University of Waterloo
Faculty of Engineering
Department of Mechanical and Mechatronics Engineering

MTE 482 Final Report

Prototype of an Automated Parking Management System

Prepared By Group 18:

Hanson Ng | 20614977

John Hwang | 20362107

Michael Peet | 20604483

Kelvin Tezinde | 20619069

March 27th, 2020

Kennings & Khamesee

Associate Professors
Department of Mechatronics Engineering
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Dr. Kennings and Dr. Khamesee,

This report, entitled "Prototype of an Automated Parking Management System", was written as Group 18's MTE 482 final report. The report initially reviews the background of the problem and the proposed design from MTE 481. The report examines the final design, the process of developing the prototype, and the final scheduling and budgeting.

The report was written entirely by Group 18, unless otherwise stated and properly referenced in the report. We want to thank Dr. Nassar, Dr. Kennings, and Dr. Khamesee for their guidance in the design process throughout the past two terms. The content of this report has not received credit in this or any other course that we have taken in the past or are currently taking at this time.

Sincerely,

John Hwang Hanson Ng Michael Peet Kelvin Tezinde

Executive Summary

The parking experience continues to be riddled with difficulties and inefficiencies for drivers and parking lot owners. As the number of vehicles in Canada continues to increase, parking space issues are a growing concern. Drivers' lack of information and insight into the parking situation around them results in costs of fuel, time, and emissions. This lack of information also results in missed opportunities for parking lot owners to effectively plan and optimize lot utilization. There is a need for drivers to efficiently find parking spaces as well as for parking lot owners to obtain insight into their lots.

The final report reviews the constraints and criteria of the solution and the design of the proposed solution in MTE 481. Then it explores the final design of the solution and creation of a prototype. The solution consists of four main components: pods, a gateway, a network server, and a web application. Each pod contains a BNO055 IMU for vehicle detection and uses a Feather M0 microcontroller equipped with an RFM95 chip, used to communicate its occupancy state using LoRa (Long Range) technology. Each pod is also equipped with a 1S 1200 mAh battery, expected to run the pod for approximately a year. Each pod also has a low-profile 3D printed chassis designed to withstand loading under medium-duty vehicles. The waterproof chassis can be installed either on the road surface or underground in any parking lot configuration. Each pod broadcasts a parking state every 30 seconds to a Dragino LG02 LoRa Gateway. The gateway relies on WiFi to forward the information to a network server provided by The Things Network. A web-application developed using Express.JS and React.JS provided real-time parking visualization by listening to network server events. Historic and testing data is stored in a PostgreSQL database to generate analytics. The final details, modifications from original plans, construction, and testing process are discussed for the system architecture, pod chassis, pod hardware, communication, and web application.

Despite minor setbacks and additional costs compared to the original schedule and budget set forth in MTE 481, the final project takes a total of seven months to complete while all the deliverables and goals are accomplished in a timely manner. The project cost totals to $693.18, well within the constraint of $1,000 set forth initially.

Table of Contents

Executive Summary **3**

Table of Contents **4**

List of Figures **6**

List of Tables **6**

1. Introduction **7**

1.1 Background 7

1.2 Needs Assessment 7

1.3 Problem Formulation 7

1.4 Constraints and Criteria 7

1.4.1 Constraints 8

1.4.2 Criteria 8

1.5 Design Review 9

1.5.1 System Architecture 9

1.5.2 Mechanical 9

1.5.3 Hardware 10

1.5.4 Communication Protocol 10

1.5.5 Web Application 10

2. Final Design **11**

2.1 System Architecture 11

2.1.1 Final Design Details 11

2.1.2 Testing and Performance 12

2.2 Mechanical Module: System Chassis 12

2.2.1 Final Design Details 13

2.2.2 Modifications/Deviation from Original Design 14

2.2.3 Construction/Manufacturing 14

2.2.4 Testing and Performance 14

2.2.5 Commissioning 15

2.3 Microcontroller, Sensor and Power System 15

2.3.1 Final Design Details 15

2.3.2 Modifications/Deviation from Original Design 16

2.3.2.1 Microcontroller 16

2.3.2.2 Battery 16

2.3.2.3 Sensors 17

2.3.3 Testing and Performance 17
2.4 Communication 18
2.4.1 Final Design Details 18
2.4.1.1 The Things Network 19
2.4.1.2 LoRa Channels, Bandwidth & Spread Factor 19
2.4.1.2.1 LoRa Channels 19
2.4.1.2.2 Bandwidth 19
2.4.1.2.3 Spread Factor 19
2.4.1.3 Gateway 20
2.4.2 Testing and Performance 20
2.5 Web Application and UI/UX 21
2.5.1 Final Design Details 21
2.5.1.1 Database and Server Code 21
2.5.1.2 Client Application and UI/UX 22
2.5.2 Modifications/Deviation from Original Design 26
2.5.3 Testing and Performance 26

3. Schedule and Budgeting 26
3.1 Schedule 27
3.2 Budgeting 28

4. Conclusions 28

5. Recommendations 29
5.1 Mechanical 29
5.2 Hardware 29
5.3 Web Application 29

6. Teamwork Effort 30

Reference 31

Appendix A: As-built Drawings 32

Appendix B: Simulation Results 33

Appendix C: Project Schedules 34

Appendix D: Cost Breakdown Summaries 35

Appendix E: Database Schema 36

List of Figures

Figure 1: System Architecture 10

Figure 2: Sensor Pod Module Chassis Overview 12

Figure 3: Diagram of an Individual Pod Hardware 15

Figure 4: Visualization of Magnetometer Tuning 17

Figure 5: Popup and Sidebar Components 22

Figure 6: Parking Lot View 22

Figure 7: Analytics View 23

Figure 8: Example Time Frame Comparison 24

Figure 9: General Usage Information to Direct Analytics 24

List of Tables

Table 1: 3D Print Test Summary 14

Table 2: Web Application Design Goals 21

1. Introduction

1.1 Background

The demand for parking in Canada continues to increase as the number of cars steadily rises [1], creating issues for drivers and parking lot owners.

Difficulties with finding parking for many drivers arise from a general lack of information and insight into parking space availability. On average, U.S. drivers spend a total of 17 hours a year searching for spots to park in. When aggregating the costs of the time, fuel, and emissions, the average driver spends approximately $350 annually. City drivers deal with the worst parking challenges. At the top of the list, drivers in New York City spend 107 hours and upwards of $2000 annually on the search for parking [2].

A lack of information and insight into a lot's parking spaces results in missed opportunities for parking lot owners to effectively plan and optimize lot utilization. Without an efficient way of collecting parking data, trends in parking lot usage that could be leveraged in planning decisions and creating parking simulations are unavailable. Furthermore, parking lots currently heavily rely on manual enforcement techniques for illegal parking or parking beyond time limits which could be made more efficient with more insight into parking lots.

1.2 Needs Assessment

There exists a need for drivers to find parking spaces in a more efficient manner and for management to gain insight into parking trends.

1.3 Problem Formulation

To solve this need, the goal of the project is to develop a solution to track and manage parking spaces.

The goal can be broken down into the following four objectives:

- The system should work with most parking configurations
- The system should monitor parking space usage
- The system should operate mostly without human intervention
- The system should provide public data on up-to-date parking availability

1.4 Constraints and Criteria

The constraints and criteria stated are for the prototype of the solution and are more lenient than those for a fully finished product.

1.4.1 Constraints

The following set of constraints were defined to meet the objectives of the project.

