Lab: LoRa

Introduction

The purpose of this lab is to get you some hands-on experience with LoRa on a microcontroller. We'll focus on directly using the physical layer: LoRa, although we'll use some configurations from the LoRaWAN protocol.

Since we have control over low-level packet transmissions, this is also an opportunity to design our own protocol on top of LoRa. Groups will implement a network stack that allows data to be communicated between low-energy end-devices.

For this lab we will again be writing code using PlatformIO, and we'll again be working on the Heltec WiFi LoRa 32 v3 boards. We'll use different libraries to enable LoRa communication.

Goals

- Communicate between devices over LoRa
- Experiment with the capabilities of wide-area communication
- Design and implement a custom communication protocol

Equipment

- Computer
- Heltec WiFi LoRa 32 v3 + USB cable (3 total for the group)

Github Classroom

- https://classroom.github.com/a/ir3raOmY

Partners

- This lab should be done with **your group of three**

Submission

- Write your answers up for each task and submit a PDF to Gradescope.

  Remember: I'm not looking for a formal lab report. Just your answers in any format that makes sense. The goal is to prove that you did the lab and spent some time thinking about it.

Table of Contents

Introduction

Table of Contents

List of Tasks

1. Hardware Requirements

2. Software Requirements

3. RadioLib Library

4. LoRaWAN Documentation

5. Basic Communication

6. Range Testing

7. Find My Signal

8. Protocol Design and Implementation

List of Tasks

- Section 5.1: Which LoRaWAN configurations did you choose?
- Section 5.2: Demonstrate ability to communicate between two devices

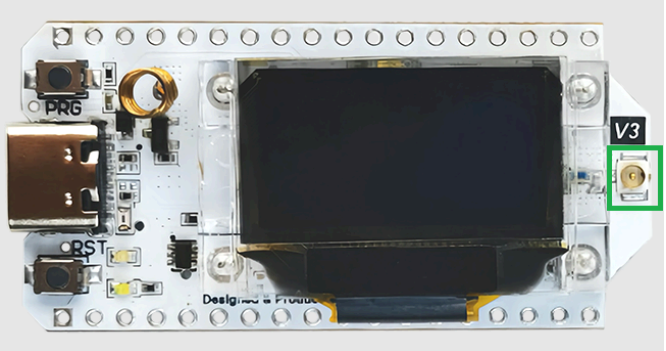
- Section 6.1: How did you maximize range?
- Section 6.2: What is the maximum range you achieved?

- Section 7.1: Find my signal
- Section 7.2: Submit your scanner application code

- Section 8.1: Describe your protocol design
- Section 8.2: Demonstrate your full application

1. Hardware Requirements

To communicate over LoRa, you must attach the antenna to your Heltec board.



- Attach the antenna cable to the U.FL connector on the Heltec WiFi LoRa 32 v3 board. The connector is the gold one near the “V3” label on the opposite end of the board from the USB connector. It is highlighted in green in the image above.

  You have to line it up and then push it on. It’ll click when it connects and should be able to rotate without disconnecting.

- Then screw the antenna onto the cable. The order doesn’t matter, but it’s easier to attach the cable without a big antenna on it.

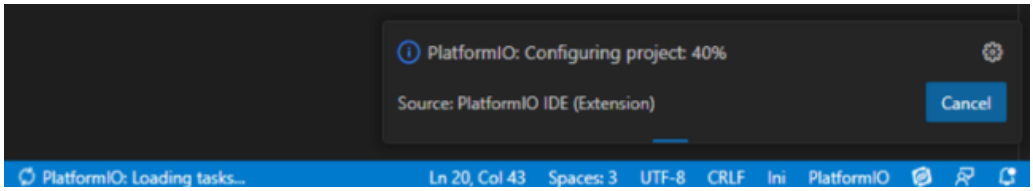
- **IMPORTANT:** do not remove the antenna cable once connected. U.FL connectors are not intended for disconnection and are only rated for about 30 plug/unplug operations before they break. You may, however, unscrew the antenna from the cable.

