and advertising the Wi-Fi access point while continuing to act as a LoRa mesh node. Low-Power: The low voltage disconnect has taken the node offline until the battery can be recharged. F1 – An RGB LED indicates the state the hardware is in with different colours (Passive, Message Available, Active, Waiting, Low battery). F2 – An SOS mode that activates across the entire network. It connects directly to emergency services, minimizes network	T1.0 & FT1.0 – Send a message from an Active node to a Passive node. T1.1 – Receive a message repeated by a Passive node. T1.2 – Test the current draw in passive mode. FT1.1 – Input a voltage lower than the low-voltage disconnect but larger than the minimum voltage required by the linear voltage regulator.	C1.0 – The message is available when the Passive node is powered up later. C1.1 – The message is received at the active node. C1.2 – The current draw in passive mode is less than the current draw in active mode. FC1.0 – The LED indicates a New Message state. FC1.1 – The LED indicates low voltage.

	latency at the expense of power efficiency, and disables the low voltage disconnect.		

APPENDIX C – TEST PLAN

ACCESSIBILITY

Testing the accessibility of the AVAlink service for users. Covers all of 1-SW.

Mobile Devices

Test the accessibility of the user interface on a mobile device.

1. Provide power to a Petal Radio using the micro-USB or an appropriate battery configuration.
2. Have a friend/parent follow these instructions without assistance:
    a) Press the User button to activate the node.
    b) Using a smartphone, connect to the AVAlink Wi-Fi network.
    c) Scan the QR code or navigate to *avalink.local* in a web browser **(CHECKPOINT)**.

Pass/Fail

Criteria for user accessibility requirements. The order of the below criteria corresponds to the order that checkpoints appear in the testing steps.
1-SW-C1.0. The user interface is rendered correctly and readably on a mobile device.

PCs

Test the accessibility of the user interface on a PC/desktop computer.

3. Provide power to a Petal Radio using the micro-USB or an appropriate battery configuration.
4. Have a friend/parent follow these instructions without assistance:
    a. Press the User button to active the node.
    b. Using a desktop or laptop computer, connect to the AVAlink Wi-Fi network.
    c. Navigate to *avalink.local* in a web browser **(CHECKPOINT)**.

Pass/Fail

Criteria for user accessibility requirements. The order of the criteria below corresponds to the order that checkpoints appear in the testing steps.
1-SW-C1.1. The user interface is rendered correctly and readably on a desktop/laptop computer.

LORA NETWORKING

Tests the LoRa networking capabilities. Covers all of 2-SW, 3-SW, 5-SW, 10-HW.

LoRa Tx/Rx & Collision Avoidance

Purposefully introduce the potential for packet collisions to test the performance of the collision avoidance protocol wde designed for the AVAlink network.

1. Connect a smartphone or PC to each node (Mobile Devices).
2. Send a message in the web application from both nodes with at least a 30 second delay between pressing send on the applications **(2 CHECKPOINTS).**
3. Next, introduce a packet collision scenario. Enter a message into the text box of both nodes' web applications, but do not press send yet.
4. Once both applications are loaded with a message, press the send button on both node applications at the same time.
5. See that both messages are displayed correctly in the application at both nodes. If one or both messages do not appear, those packets were lost in a collision. Record the success or failure for that packet collision scenario.
6. Repeat Steps 3, 4, and 5 nineteen more times, for a total of 20 potential packet collision scenarios. **(CHECKPOINT)**.

Pass/Fail

The order of the below criteria corresponds to the order that checkpoints appear in the testing steps.
2-SW-C1. The sent messages appear in the other device's web application.
3-SW-C1. The messages appear in the chat box in the order they were sent with formatting and information that indicates which node sent each message (i.e. transmitted messages appear on the right in green and received messages appear on the left in blue with the associated node ID).
5-SW-C1. Out of 20 packet collision scenarios, 3 or less result in lost messages (packet loss of 15%).

