SMART PARK

TECHNICAL JOURNAL

Senior Design II

Recorded By Damian Avery

(EE undergrad)

Introduction

Smart Park is an application that will direct users to an available parking stall on campus via a UI (user interface). To accurately determine an occupied or vacant parking stall, the team has implemented a sensor package that can detect a vehicles presence and then send the data to the app. The team is split into a CS component and an EE component. The CS side is responsible to the app while the EE team is responsible to the Node and Gateway. Since this is an EE technical Journal only the EE component will be discussed in detail.

The node consists of a LoRa module, two sensor types, and a housing. The Lora module is an Arduino microcontroller called the MKR WAN 1310. This microcontroller can read data collected by the sensors and be programmed to determine a Boolean logic of this data. It can then send a LoRa signal (Long Range) to a Gateway. The Gateway can demodulate the data and transfer the data to our app. Detection is accomplished with a Magnetometer that can detect ferrous metals and a time-of-flight sensor that uses line of sight to recognize an object. Lastly the housing is an enclosure that protects and insulates the components from the outdoor environment.

Progress:

In Senior design I the EE team successfully created the node that could detect ferrous objects and then transmit the data to the application through an elaborate data infrastructure. The EE’s primary focus in senior design II is to improve our sensor package and make it a feasible / practical option for our target customer.

Key Take-aways:

Feedback provided on the performance of team members has been heard and will be addressed. Workload and clear contributions have been a concern for the Smart Park group. In Senior Design II we have delegated each EE team member a component to be responsible for. Records and data collection of tests performed on the project has also been inadequate. For this semester I will be following a standard template for test plans. This will be a more technical document that will include screenshots and a record of tests that have been performed on the project.

Issues that have been observed on the hardware and components have also been taken into consideration. We have observed that our signal connectivity from the Node to the Gateway was inadequate. One hypothesis is that the microcontroller needs to run on a high-power mode to have the signal strength to connect to the gateway. This obviously presents a problem of battery life for the finished product. The second hypothesis is that the Module itself is not suitable for the range needed for our application. To solve this issue, I have been delegated to test and verify signal integrity of the Node.

**Entry 1**

**Date: 02/13/22**

My responsibility has been to test the signal integrity of our current sensor package. Before doing so I have created a standard test plan to record all data. On the test plan I have included a hypothesis for why we have inadequate signal connectivity. Then I have listed ways to diagnose the problem and find the root cause. These steps are taken sequentially to rule out root causes and save time if applicable. The first step was to read literature on the MKR WAN 1310 transceiver and antenna for technical specifications. I was not able to determine the signal range of the configuration. I was however able to determine settings that could be changed to increase signal strength at the cost of higher power consumption. I also read on forums, which I will not use as a credible source, that the Arduino microcontroller is cable of sending a LoRa signal 6 km. I will also note that the entire purpose of LoRa is for long range connectivity. The next step is to connect to the gateway with our node and test several uplinks from various distances. Then I would like to test uplinks using the high-power mode called PA\_BOOST and compare both data sets. This part will take a week of preparation to perform the official tests.

Problems:

The Smart Park team has had multiple discrepancies with the ChirpStack account provided at Go Create. Due to this continued issue, I have decided to build our own gateway. Successfully configuring the gateway, troubleshooting the PA\_BOOST functionality, and performing the tests will take a week’s time. See attached test plan for this weeks Technical Journal entry.

Test Plan: Signal Integrity

Date: 02/12/2022

Hypothesis:

The signal integrity of the MKR WAN 1310 has been inadequate. The problem may lie in the capability of the microcontroller. The antenna may not be capable of reliably connecting due to low power or the manufacturer intended for shorter distances. Obstructions such as buildings could also be a culprit as well as the mode set on the component. I must determine if the current hardware is suitable for continued use of our application. We need thousands of nodes to be able to reliably send signals to a gateway.

Plan:

First step is to locate literature on the MKR WAN 1310 and thoroughly look up the signal distance of the antenna. Depending on the results of this search the next step will be to connect a module to the gateway and send signals from varying distances. This will require regaining access to the Chirpstack server to observe data received by the gateway. A large sample size of tests will be performed to provide a reliable statistic of the results collected. Lasty the Module will be tested using the high-power mode and results will be compared.

Results:

Step 1 Research (literature)

7.2 On page 6 of the Sub-G Module Data Sheet discusses the LoRa transceiver Specifications. Under the LoRa Transmitter Specification table below it is observed that different modes can be set to achieve a longer range while consuming more power. Using the PA\_BOOST pin we can reach a 20 dBm signal (decibels relative to one milliwatt). The power consumption can be calculated using the equation below.

Table

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

20 decibels relative to milliwatt consumes 100 milliwatts of power. I was not able to determine the capable range of the transmitter. I did however read on a forum (not the reliability I was hoping for) that the range of the microcontroller could reach up to 6 miles. The entire LoRa network and wireless communication type is meant for Long Range data transmission, so I do not believe that the module is not intended for long range connectivity.