2. Software Requirements

This lab will require writing code with PlatformIO again. **These instructions are repeated from the previous lab in case you run into bugs and they're useful.**

- Follow the Github classroom link to create or join your own private repo of starter code:https://classroom.github.com/a/ir3raOmY

- The first time you open a folder in that repo with PlatformIO, expect a lot of loading to occur. It needs to install an entirely new toolchain to compile and upload code for ESP-32 microcontrollers. On my desktop, this took about five minutes to complete. You'll see something like this:



- When you plug in the board for the first time, check that orange LED by the "RST" button lights up. That means the board has power.

  A weird quirk of this hardware is that it MUST have a USB adapter somewhere inline when you plug it in. Plugging in directly via a USB-C to USB-C cable fails to power the board (the implemented something about that incorrectly…). Each box should have a USB-C to Micro USB adapter, which you can use in conjunction with the USB micro cable you already have from previous labs.

- There's some small, but non-zero chance you need to install a USB driver for the board. It worked fine for me on one of my Windows computers, but not the other. I needed to install the "CSP210x Windows Drivers" (not the Universal one), and then it worked. https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

  Even after that, you might get a very cryptic error on Windows, about not having access to the COM port. A reboot resolves this issue. Linux and MacOS should hopefully be fine, but let me know if you have issues.

  I've also sometimes found that commands fail the first time, but succeed the second time. It didn't happen too often, but be willing to try it if things are going wrong.

3. RadioLib Library

Our Heltec board has an SX1262 LoRa transceiver connected to the ESP32 microcontroller over SPI. We'll be programming the ESP32 with code that communicates with the transceiver to transmit/receive LoRa packets.

RadioLib is a **very** high quality Arduino library for LoRa communication. It provides high-level radio commands, which underneath send SPI commands to the specific transceiver used. It also implements several link-layer protocols, but we will not be using those.

Documentation:

- RadioLib code: https://github.com/jgromes/RadioLib
    - Occasionally it's useful to look through the code. Although VSCode "Go to Declaration/Definition" options can do this well too.

- RadioLib SX126x documentation: https://jgromes.github.io/RadioLib/class_s_x126x.html
    - Generated from the same doc comments that the VSCode pop-ups are. So those are probably sufficient, but it's nice to see a written list of functions somewhere.

- RadioLib SX126x examples: https://github.com/jgromes/RadioLib/tree/master/examples/SX126x
    - Receive_interrupt and Transmit are definitely the most useful
    - Settings is useful for understanding how to configure things
        - Although, be warned that some of the comments are incorrect for the SX1262 radio, likely copied from other radio example code.

Reminder: for this particular class, **you are allowed to use code you find online** as long as you cite it. There isn't as much of a plagiarism concern here since our goal is really to learn about networks, not write software. Citing where the code came from is a good practice for others who might have to understand your own code.

4. LoRaWAN Documentation

We'll mostly just be using the raw LoRa protocol, and not the entire LoRaWAN network stack. However, whenever possible we will rely on LoRaWAN configurations for communication in the US.

- LoRaWAN Regional Parameters v1.0.3
  https://hz137b.p3cdn1.secureserver.net/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf
  (Not sure if that's a long-term stable link. If not, you can find it here: https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/)
    - Sections that could be useful here:
        - Section 2.5: US902-928 MHz ISM Band
            - Section 2.5.2: Channel Frequencies
            - Section 2.5.3: Data Rate and End-device Output Power encoding
            - Section 2.5.6: Maximum payload size
        - Section 4.1: Physical layer, LoRa description

- **WARNING:** you MUST always respect the FCC limitations for communication in the 902-928 MHz ISM band.
    - **Power:** The power limitations are not a concern as the maximum SX1262 transmit power is below the limit.

    - **Duration:** Maximum transmission duration of 400 ms **is** important though.