- The system must be non-obstructive
- The system must have a minimum life of 1 year under ordinary circumstances
- The system must follow Canadian Electrical Code and other standards
- The system must cost less than $1000
- The system must be at least 90% accurate
- The system must update the parking status within ~1 minute of event

The solution must be non-obstructive. The product should not obstruct parking for most vehicles. The solution is restricted to a maximum height of 100 mm or the ground clearance of a Lamborghini Aventador [3].

The solution must have a minimum life of one year with limited maintenance. A year is a standard warranty period for most consumer electronics.

As the solution is operating in public areas, the device must follow Canadian transmission and electrical regulations.

A prototype cost of $1000 for the complete solution was set as customers could come from a variety of financial backgrounds.

Based on market research and the nature of the problem of dealing with crowded lots, a minimum accuracy of 90% was chosen for the prototype.

The solution must be responsive enough to update the parking state within one minute of a parking event. This time value corresponds to experimental testing of a reasonable amount of time for a driver to leave a space and a new driver to park.

1.4.2 Criteria

The following set of criteria were defined in order to choose the most suitable solution for the project.

- The system should have sufficient coverage area
- The system should require minimal maintenance
- The system should have minimal cost
- The system should be as accurate as possible
- The system should have the capability to work in a variety of parking configurations
- The system should not hinder a driver's ability to park
- The system should be easy and quick to install
- The system should have minimal response time

Firstly, the solution should have sufficient coverage area. If the coverage area is too small, it will result in increased complexity, and a prohibitive cost.

The solution should need minimal maintenance. It must be durable enough to handle contact with vehicles or put where it is unlikely to get damaged.

The solution should have minimal cost. To be useful for drivers, the solution must be deployed at many locations. As a result, the price should be low enough for lot owners to purchase and implement on a large scale.

The solution should be as accurate as possible. An accuracy that is too low would make the information less useful to drivers and it would be difficult to enforce parking limits.

The solution should be flexible enough to be deployed in any lot. It must be able to work in a variety of parking configurations such as in parking structures or street parking. It should be unobstructive - either out of the way of the vehicles, or small enough to fit underneath them. Installation of the solution should be easy and quick.

Finally, the system should respond to occupancy state changes as fast as possible. Operating at real-time provides the most accurate information and mitigates errors such as a car pulling out of a spot and another car pulling in before the system can respond.

1.5 Design Review

After review of three patents and a similar product, three unique solutions were conceptualized: a camera mounted computer-vision detection system, a rail mounted multi-spot sensor, and an individual parking space sensor pod. After ranking the criteria, a decision matrix was used to rank the solutions and it was decided that the team would track parking occupancy through the use of embedded sensors, relaying messages to a server through the LoRa network. It was determined that the proposed solution would have the best chance of meeting the specified constraints and criteria. This solution had the highest accuracy and flexibility scores, the two highest weighted criteria. It also had an average score for cost, coverage area, and responsiveness. It had the lowest durability score.

1.5.1 System Architecture

A star topology for the system was chosen over a mesh network where the pods will only send messages to a gateway. In this manner, the power draw is lower which helps meet the constraint of a 1 year lifespan and the singular function of the pod will keep update times under 1 minute.

The flow of information from pods to gateway to The Things Network enabled the use of an established and proven network system providing better reliability and works towards meeting the cost criterion.

Due to the difficulty of determining time elapsed between when a car is detected to when the data arrives at the application server, the constraint of updating parking state within 1 minute cannot be confirmed. However, considering the prototype setup of 4 pods and a 2 channel gateway, it can be expected with high certainty that the time taken to update parking status is well within 1 minute.

1.5.2 Mechanical

A compact underground pod chassis design was initially selected. The underground installation mode was chosen to minimize external damage to the pod chassis. ABS plastic was chosen for the chassis material

due to its low cost, excellent strength and resistance for outdoor usage. The cylindrical chassis was designed to have a dimension of ø $6\ cm \times 6\ cm$, a volume of $66.15\ cm^3$ and material weight of $67.48\ g$. 3D printing was the selected manufacturing process due to its cost effectiveness and time efficiency. Consequently, the chassis would be durable and inexpensive in terms of maintenance, production cost and labor. Thus, the chassis design would satisfy all the relevant objectives and constraints.

1.5.3 Hardware

The BNO055 IMU and Sharp GP2Y0A21YK0F IR sensors were initially selected for the purpose of vehicle detection. The combination of using the IMUs magnetometer for detecting vehicles and an IR sensor for detecting objects was calculated to be a low power and reliable detection solution. They were expected to meet the required accuracy and help with the power draw requirements. An ESP32 was also chosen as the microcontroller of choice, due to its low power usage.

The batteries initially configured to be used for the design would either be a 1S LiPo with an LDO or 2S LiPo with a buck-boost converter module. Previous calculations indicated that this battery choice would be able to run continuously for a year, and would keep the hardware budget below $1000. The battery was found to comply with Canadian Standards, and the chosen sensors were expected to meet the required accuracy.

1.5.4 Communication Protocol

LoRa technology and the LoRaWAN protocol were selected for the pod to server communication. LoRa has a range of 2-5 km in urban areas, can have thousands of devices connected to a single gateway, and is designed to be as low power as possible [4]. These properties made it an attractive choice for meeting the criteria of a 1 year lifespan and enabling a decent area of coverage.

1.5.5 Web Application

A relational database was chosen to store/organize the data over a non-relational database due to the static structure and relational nature of the data. The ability of a relational database to take advantage of this structure for fast query performance was a key factor in the decision. PostgreSQL was chosen as the relational database management system, as many team members had some experience with it.

When choosing a language for the server, multiple details needed to be considered. The language needed an API to connect to The Things Network so that updates from the parking sensors could be gathered. It also needed to be a language that multiple team members were familiar with, and should keep development as simple as possible. It was decided that the web server should be written in Javascript. It had an API to connect to The Things Network, and had a lot of community support for web development. Additionally, Javascript offered the opportunity to have the client and server written in the same language, using Node.js for the server, and React for the client. This would simplify development, as it avoided the need to use multiple languages in the same code base.

2. Final Design

The detailed final design of the data pipeline architecture, dot sensor modules, and web-application architecture is in the subsection below.

2.1 System Architecture

2.1.1 Final Design Details

The system architecture consists of 4 components which can be divided into two sections. The first section is made up of the parts physically required to be at the parking lot. The pods and the gateway are placed into this category. The second section is made up of the infrastructure and software required to gather, process, and visualize the parking data. The network server and web application are placed in this category. The final system architecture can be seen in Figure 1.

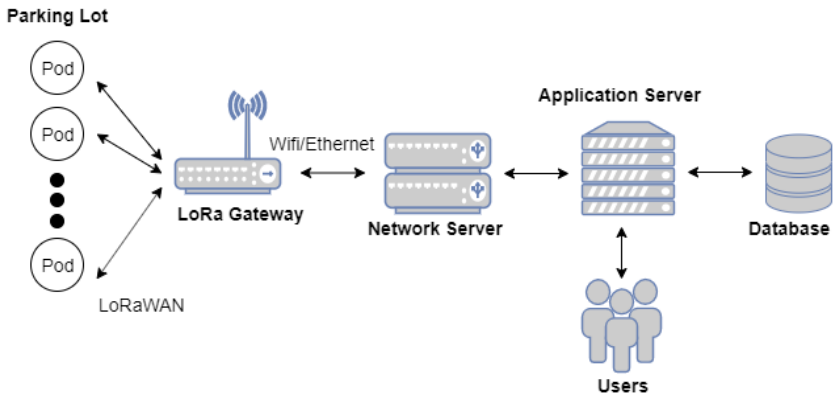


Figure 1: System Architecture

Each pod uses a sensor to determine the current state of a parking space and uses the LoRa transceiver to broadcast to in range LoRa gateways. The details of the hardware used in the pod are discussed in Section 2.3.