Impedance Match

Test the match between the LoRa module and the antenna.

1. Connect the IPEX connector on the LoRa module to a VSWR tap DUT input.
2. Connect a LoRa antenna to the VSWR tap DUT output.
3. Connect a spectrum analyser to the VSWR tap output.
4. Set the spectrum analyser to view the ISM band (900-928MHz) with persistence enabled.
5. Put the node into USB mode following the user manual instructions.
6. Provide 5V power to the node.
7. Put the node into its test configuration* following the user manual instructions.
8. Clear the persistent traces on the spectrum analyser.
9. Press the User button on the node to initiate the test.
10. Upon completion of the test, save the spectrum analyser trace.

11. Note the frequency and peak power of the smallest peak on the analyser output **(CHECKPOINT)**.

*The test configuration transmits LoRa packets at 14dBm at frequency slots across the ISM band.

Pass/Fail

The order of the below criteria corresponds to the order that checkpoints appear in the testing steps.

10-HW-C1. The smallest peak in the ISM band has a peak power less than 4dBm (VSWR < 2).

POWER

Tests the power and battery management capabilities of the repeater node. Covers *4-HW, 8-HW, 9-HW, 11-HW, 12-SW.*

Low Voltage Disconnect

Test the low voltage disconnect capability of the node that protects batteries from full discharge.

1. Use a variable voltage supply that can provide at least 0V-20V and 3A and set the supply to 7V.
2. Follow the instructions in the user manual to set the power supply to the TPS62933P option in 6V Lithium mode.
3. Remove the jumper from J9 to expose the pins.
4. Connect the circuit shown in Figure 12 from the left-most pin of J9 to ground.
5. Connect the power supply to the board at the battery terminal block and power up the board.
6. Measure the input voltage across the battery terminal screw heads and verify it is 7.00V.
7. Measure the output voltage across the load circuit and confirm it is between 3.25V and 3.35V.
8. Place a lead on that pin and continue to monitor the voltage there using an oscilloscope.
9. Set the oscilloscope to single capture, triggering on a falling edge between 3.3V and 0V and capturing at least 20ms after the falling edge.
10. Slowly decrease the input voltage at a rate of ≈10mV/second until the load circuit LED turns off. This is the low voltage disconnect.
11. Record the input voltage that caused the low voltage disconnect in Table 2 in the appropriate column.
12. Save the oscilloscope trace. If the trace shows the load resistor voltage cycling between 0V and 3.3V after the initial power disconnect, put a “y” in the Power Cycling row of Table 2, otherwise put an “n”**.**
13. Repeat Steps 5-12 for the 12V Pb/AGM battery mode and the 12V Lithium mode but have the starting voltages about 0.5V higher than the “Reconnect” target voltages in Table 2 **(CHECKPOINT)**.

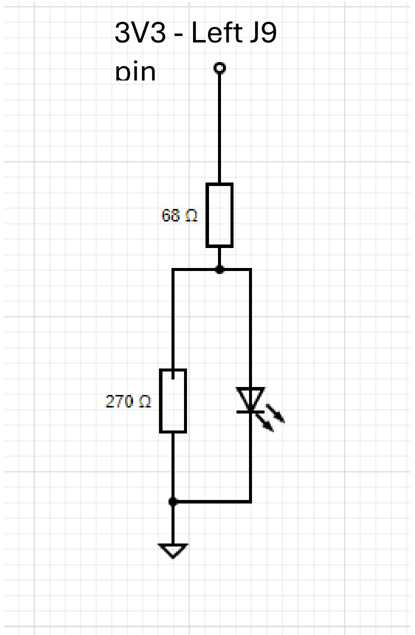


Figure 12: LVD testing load circuit.

Pass/Fail

Complete Table 2 below and compare to the requirement specifications.

Table 2: Low voltage disconnect testing values.