      Do not use a spreading factor above 10, and do not transmit more than 47 bytes in a packet payload. See sections 2.5.3 and 2.5.6 of the LoRaWAN Regional Parameters document. (The 19 byte limit there accounts for 28 bytes of LoRaWAN header overhead that we don't have.)

    - **Transmission Rate:** If transmitting for a long period of time (more than a few minutes) on a single channel, you should only transmit one packet per 20 seconds maximum. The FCC dwell time rule in full is 400 ms per 20 seconds.

      When doing initial testing or sporadically sending packets when pushing a button, this limitation is not so important. Just make sure you're not incidentally "jamming" the channel.

    - **Frequency:** Only transmit on the US channels as defined in section 2.5.2 of the LoRaWAN Regional Parameters document. You can transmit on the upstream or downstream channels, but staying on one of the sixty-four 125 kHZ uplink channels makes sense to start off.

5. Basic Communication

Let's start by sending and receiving packets between two devices.

- Create an app that is always receiving and can transmit whenever a button is pressed. We've provided some starter code in `lora-communication`

  In fact, the code we give you is almost a complete app. It is already capable of receiving continuously and transmitting on button presses. What you'll need to do is pick a valid LoRaWAN channel and data rate to configure it. Also make sure you understand the code, as you'll need to modify it for future steps.

- To keep yourself from getting interference from other class groups, I recommend you pick a LoRaWAN channel at random from the 64 uplink channels. See the section on LoRaWAN Documentation for details.

1. **TASK:** What LoRaWAN configurations did you choose?
   - Which channel are you communicating on?
   - Which data rate are you communicating with?

2. **TASK:** Demonstrate ability to communicate between two devices
   - Show me the terminal output from communication
   - Commit your `lora-communication` code to your shared repo

6. Range Testing

What is the maximum range from which you can still transfer data between two Heltec devices?

- Start off by copying your code from the `lora-communication` example into the `lora-range` application.

- You'll likely need to modify or add some configuration parameters of the physical layer and chip configurations to maximize range. Some things to consider:
    - Transmit at the maximum output power the chip is capable of for maximum range.

    - You must also ensure that the "current limit" allows transmissions to use that much energy (it exists to ensure that the transceiver doesn't accidentally use more power than is available). You can safely set it to the maximum 140 mA configuration, although whatever you're plugged into will need to provide that much. Any laptop port should be fine.

    - The SX1262 has a "boosted gain" reception mode, which uses more energy, but allows it to successfully receive packets with lower signal strength

    - Higher spreading factors result in lower bit rates, but longer range. However, you are limited to the valid LoRaWAN data rate configurations (due to dwell time restrictions)

    - Other physical concerns are a little fuzzy.
        - Having one of the two devices up high definitely helps but being indoors transmitting through walls hurts, and it's not clear which is dominant.
        - Holding the antennas vertically may help as their gain patterns are stronger going horizontally from the antenna. But it likely does not matter much.
        - Moving while transmitting or receiving likely hurts a little. Something something Doppler.

- **WARNING:** make sure you respect the FCC limitations for communication.
    - See LoRaWAN Documentation for details

- You definitely should modify the application code however is useful to you. It may be helpful to have your device automatically transmit LoRa packets, or automatically respond to LoRa packets.

- To actually do the testing, at least one of you is going to have to physically walk around to other locations. You can probably use a GPS plus some map application online to figure out how much range you get.
    - I expect this to take a little while to test out properly, but it's not a rigorous scientific study.

      If it's only been ten minutes, I don't think you tried hard enough. If it's been more than a few hours, you probably tried too hard and can cool it.