The LoRa gateway acts as an intermediate device to forward the pod's information to the internet. To handle the information from many pods, the gateway is connected to the internet using WiFi, a higher bandwidth network. Additionally, the gateway is within 250 metres of the pods and needs to be constantly connected to power.

The network server is responsible for routing the data from gateways to the correct application. The network enables the management and mapping of devices, gateways, and applications. Using the network server, the status of the pods, gateway, application, and data flowing between the three can be tracked. Additionally, due to the possibility that multiple gateways receive the broadcasted information from a

pod, the network server also handles data duplication issues. A network server provided by “The Things Network”, an open Internet of Things infrastructure was used.

The prototyped system consisted of 3 pods and a single gateway. Pods were configured to broadcast the state of its parking space every 30 seconds under a spread factor of 7, using channel 10 of the 915Mhz LoRa frequency. More details on specific configuration values, what they mean and why they were chosen will be explored in Section 2.4.

LoRa technology and infrastructure supports bi-directional communication and mesh topology. Each pod has the potential to redirect information through other pods to help handle gateway failure as well as receive information from the network server for device or gateway status. For simplicity, a star topology is used where a pod will only broadcast information and not be aware of other pods. As a method of determining if pods are active, pods which are detected to have not broadcasted within a time span larger than its set broadcast interval will be flagged.

2.1.2 Testing and Performance

Each component of the system required different tests to ensure that it was performing as expected. Tests for individual components will be reviewed in the following sections.

To ensure that the whole system was operating as expected, complete end-to-end tests were carried out. In this process, a single pod was placed on the ground with a car parking above it and reversing off multiple times in intervals of 2 pod broadcast periods. Visual logs were placed on the network server as well as the web application to determine the sent information was correct and within a reasonable timeframe. The application was also running to see if the parking space counters and visualizations were updated.

It was found that the delay experienced from the pod broadcasting its state to the web application detecting an updated state was consistently within 1 second. Updates within the web application to the database and visualization was consistently within 400 milliseconds.

In the operation phase, the entire system underwent end-to-end tests to ensure its full functionality. The live demonstration was presented to Profs. Kennings and Khamesee on March 6, 2019 and its reception was very positive. The testing process and results are recorded on the official project blog [5].

2.2 Mechanical Module: System Chassis

Several design iterations have occurred since the MTE 481 report submission. There are some changes and additions to the final design as a result of hardware and manufacturing modifications. These changes take results from the original design analysis and improve overall functionality and product appeal while accommodating other subsystems.

In the design phase, each iteration undergoes static analysis in SolidWorks in accordance with the requirements originally detailed in the MTE 481 report. Then the design is reviewed and verified by the

team. In the construction phase, the prototype chassis is 3D printed and undergoes functional tests to ensure all the specifications are met and verified. As the final step in the verification process, the integrated system test (live demonstration) takes place to ensure full system functionality before the product release (symposium).

2.2.1 Final Design Details

The original mechanical design reflects relevant constraints and criteria and targets optimal modularity, durability and manufacturing cost. Likewise, the final design abides by the same design intent and requirements, with additional modifications and additions.

The final mechanical design has a disk-shape geometry with an 18 cm base diameter, a 45° side chamfer and a 3.2 cm height (Fig. 2). It has an average weight of 246 g and a total volume of $306\ cm^3$. The chassis contains two parts: a top shell and a bottom tray. The top shell has an average 6 mm wall thickness and the bottom tray has an average 3 mm wall thickness. The top shell has 16 radial ribs to support the outer chamfered edge of the chassis, and the bottom tray has 7 partition walls acting as H-beams to support the center. The bottom tray has a fluid seal design to set the gasket sealant/glue between the shell and tray for water proofing. The four bolt slots allow the chassis to be anchored to pavement or cement, giving the on-surface or underground installation options once. Subsequently, the full design details are provided in the as-built drawings (Fig. A1, Appendix A).

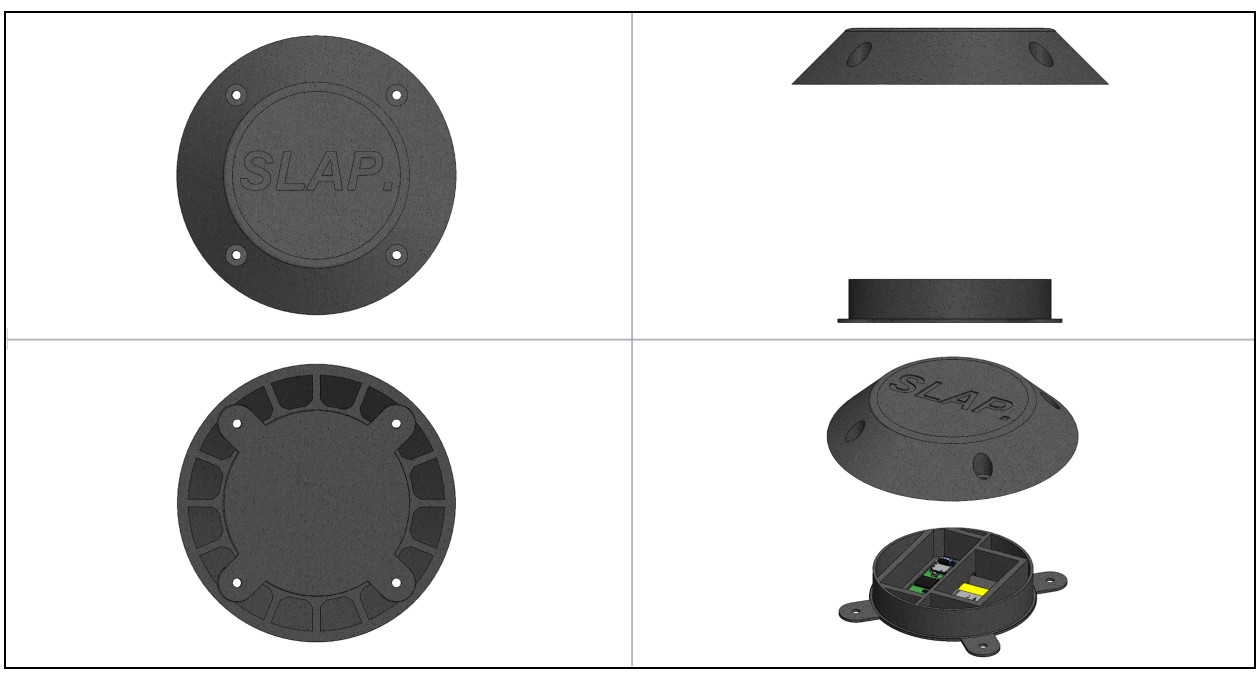


Figure 2: Sensor Pod Module Chassis Overview

2.2.2 Modifications/Deviation from Original Design

This section includes all mechanical modifications made in the final design. These modifications are necessary to accommodate changes in other areas while satisfying all specifications set out in the original report.