Step 2 Uplink Test (Will not be completed until 02/27/22)

I will now prepare for the gateway connection to test the uplinks and collect the data using the table below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Signal Integrity Test | | | | | | | |
|  |  | No PA\_BOOST | | PA\_BOOST | | No Total | PA Total |
| # | Distance | Signal | Uplink | Signal | Uplink |  |  |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |

[dBm - Wikipedia](https://en.wikipedia.org/wiki/DBm)

[Type ABZ | CMWX1ZZABZ | Datasheet | LoRa Module | Murata Manufacturing (arduino.cc)](https://content.arduino.cc/assets/mkrwan1310-murata_lora_module-type_abz.pdf?_gl=1*d51rbg*_ga*MjA5ODU3NDE1My4xNjQ0NzE5MTQw*_ga_NEXN8H46L5*MTY0NDcxOTEzOS4xLjEuMTY0NDcxOTE0MC4w)

**Entry 2**

**Date: 02/20/22**

I have continued to attempt to connect to the gateway to utilize our first piece of infrastructure. I was unable to connect to the application server while uplinking a node directly under the Tektelic Kona Macro gateway supplied by Go Create. Although I cannot officially confirm without access to the Chirpstack account, our application no longer exists. This is needed to send successful uplinks and join requests to the gateway via the application key and device EUI keys. Since I cannot perform signal integrity tests without a working gateway, I have been pursuing our own gateway.

Gateway Build:

We had previously purchased a Rasberry Pi and RAK2245 LoRa Concentrator that will be suitable for the gateway and gateway bridge. The gateway application has been successful; however, the application server and network server must still be integrated to officially have a working system. To accomplish this without the available hardware, I am using a virtual machine through the GCP (Google Cloud Platform). The virtual machine will act as both the application and network server allowing Chirpstack to run successfully.

Key Info:

Gateway Channel – US915 **16-23** +66 (frequency hop channels)

Gateway ID- e45f01fffe17080d

Operating system- Rasberry Pi 4 OS-Full gateway operating system (provided by Chirpstack)

Application Server- GCP virtual machine N2 8GB (Ice Lake)

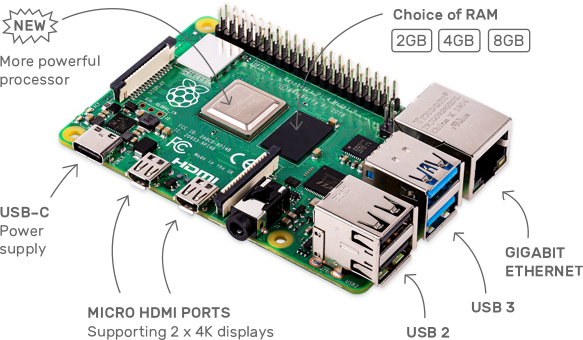
Network Server- GCP virtual machine N2 8GB (Ice Lake)

Broker- MQTT

Application- smart-park-342405

<https://artifacts.chirpstack.io/downloads/chirpstac-gateway-os/rasberrypi/rasberrypi4/3.5.1/>

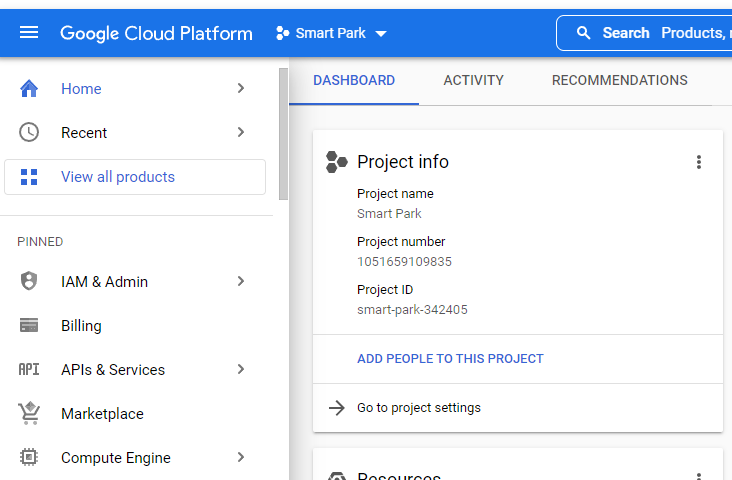
**Rasberry Pi 4 Bridge from Lora to IOT**



**Rak2245 for demodulation of Lora data**



**Google Cloud Platform for Application and Network Server**



Once these steps are complete and the gateway is functional, I can resume signal integrity tests.

Project Tracking (current work): Assignments and activities are to be tracked until completed.