1. **TASK:** How did you maximize range?
    - What configuration choices did you make to maximize your communication range?
    - Commit your `lora-range` code to your shared repo

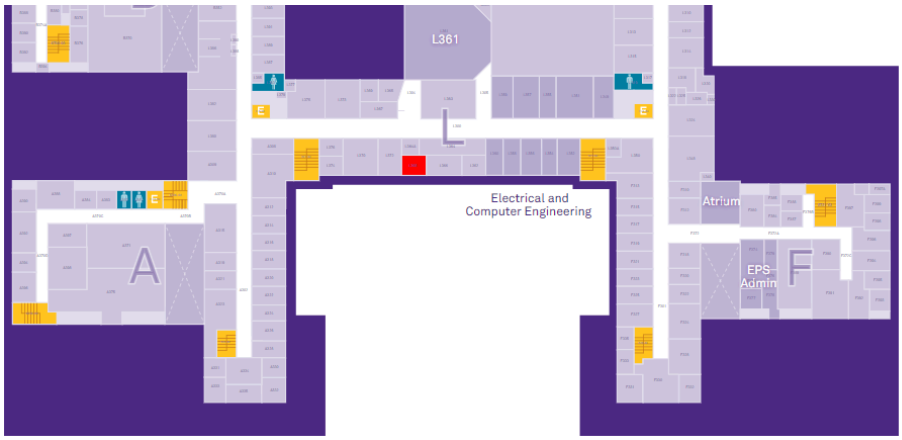
2. **TASK:** What is the maximum range you achieved?
    - Include an estimate in meters
    - A picture of a map seems like it would be helpful here

7. Find My Signal

This task will require some automated scanning and reception to find a mysterious, but valid LoRa packet that is being transmitted.

- I have placed a device in my office that sends a LoRa transmission once every 20 seconds. It is transmitting at high power, such that it should be detectable from some places in Tech.
    - The walls in Tech seem to be really hurting the signal. I found signal anywhere in the L wing of tech on the 1st-4th floors. The ground floor would occasionally get signal.

    - My office is located in the middle of the L wing (L368). The front of the Tech building (towards Sheridan Road) should get great signal, and given an absence of walls, I would expect to hear it further out that direction.



- The parameters of the communication are:
    - Data Rate 0
    - On one of the sixty-four 125 kHz upstream channels
    - Otherwise default LoRaWAN as we provided in the example code
        - Coding rate: 4/5
        - Sync word: 0x34
        - Preamble length: 8
    - One transmission every 20 seconds which starts with “397Ghena”

- Write an app that can scan LoRaWAN channels and display received transmissions. I recommend you start by copying your previous example code into the `lora-scanner` application (but no need to transmit here)
    - To find my particular device you’ll need to dwell on each individual channel for long enough that any repeated transmission will be heard.

    - To speed up the process, you could scan on three different devices simultaneously with different starting channels.

- WARNING: you might have to restart continuous reception after changing configurations…

- A suggestion is that you should make sure you can find your own signal before attempting to find my signal. That way you can make sure the whole system is working before sitting around for a while searching for a packet.

- A complication: it seems that something is a little weird with these transmitters and they leak packet data on other channels when devices are very close together. Not necessarily adjacent channels, but rather multiples of the frequency. I'll be honest, I'm not entirely sure why it's happening.

  So, the net result is that if you're close in proximity to the transmitting device, you might hear the secret message transmitted on several channels. One of those channels will be *significantly* stronger RSSI than the others though, so you can definitely still tell it apart.

1. **TASK:** Find my signal
    - Which channel am I transmitting on? (make sure this is the channel with the strongest RSSI, if you see multiple) Record the actual frequency, so we know we're talking about the same thing.
    - What are the contents of the message?

2. **TASK:** Submit your scanner application code
    - Commit your `lora-scanner` code to your shared repo

8. Protocol Design and Implementation

As our last lab assignment, and the assignment with the most raw control over the radio interface, this is a good time to design our own radio protocol! Our goal is to create a system with one gateway and two end devices where data can be communicated between the two end devices even though they keep their radios disabled the majority of the time.

- Work with the `lora-custom-gateway` and `lora-custom-device` applications. You should probably copy over some starter code from earlier to start from.

- Application end-goal: two sleepy (not always listening) end devices connected to a single always-listening gateway communicating with each other. End devices declare their own address when joining and can transmit data with a destination address. The gateway will hold onto the data for a destination that has joined the network until it requests it, at which point it is transmitted to the proper end device.