First, the chassis geometry has changed to accommodate hardware design modifications. The final design bears a disk shape to decrease its vertical profile (from 6 to 3.2 cm) and increase its base diameter (from 6 to 18 cm), instead of the original cylindrical shape. Subsequently, the 300% base diameter expansion and 50% height reduction allow additional structural support for better load capacity. It also increases the total material volume by 363% (from 66 to306 cm^3), and the weight by 262% (from 68 to 246 g).

Second, the chassis material has changed from ABS (Acrylonitrile Butadiene Styrene) to PLA (Polylactic Acid) since PLA is easier to 3D print with and more suitable for rapid prototyping. Particularly, PLA is preferred due to its low printing temperature and resistance to material warping [6]. With similar mechanical properties as ABS, PLA is strong enough to pass the load pressure test and suitable for the prototype demonstration [7]. In addition, the final design has enabled the system to be installed on road surfaces as well as underground, providing more flexibility.

2.2.3 Construction/Manufacturing

Key operation factors such as turnaround time, change readiness, and simplicity help determine the construction method. It is necessary to have a construction method that allows rapid prototyping without delay. Therefore, the selected solution is to 3D print the chassis using FDM (fused deposition modeling). After testing several FDM printers available, the Cubicon Single Plus desktop 3D printer is chosen to print the chassis using the Hatchbox 3D 1.75 PLA filament.

Next, 3D printer settings are systematically tuned using combinations of fill density, layer height and print speed. Consequently, printing at “medium” print speed with a 60% fill density and 0.20 mm layer height yields the optimal result that balances expense and quality. The 3D printing process requires minimum support material and takes about 16 hours to finish.

2.2.4 Testing and Performance

The mechanical module testing, from design to operation, is summarized in this section. The purpose for testing is to verify and ensure all requirements and design intent are conformed.

In the design phase, the chassis model undergoes several static simulations in SolidWorks to test the flexural and compressive performance under the worst-case load pressure of 1.245 MPa. The final design has a maximum strain value of 4.66e-3 and maximum Von Mises stress of 11.5 MPa, which are well below the allowable flexural [8]. Compared to the original model, it reduces the fracture stress by 23% (down from 15.0 MPa) and strain by 20% (down from 5.97e-3). The average yield stress and deformation have decreased due to structural enhancements such as interior support beams and radial ribs.

Subsequently, the full static analysis of the original and final designs are included in the simulation results (Appendix B).

In the construction phase, 3D print settings are tested to determine the optimal print result with respect to print time, quality and durability. These factors are important because they directly impact several aspects of the project such as scheduling, budgeting and, ultimately, product quality. Four combinations of print settings such as fill density levels (40, 60 and 80%), layer heights (0.15 and 0.20 mm), and print speed levels (fast and medium) are tested. The print speed has three presets: fast, medium and slow, and the slow speed option is omitted in testing due to lengthy print time. The print quality is tested based on inspection and the durability is done through load pressure testing. Consequently, print setting option #3 yields the optimal outcome (Table 1).

Table 1: 3D Print Test Summary

#	Print Setting Options	Print Time (hr)	Print Quality	Durability (Pass/Fail)
1	Fast, 40% fill, 0.20 mm	12	Worst	Fail
2	Medium, 60% fill, 0.15 mm	20	Best	Pass
3	Medium, 60% fill, 0.20 mm	16	Good	Pass
4	Medium, 80% fill, 0.15 mm	27	Best	Pass

2.2.5 Commissioning

The mechanical module commissioning consolidates with the entire system while fulfilling scheduling, budgeting and functional requirements. There are three main phases for commissioning: design, construction and operation and, in each phase, it is verified with respect to the design intent and requirements. In the design phase, the final design abides by the requirements and undergoes static analysis to search for failure modes while integrating with the hardware module. In the construction phase, the chassis is 3D printed using optimal print settings to minimize turnaround time and cost. Then the chassis undergoes functional testing for quality assurance. In the operation phase, the chassis allows the dot sensor system to be installed on road surfaces or underground. It also has a water seal design that prevents liquid damage on the hardware. The integrated system test and live demonstration occur at the end of the verification process where the entire system is tested and showcased from end to end.

2.3 Microcontroller, Sensor and Power System

2.3.1 Final Design Details

The hardware of an individual pod consists of the Feather M0 RFM95, a 1S 1200 mAh battery, a 78mm antenna and the BNO055 IMU. The system, then, is split into three components; power, sensing and

controller. Each unique component differs slightly from the initial design and as such, the section below shall expand on why such changes were deemed necessary. Figure 3 below shows a diagram of the hardware for a pod.

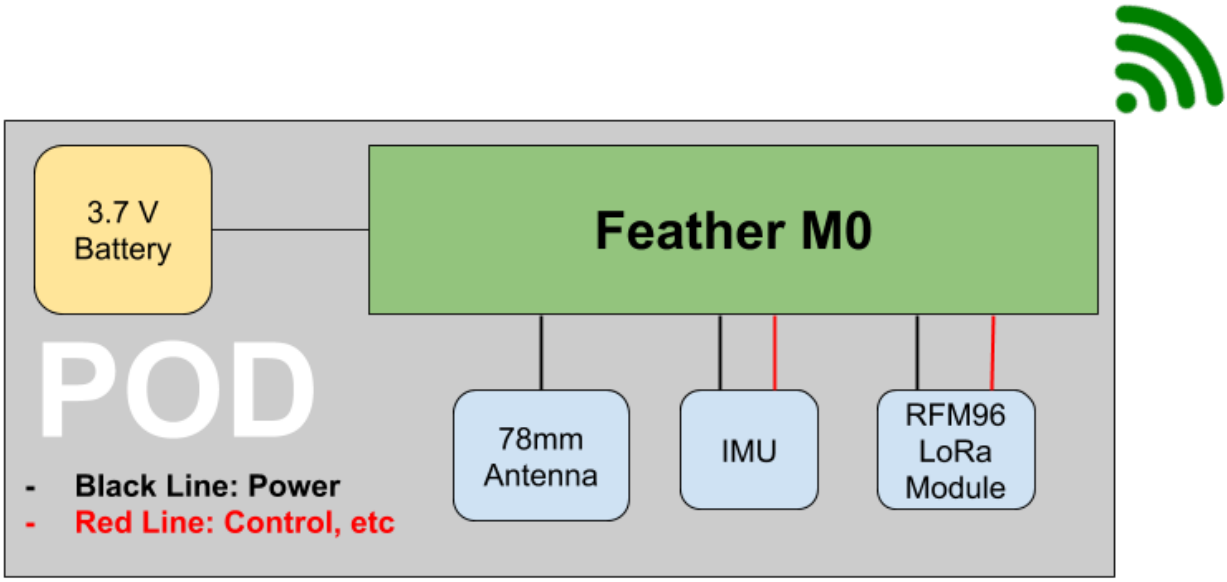


Figure 3: Diagram of an Individual Pod Hardware

2.3.2 Modifications/Deviation from Original Design

2.3.2.1 Microcontroller

Initially, the pods were to be controlled using the ESP32 microcontroller. The ESP was the chosen option for a significant amount of time because of its power mode options and in-built LoRa module. However, issues arose when attempting to use LoRa libraries and the Arduino IDE. Unlike Adafruit products which have a lot of support in the Arduino IDE (and consequently, the Arduino community), the ESP32 requires several drivers and JSON configurations to work with Arduino. Furthermore, because there are less dedicated forums for the controller, finding schematics for the ESP proved to be a challenge, schematics which were necessary to properly map certain pin locations for the LoRa chip to locate. Finally, the LoRa libraries have many built-in Arduino sketches that were built with Adafruit products in mind, meaning that the team would not have to spend countless hours attempting to learn how to use said products. As such, the Adafruit Feather M0 was chosen in favor of the ESP, since it also had good power mode options, included the RFM95 chip for LoRa communication and has in-built connectors for 1S batteries, relieving the need of further circuitry to include an LDO.

