# Requirements from Coach’s Email

## EE Research

* LoRaWAN specifications 1.0 & 1.1 - All
* RF emissions standards - US915
* Interference e.g. In the Wichita area entities like Explosives Ordinance Disposal Teams use nearly the same frequency as LoRa devices. Your jobs to is to ensure that the spec is met so as not to interfere with other critical infrastructure. During your brief, you must demonstrate that you know the spec and your plans to mitigate violations.
* Antenna design

## CS Research

* LoRaWAN specifications 1.0 & 1.1 - All
* AT Commands
* C++, Python, or Embedded C
* Server and Gateways function
* OTAA or ABP - We will use OTAA, it’s more secure
* ChirpStack, ThingsBoard, Grafana, NodeRed, MQTT
* Data management, encryption and decryption requirements (if any)
* Libraries to use e.g. LMIC
* Device choice e.g. Arduino, STM32, or Raspberry Pi. Support choices with LoRa Specifications.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Everyone | Alex | Moe | Edgar | Aron |

# OTAA vs ABP

OTAA

Dynamic DevAddr and Session keys, these change when each new session is established

Can move to different networks/clusters since the DevAddr can change

Negotiate frame counters when it re-established the connection so it doesn’t have a specified end date

Generally no drawbacks but some requirements are needed:

1. **MUST** be within range of the network that it was registered on when turning on, otherwise the join procedure will fail

ABP

Fixed DevAddr and Session keys, more limitations from this method.

Works only in it’s defined networks even if it can work in others

Ends when it runs out of frame counters

Session keys are fixed, so if they are leaked they can

ABP uses fixed network parameters so if not configured correctly it will not work.

Conclusion: **OTAA**

It’s more secure, easier to use, just needs to be near the server when it starts originally

# ChirpStack

A server that is able to do everything, it also has an MQTT broker that transmits data so we can have applications on the network reading in information.

This is a good idea for us to use because we can create a program to show information form the MQTT data (and we can send signals from there)

Open source, and free to use.

# Gateway to Network Server

Once the hardware side of the gateway demodulates the uplink messages, it needs to forward them (usually through TCP/IP) to the network server. There are several ways to accomplish this.

## MQTT

A way to publish and subscribe to data, it is light weight and can connect to many devices. You have devices that publish to a broker who then sends out the information to the connected devices.

<https://github.com/chkr1011/MQTTnet> - C# .NET MQTT server/client (we want client if we use chirpstack)

ThingsBoard

Free but advance version is not

Seems a bit too complex for what we want, also doesn’t give enough

Also seems to rely on MQTT mainly which ChirpStack also does that

Grafana

Seems good, tons of data monitoring. It can connect to a ton of different devices. And has a lot of different monitoring options.

From LoRaWan PDF 1.0

Developed by semtech

ISM band range, USA 915 MHz

Little endian is used

# Device Classes

Class A

Lowest power, only down links when its uplinks

After an uplink from the node, the node opens 2 windows (+/- 20 microseconds after uplink) (time to receive) downlinks

The first downlink is on the same frequency as the uplink

The second downlink is on a configured frequency

The time for the downlinks must be long enough for the nodes antenna to open up and receive data

**MUST** wait for the downlinks to go through before sending another uplink

We can do other transmissions (other than LoRaWan) except we must follow local regulations

Class B

Class A + scheduled downlinks

Class C

Class A + continuous downlink availability. Because the end-device is constantly listening for a downlink, Class C is best reserved for applications where powre is not a concern.

All devices must implement Class A functionality

**Uplink** – message sent from one or more end-device (node) to a server

**Downlink** – A message sent from a server to one end-device (node)

**Message types:** There are 6 different message types (join request, join 2   
accept, unconfirmed data up/down, and confirmed data up/down.) **LORAWAN 1.0**

|  |  |
| --- | --- |
| **Message Type** | **Description** |
| 000 | Join Request |
| 001 | Join Accept |
| 010 | Unconfirmed Data Up |
| 011 | Unconfirmed Data Down |
| 100 | Confirmed Data Up |
| 101 | Confirmed Data Down |
| 110 | RFU |
| 111 | Proprietary |

**Links:**

1.0 vs 1.1: <https://www.tarlogic.com/blog/lorawan-1-0-vulnerabilities-and-backward-compatibility-in-version-1-1/>

1.0 vs 1.1 security PPT: <https://speakerdeck.com/rlifchitz/from-lorawan-1-dot-0-to-1-dot-1-what-are-the-security-enhancements?slide=1>

Server stack: <https://www.chirpstack.io/>

Overview of everything: <https://www.thethingsnetwork.org/docs/lorawan/>

^This one is very useful and gives a great technical ish overview