Network Parameters

- Physical Layer
    - **Pick two channels for communication**: one for uplink to the gateway and one for downlink from the gateway out of the sixty-four 125 kHz LoRaWAN channels.
    - **Choose a LoRaWAN data rate** that all devices will always communicate at. No need to implement multiple data rates or adaptation.
    - Keep LoRaWAN standard parameters for other configurations.
- Link Layer
    - Star topology network, with a Gateway and End Devices
    - End Devices use Aloha access control for transmissions. They MUST listen for a brief window after each time they transmit, with exact duration and offset are up to you. Remember that they should listen on a different channel from the one they transmit on.
    - The Gateway always listens for uplink packets. When it receives a packet, it should respond on the downlink channel, if it makes sense to do so, and then immediately return to listening on the uplink channel.
    - **The packet format is entirely up to you.** You might consider standard fields for length, type, source/destination addresses, or anything else that seems useful. You can also make packet-type specific formats.

- Network Layer
    - Gateway stores packets destined for an address that exists in the network. There can be some reasonable limit to the number of stored packets or duration that they are stored for. If data arrives when it cannot be stored, a negative acknowledgement is a reasonable response. The Gateway will indicate if stored packets are available for an End Device when sending an Acknowledgement.

    - End Devices must “join” the network to communicate on it and have their address be known by the Gateway. Once joined, all communication goes to the Gateway only and is always responded to (either Acknowledgement, Negative Acknowledgement, or other packets as appropriate). The End Device can request a stored data packet to be delivered to it by the Gateway.

    - A number of standard network communication packet types are defined below. Which MUST be implemented. Invalid packet types can be entirely ignored.

    - This network does NOT implement confidentiality or authentication.

Packet Types

Packet Type	**Destination**	**Response Type**	**Purpose**
Scan	Gateway	Scan Response	Determine if a network exists on this channel.
Scan Response	End Device	None	Indicates details about the network: at least a network name, possibly channels it uses, anything else that seems important.
Join Request	Gateway	Join Response	Request to join the network. Includes the address to refer to this device by.
Join Response	End Device	None	Indicates if the join was successful. Should reject a join request if the address is already in use.
Data Uplink	Gateway	Acknowled gement	Packet data to be stored for another End Device that has joined the network. Packet payload may be zero length with a destination of the gateway, which is used to test network connectivity or check for stored packets.

Acknowledgement	End Device	None	Indicates whether data has been successfully received and stored. A “Negative Acknowledgement” occurs when the Gateway fails to store the data because the destination is unknown or too many packets are already stored. The Acknowledgement packet also indicates whether, and possibly how many, packets are available for this particular End Device (the one that originally transmitted the Data Uplink).
Data Request	Gateway	Data Response	Request a stored packet from the Gateway for this particular End Device address.
Data Response	End Device	None	Contains packet data destined for this End Device address if any exists. Indicates that no data exists, if there were no stored packets. The Data Response also indicates whether, and possibly how many, remaining packets are stored on the Gateway for this End Device address.

- Generally, I’m sure I left some ambiguous things in that specification. You are free to interpret ambiguous items in any way you please, within the spirit of the assignment. I’m happy to answer questions about what likely makes the most sense.

  If you find any out-right contradictions, please let me know so I can fix them. If there are minor modifications to the protocol requirements which would make implementation easier, feel free to ask for them, although I may or may not grant them.

- Implement your protocol for End Devices and a Gateway and demonstrate an example application that can communicate between the two End Devices successfully.

1. **TASK:** Describe your protocol design
   - What are the physical layer parameters you chose for your protocol?
   - What is the packet format(s) for your protocol?
   - Also describe any other protocol decisions you made.

2. **TASK:** Demonstrate your full application
   - Show me screenshots of it working in terminal
   - Commit your `lora-custom-gateway` and `lora-custom-device` code to your shared repo