2.3.2.2 Battery

As stated above, because of the microcontroller, the team no longer required an LDO. By removing the LDO, the previously recommended battery size of 2000 mAh could be reduced. Choosing between a 1S and 2S battery was also solved, as the controller came built with a typical 1S battery connector. Finally, a capacity of 1200 mAh was deemed fine because one of the sensors, the IR, was removed from the design.

2.3.2.3 Sensors

The IR sensor was removed from the initial design because of 3 main reasons:

1. The IR consumes a lot of power, requiring the team to purchase larger batteries (2000 mAh vs 1200 mAh). Because a low-profile pod was preferred, getting a smaller battery would help.
2. The sensor itself is extremely large, requiring the pod to be large. Also, due to the nature of IR sensors, the transceiver and receiver must not be blocked for the IR signals to be transmitted and received for distance measurements. As such, the pod would be required to have small holes that would allow the sensor to transmit and receive. Doing so, however, would compromise the weather proofing of the system, as water or debris could enter the pod, potentially damaging the electronics within.
3. The sensor can only work properly if there is line of sight. In the case of snow (or ice), if the pod was completely encased in it, the sensor would not function as intended, potentially giving users false parking states constantly, reducing the accuracy of the system.

When the team was building the pods, the IMU was tested in various scenarios such as snow, being covered with plastic, etc, to determine if it could be the sole sensor for the system. End results of the testing determined that it was the case. Specific testing can be found in the next subsection.

2.3.3 Testing and Performance

Testing the IMU was mainly split up into two parts. Part 1 involved testing (with a magnet) saturation values for the IMU and testing the IMUs sensitivity when it's snowing, indoors, outdoors in the cold and under surfaces and plastic. Such tests were simple in nature and did not require any sort of tuning. Their main purpose was to determine if the IMU was suitable to act as the only sensing device.

Part 2 involved tuning the IMU for vehicles. Due to the nature of magnetic fields, changing the position of the IMU or moving close to it with a phone will change the surrounding magnetic field. To ensure that the magnetometer never has any false read, on startup (on an empty spot) the device will read the magnetic field for a few seconds and create a ‘not parked’ range of magnetometer values unique to its spot. After establishing a ‘not parked’ calibration range, said range is expanded to ensure that any changes in the field are due to vehicles and not something such as a person walking over the pod. Figure 4 expands on the idea with a visual.

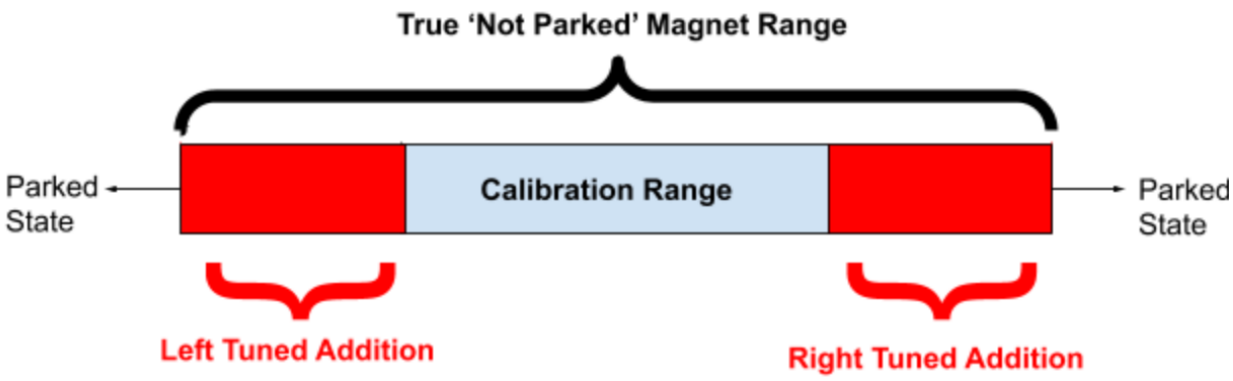


Figure 4: Visualization of Magnetometer Tuning

At this point, extreme fine-tuning of the device for cars comes into play. To start the tuning process of finding exactly what the 'left tuned' and 'right tuned' values are, the team placed the pod in an indoor parking lot (an empty spot) and with a members car, would slowly drive in and out of the spot (at different angles), observing how the values differ from the initial calibration scan. This would give the team an initial estimate on the true 'not parked' state. Further testing included viewing how the values change when a car parks beside an empty spot (where the pod is), when a car parks at an angle next to an empty spot, when there are two cars parked at angles facing the empty spot, etc. One thing to note as well, is that a vehicle's magnetic field is strongest at the front, where the engine is located. So a final test is doing the same tests mentioned above, but with reversing vehicles. Having too large of tuning value makes it so no vehicles are detected, too small and any minor field change is detected.

Doing the tests gave the team varying tuning values, and all the values would be loaded onto the node. The team would then test the pod to ensure that the chosen tuning values yielded correct parked states. Final tuning values were -4 (left tune) and +4.3 (right tune).

2.4 Communication

LPWAN (Low Power Wide Area Network) technology, specifically Semtech's LoRa (Long Range) Technology, was chosen due to its long range and low power consumption. LoRa claims a range of 2-5 km and 45 km in urban and rural regions respectively. Due to the power efficient and lower power consumption of the protocol, applications of LoRa consistently have longer battery life [4]. Designed for Internet of Things products, LoRa has the ability to handle thousands of connected devices.

2.4.1 Final Design Details

The Gateway chosen was the Dragino LG02, which supports up to 2 channel communication. Through extensive testing, the final configuration state chosen for the gateway and node was a spread factor of 7, using channel 10 (125Mhz), at a message frequency of every 30 seconds, sending a packet size of 14 bytes. The ABP device activation method was chosen, since the chosen gateway only accepts that form.

2.4.1.1 The Things Network

The systems communication with LoRa uses The Things Network (TTN) to 'host' its parking states, with the teams app grabbing the parking states and displaying it. LoRa only accepts binary data to transmit and includes a header of 13 bytes, meaning that 14+ bytes are always transmitted with a LoRa message. Because SLAP. only needs to send a parking state or 'parked' or 'not parked', binary data in the form of 1 hex byte (00 or 01) is sent to the TTN, yielding a payload size of 14 bytes. The TTN then decodes the message with a JavaScript function that the team wrote, and said decoded message is sent to the TTN dashboard where the app reads the state.

Because the TTN is used, the team has to comply with its specific transmission rules, one of which limits the amount a node can transmit to 30s per day (lightly enforced). Because airtime is an important metric for the pod and gateway communications, several terms will need to be explained for the testing section.

2.4.1.2 LoRa Channels, Bandwidth & Spread Factor

To determine the airtime that a single LoRa message will take, the spread factor and bandwidth of the message are important. The sections below will go into detail about what each term within LoRa means, and the testing section will provide more information regarding the tuning process used to determine gateway location.

2.4.1.2.1 LoRa Channels

Each pod uses 1 of 2 channels, as the chosen gateway is only a 2 channel one, meaning that it only receives messages from 1 of the 2 channels. Each channel, in groups of 8, lives within a sub-band in the LoRa frequency plans and each country has its own unique set of sub-bands, with the US and Canada using the 915Mhz frequency. Selecting the correct band (and channel) for the nodes is important since it determines which bandwidth the gateway would receive. Furthermore, knowing the frequency the band/channel works in is important as the gateway looks for those exact frequencies when looking for node messages.