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Assignment | Due Date | % Complete |
| Alex Chiem | Casing (research phase) | 03/12/2022 | 5% |
| Damian Avery | Signal integrity and repackaging | 02/25/2022 | 0% |
| Martin Kairuki | Sensor package verification | 02/25/2022 | 0% |
| Max Burrell | Battery and Solar panel | 02/25/2022 | 0% |

Plan (future work):

|  |  |
| --- | --- |
| Assignment | Due Date |
| Cohesiveness of individual efforts | 02/26/22 |
| Testing of any repackaging | 03/05/2022 |
| Data collection and recording | 03/03/2022 |

**Entry 3**

**Date: 03/13/22**

The gateway has successfully been built using the above materials and Rasberry Pi 4 OS-Full gateway operating system (provided by Chirpstack). The prompt allows me to configure the gateway and gateway bridge. I have created a device in the GCP using self-signed PEM certificate and key. This has been the most challenging step as the GCP did not accept the public or private PEM formatted key to be entered as my gateway’s unique identifier. I may have generated 100 different keys using different methods to have successful device integration.

Through word of mouth our sponsor has communicated that they will be moving over to a cox server to provide the gateway infrastructure that was provided from last semester. I have not received any direct communication from our sponsor. I’m hoping to finally have the infrastructure completed before the end of spring break. I can then transition to signal boosting for optimal device communication.

**Entry 4**

**Date: 04/03/22**

I have successfully integrated the Raspi gateway with the Chirpstack application and network server. Utilizing the Chirpstack full OS I did not need to use the GCP as assumed. The full OS includes all the gateway components. The Raspi houses the application UI internally allowing for connection via <http://[IPaddress]:8080> entered on any browser. The Raspi must be connected to the local network to access the UI. The OS allows for wifi connection however I was unable to get the wifi connection working so I used ethernet. With my laptop I was able to login and create a new network server, app user accounts, Smart Park organization, and a new Smart Park application. These are like the steps taken last semester using the pre-existing Go Create Gateway setup. The benefit is now we control the entire infrastructure ensuring no tampering and reliability.

This crucial step allows for signal testing between the node and the gateway monitoring the connection status via the Live Lorawan Frame. I made successful connections to the gateway with our Lora module using OTAA (Over the air activation). When running the code seen below the frame would populate a “Join Request” followed by a “Join Accept”. This step completes the connectivity of the end device to the gateway. After this connection the program must be looped to continuously send uplinks. If the program ends, a new OTAA will have to be administered. Because of this I altered our showcase code from last semester allowing for the user to input a printed message 10 times to send 10 uplinks to the gateway without having to redo an OTAA. Now the stage is set to complete the tests needed to determine the optimal configuration. I began my testing by uplinking “hello world” to the gateway. It should be noted that the frames will not display the message rather it will populate “confirmed data up”. This worked almost instantly after the activation was complete. Shortly after the Raspi stopped receiving data from my router. For an unknown reason the gateway system was stuck trying to complete a connection as indicated by the RJ45 port LEDs. A lot of progress was made on the gateway efforts however, unfortunately hardware reliability could be an issue as I’m using an exposed shield with numerous connections and disconnections.

Updated (04/03/22) Key Info:

Gateway Channel – US915 **0-7** +66 (frequency hop channels)

Gateway ID- e45f01fffe17080d

Operating system- Rasberry Pi 4 OS-Full gateway operating system (provided by Chirpstack)

Application Server- Included on OS-Full gateway

Network Server- Included on OS-Full gateway

Broker- MQTT

Application- Smart Park

Raspi login: admin

Password: admin

Connection: RJ45

IP: 192.168.2.1, 192.168.6.1

(Note the IP randomly changes upon boot up, I use an IP scanner to locate the Gateways IP needed to login in to the Chirpstack application server)

Chirpstack website: admin (user 1), djavery17@gmail.com (user 2)

Password: admin (user 1), Smartpark22 (user 2 – case sensitive)

Device code:

AppKey: “00000000000000000000000000000000” (this is the application key that I defined in the Chirpstack application server, must match on the device for activation)

DevEUI: “a8610a34343c800f” (this is the device EUI key that is set by the device, it must match on the Chirpstack application server for device activation)

<https://artifacts.chirpstack.io/downloads/chirpstac-gateway-os/rasberrypi/rasberrypi4/3.5.1/>

Graphical user interface, text, application, email

Description automatically generated

**Entry 5**

**Date: 04/17/22**

**The last piece needed to effectively complete the Smart Park data infrastructure is to configure the Chirpstack Integration. For this we will be using Things Board. To configure the two components into one object I will be taking to separate approaches. The first approach will utilize the original Chirpstack OS and a cloud version of things board. The second approach will use a completely new gateway Chirpstack software setup using Ubuntu to add the Things Board application locally to the Raspberry Pi. The second approach will require setting up the network and application server as well as the gateway bridge and MQTT broker all over and individually instead of using the precompiled binary OS. Each scenario will have its own micro SD card for install. This will give us backup options in case of faulty gateway operation .**

**Lastly through Things Board we can more accurately monitor gateway to node connectivity. Using the original Chirpstack frames was not optimal enough for a reliable way to test signal between devices. Things Board provides user interfaceable diagnostics that include things such as signal strength to device.**