6V Li		
	Disconnect	6
Targets	Reconnect	6.6
Action	*Voltage (V)*	*Pass/Fail*
Disconnect		FAIL
Reconnect		FAIL
Power Cycling	y	FAIL
12V Pb/AGM		
	Disconnect	11.51
Targets	Reconnect	12.06
Action	*Voltage (V)*	*Pass/Fail*
Disconnect		FAIL
Reconnect		FAIL
Power Cycling	y	FAIL
12V Li		
	Disconnect	12
Targets	Reconnect	13
Action	*Voltage (V)*	*Pass/Fail*
Disconnect		FAIL
Reconnect		FAIL
Power Cycling	y	FAIL

4-HW-C1.1. All actions in Table 2 pass.

Power Supply

Tests the TPS62933P switching power supply circuit we designed for the Petal v0.0.

1. Put Petal v0.0 into USB mode by following the user manual instructions.
2. Remove the jumper from J9 to expose the pins.
3. Connect a programmable load to the far left pin of J9 and to ground.
4. Set the load to draw 10mA at 3.3V.
5. Connect a power supply (see "Supply Voltage" for the applicable mode voltage in Table 3).
6. Measure the output voltage across the programmable load and note it in Table 3.
7. Connect an oscilloscope across the programmable load.
8. Measure the ripple voltage across the programmable load with the oscilloscope, record it in Table 3, and save the trace data for later documentation.
9. Increase the load current draw to 650mA.
10. Repeat Steps 6-8.
11. Program the load to quickly increase from 10mA to 650mA, hold for 5ms, and then decrease back down to 10mA.
12. Set the oscilloscope to perform a single capture on an AC-coupled rising edge with a vertical range of -350mV to 350mV and a horizontal range of 10ms.
13. Run the load transient.
14. Using the oscilloscope, measure the peak-to-peak voltage change for the two load transient peaks. Divide this number in half to average the two peaks and record in Table 3.
15. At each transient spike, estimate the time it took for the voltage to correct back to it's typical 3.3V waveform. Record in Table 3.
16. Repeat Steps 4-15 for the remaining modes in Table 3 **(CHECKPOINT)**.

Pass/Fail

Complete Table 3 below and compare to the requirement specifications.

Table 3: Power supply unit testing criteria.

Mode	Supply voltage (V)	Load Current (mA)	Output Voltage (V)	Ripple Voltage (V)	Load Transient Response		Pass/Fail
USB	5V	10			Voltage Change (mV)		FAIL
		650			Correction time (us)		
6V Li	6.7V	10			Voltage Change (mV)		FAIL
		650			Correction time (us)		
12V Pb/AGM	12.6V	10			Voltage Change (mV)		FAIL
		650			Correction time (us)		
12V Li	13.5V	10			Voltage Change (mV)		FAIL
		650			Correction time (us)		

8-HW-C1.0 & C1.1. All modes have a "PASS" grade.

Efficiency

Tests that the firmware switches between power modes based on demands.

1. Put the Petal v0.0 into 6V Li mode following instructions from the user manual.
2. Connect a 7V power supply to the board through an ammeter.
3. Ensure that the Wi-Fi access point is not on.
4. Record the current drawn in this standby mode.
5. Press the User Switch to initiate the Wi-Fi access point.
6. Ensure the Wi-Fi access point is up **(CHECKPOINT)**.
7. Record the current drawn in this state.
8. Wait 5 minutes.
9. Ensure the access point has turned off **(CHECKPOINT)**.
10. Measure the current drawn by the board again **(CHECKPOINT)**.

Pass/Fail

Criteria for efficiency requirements. The order of the below criteria corresponds to the order that checkpoints appear in the testing steps.

11-HW-C1. The access point is available.

11-HW-C2. The access point turns off after 5 minutes of inactivity.

12-HW-C1. The current draw when the Wi-Fi access point is up is greater than the current draw when the node is in a standby mode