2.4.1.2.2 Bandwidth

Most LoRa sub-bands typically transmit in the 125Mhz uplink frequency, with only a few specific bands going under or over this figure. The uplink frequency, or bandwidth, of a message directly affects the speed (or airtime) that a message gets to its receiver. By increasing the bandwidth, messages arrive much faster but consequently, the effective range of LoRa drops as higher frequency signals are blocked more easily by structures and air. Decreasing the bandwidth does the opposite.

2.4.1.2.3 Spread Factor

When transmitting messages, LoRa converts its binary data into chirp signals that span the specified bandwidth. The chirp time can be controlled with the spread factor configuration. An increase in spread factor, then, increases the on-air time of a message but having a chirp run for longer can remove potential noise effects, allowing the signal to be read more reliably. Essentially, increasing the spread factor has an

inverse effect when compared to bandwidth. Note that spread factor has a larger effect on both non-obstructive signal range and signal airtime, when compared to bandwidth.

2.4.1.3 Gateway

As stated previously, the gateway chosen for the system was the Dragino LG02, a 2 channel gateway. Because the project only required the team to build a few nodes for demonstration, a cheap gateway that can support at least 3 nodes was required. In the case of LoRa gateways, the fewer channels a gateway has, the cheaper they tend to be. The LG02 is purported to be able to support up to 300 sensor nodes as per their user manual [9]. As such, the LG02, being cheaper than the TTN gateway, was deemed suitable for the project's needs.

Due to the gateway being cheap, there were some LoRa features missing from it. One such feature is its form of device activation. In LoRaWAN, for a device/node to communicate with a gateway (and the gateway to the TTN), a specific connection pattern is required by said devices. There exist two methods; Over The Air Activation (OTAA) and Activation By Personalization (ABP). Essentially, OTAA works by devices starting a join procedure with the network, where dynamic device addresses are created and security keys negotiated. ABP works, instead, with static device addresses and hard-coded keys, leading to a less secure system. The LG02 does not come with OTAA support however, this was deemed as acceptable since the nodes do not/would not contain any high-risk data, only parking states.

When the gateway is first turned on, it creates a network that can be connected to. Connecting to its network and accessing the gateways IP address leads to the configuration page of the gateway, where configuration values such as its spread factor or wifi-connection can be changed. Within the configuration options, there is the option to connect to a WiFi network, which is necessary so that the gateway can connect to the TTN. By modifying the gateway to transmit to the TTN, and registering a new gateway in the TTN, the gateway will send its uplink messages to it, where the parking app can then grab parking states.

2.4.2 Testing and Performance

Once it was confirmed that the gateway was fully functional and able to send out signals to the TTN, testing to determine the effective range and packet drop rate was performed.

The first step when beginning the testing process was determining an area where the gateway could remain for the symposium. Such a location would require a stable WiFi network, be close to campus and easily accessible at any time of the day. Initially, the team attempted to connect to eduroam, however, it was soon discovered that the network did not support device-to-device connections, meaning that the gateway could not connect to it. After asking around both on and off-campus, the team settled for the salon Beauty Time located in the plaza, with the owner's permission. The location was used as the gateway’s hub until it was announced that an engineering capstone network was being established. When that occurred, the team decided to keep the gateway on campus, near the group's booth, for quick configuration access.

The second step of testing was to determine the effective distance of the gateway. As stated previously, multiple factors affect effective distance, including antenna quality, elevation, antenna angle and radio obstacles of both the nodes and gateway. As such, when determining the effective distance, all such factors and the configuration factors (spread factor, etc) were all modified continuously to determine the best possible range that the gateway could get. LoRa is a protocol that can reach multiple kilometers of range if the antennas of the receiver and transmitter are in a line of sight, due to the effect of Fresnel Zones. Because the university plaza is full of buildings, the effective range of our gateway within the Beauty Time salon only reached a maximum of approximately 300m with a packet drop rate of about once every 30 minutes (meaning a message is lost every 30 minutes). The chosen configuration is the one mentioned in section 2.4.1. Using the rate limit of TTN, this yielded a message capacity of approximately 1200 messages per day or 10 hours of operation per day. In an open parking lot, a more ideal situation than communicating from beauty time to a node in Gear lab, a higher bandwidth could be used. Doing quick calculations with the Semtech LoRa calculator, this yields a value of approximately 1800 messages (at 500Mhz) or 15 hours of operation. By changing the rate that a state is sent to the server or creating a dynamic rate that changes based on time of day, getting a 24-hour service is very doable. For a quick example, increasing the rate from 30 seconds to 48 seconds yields 24 hours of operation per day.

2.5 Web Application and UI/UX

2.5.1 Final Design Details

2.5.1.1 Database and Server Code

The database and server were designed around two main goals. The first was to minimize the required time between a parking update arriving from The Things Network, and updated information being sent to the client. The second goal was to minimize the time needed for queries in the analytics view, which could be quite complicated. The list of tables and their respective fields can be found in Table E1, Appendix E.

This storage configuration allows the following information flow to be used to update the database when a parking update arrived.

1. Add entry to “Events” table
2. Update parking space occupancy for device in “Devices” table
3. Query Devices in Lot for Lot Occupancy, Lot for total occupancy
4. Send updates to client

Relational databases use locks to ensure information gets updated properly. When the database performs a read, the tables being accessed cannot be written to. When the database performs a write, the tables being accessed cannot be read from or written to.

This approach minimizes the amount of writing to the database that is required by not saving current lot occupancy to a table, thus avoiding the associated locks.

The configuration also isolates the information required to perform analytics queries to a single table, which can be indexed to speed up query performance.

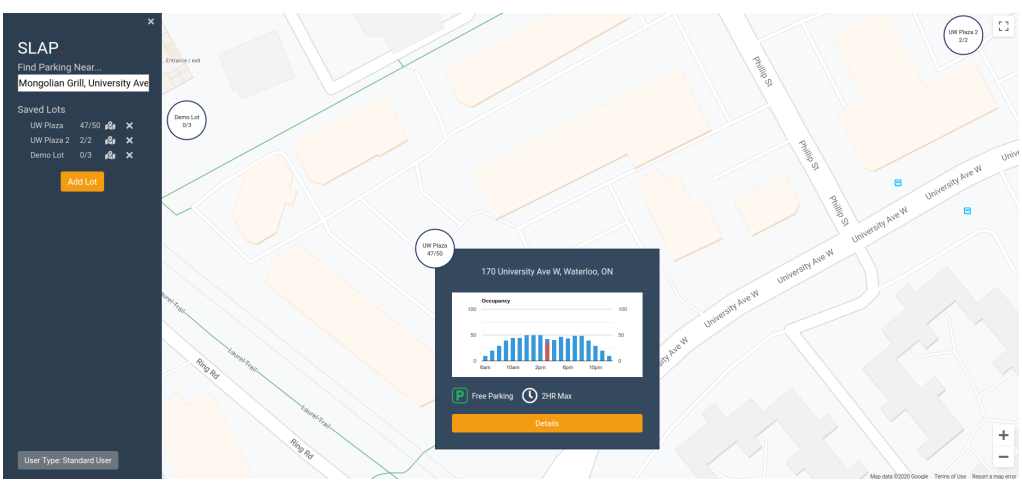
2.5.1.2 Client Application and UI/UX

The web application was designed with multiple goals in mind for use by both Drivers, and Parking Lot Owners. Design goals can be found in Table 2 below.

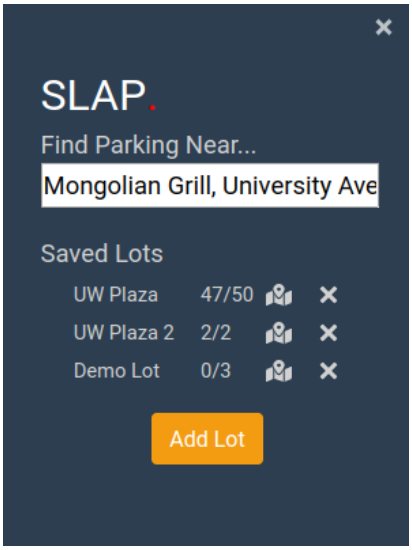
Table 2: Web Application Design Goals

Drivers	**Parking Lot Owners**
1. Provide drivers with live updates on parking lot occupancy 2. Show drivers parking lots near their desired destination 3. Allow for easy access to favorite parking lots 4. Provide drivers with other important information, such as historical trends regarding lot occupancy	1. Provide lot owners with useful visualizations of lot occupancy trends 2. Provide lot owners with usage summaries to direct their trend analysis

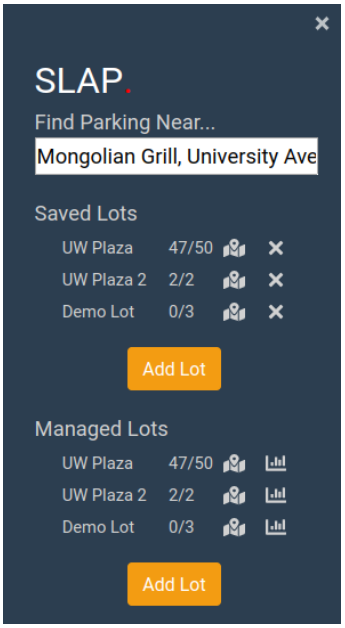
For drivers, the final design consists of two main components. The parking lot finder view pictured in Figure 5 contains a search bar, allowing drivers to search for a destination, and have nearby parking lots appear. Frequently visited lots can be saved, appearing in the side bar for easy access. These lots can be located using the map icon, or removed from the saved list using the "x" button.



(a) Parking Lot Finder View



(b) Sidebar for Drivers



(c) Sidebar for Lot Owners

Figure 5: Popup and Sidebar Components

If a user clicks on any of the parking lots, a box will pop up showing general occupancy information. Clicking the "details" view will take the user to the parking lot view, pictured in Figure 6.

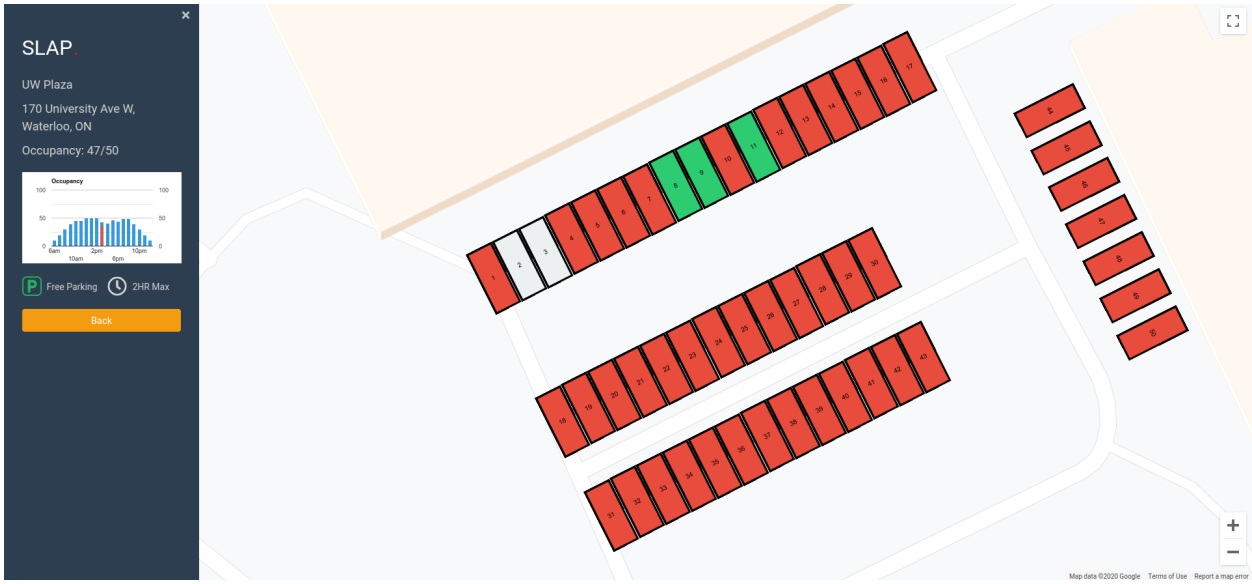


Figure 6: Parking Lot View

The parking lot view shows general lot information in the sidebar, while using the map section to show the current occupancy status of each space in the parking lot. Occupancy status is represented by the

color of the parking space (Green=Free, Red=Occupied, Grey=Inactive/Needs Servicing). Current occupancy status was reflected using a web socket connection to the server for real-time updates.

Parking lot owners have an additional menu in the sidebar, showing them all of the lots that they own. Clicking the chart icon near any of these lots will take the user to the analytics view, shown in Figure 7.

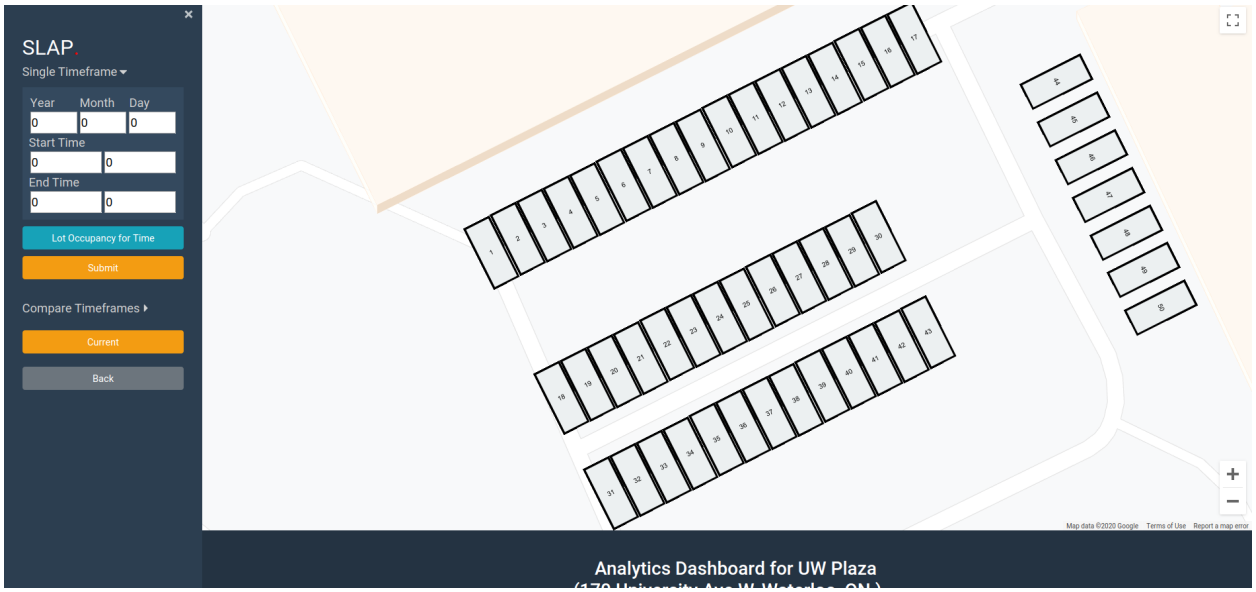


Figure 7: Analytics View

The analytics view consists of three main components. If lot owners wish to view lot occupancy during a specific time frame, they can do so using the first dropdown in the sidebar. This could provide lot owners insight into space usage during peak hours, or general trends about which spots in the parking lot tend to be used more often.

If lot owners want to compare occupancy during different time frames, they can do so using the second dropdown in the sidebar. This could provide lot owners insight into when the lot tends to be busy on a space-by-space basis. An example of this can be found in Figure 8.

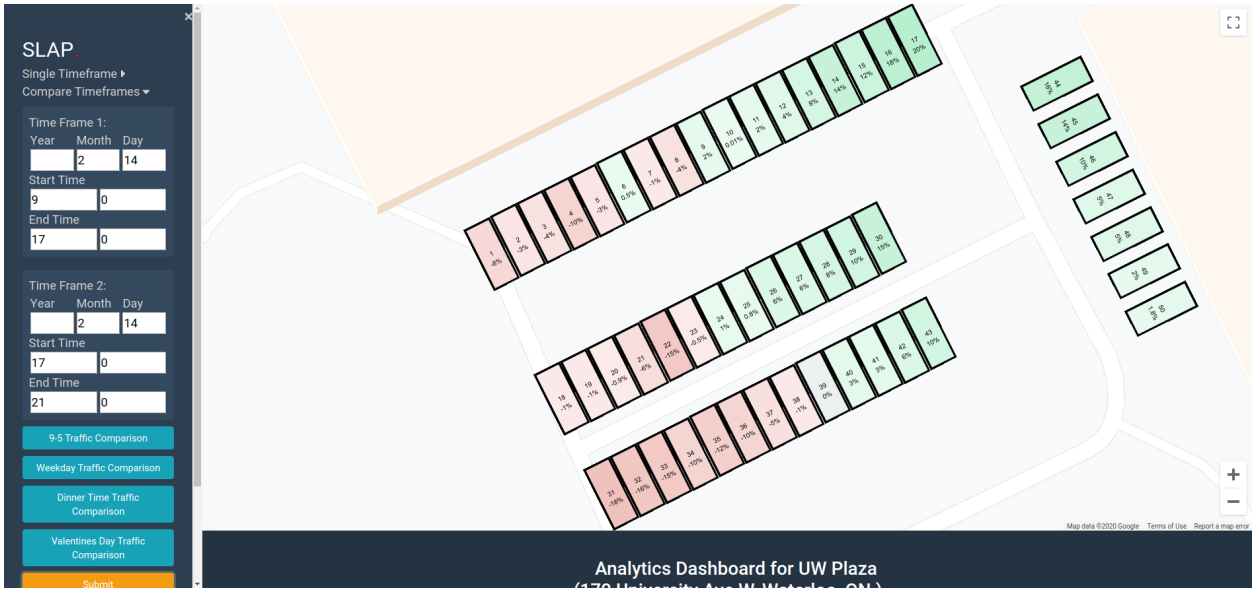


Figure 8: Example Time Frame Comparison

The example comparison of two time frames above shows that the lot tends to be most busy around Morals Village Hotpot during dinnertime. In terms of parking usage, it would be optimal to place other dinner restaurants in different areas around the plaza.

Finally, to help direct the analytics of lot owners, a general analytics section is included, showing multiple charts with general lot usage information. Some of these charts can be seen in Figure 9.

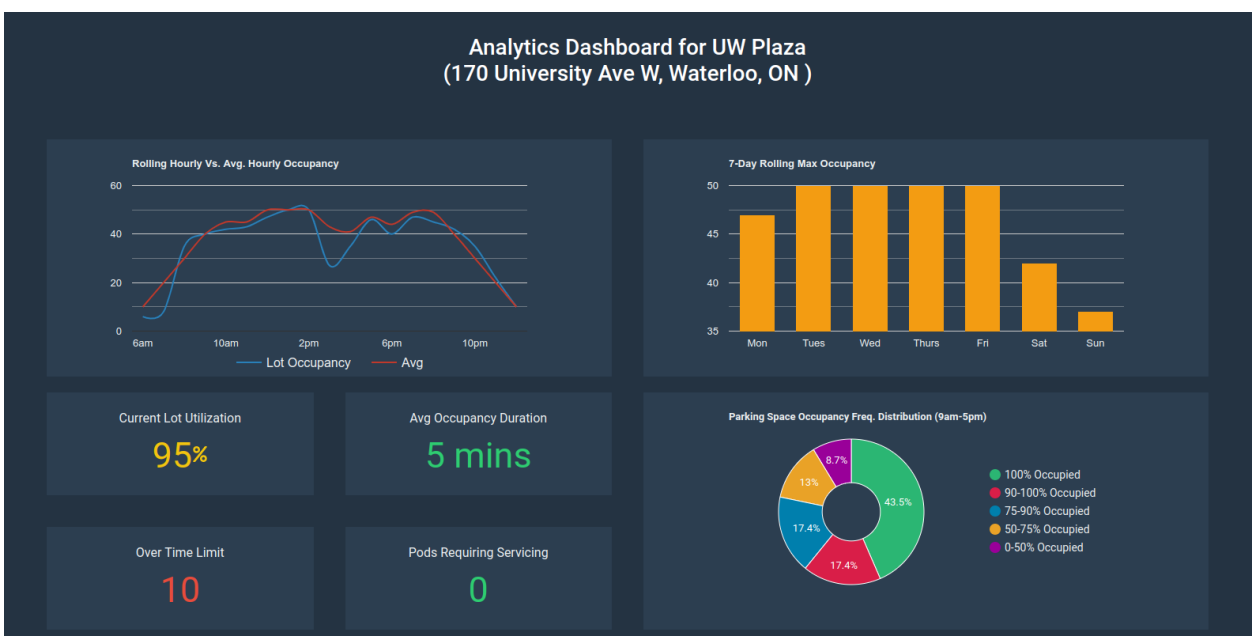


Figure 9: General Usage Information to Direct Analytics

The goal of these charts is to keep lot owners up to date on important information, such as drivers staying in the parking lot over their allotted time limit, and pods in need of servicing, as well as provide useful summary analytics to help lot owners direct their more in depth analysis. Some of these charts include rolling hourly occupancy in comparison to the average, rolling daily max occupancy, and a breakdown of how long vehicles tend to stay parked for.

2.5.2 Modifications/Deviation from Original Design

The final design ended up fairly similar to the original design, with minor differences to help achieve the goals stated in Section 2.5.1.

With the aim of providing drivers with other important information, such as historical trends regarding lot occupancy, the details popup was created (shown in Figure 5a), as well as the addition of the extra information from the details popup to the sidebar (shown in Figure 6).

With the aim of making the in depth analytics easier to perform for parking lot owners, the "Single Time Frame" option was added to the analytics view (shown in Figure 7).

Finally, to provide lot owners with usage summaries to direct their trend analysis, the charts and general usage information section was added in Figure 9.

2.5.3 Testing and Performance

The new UI design for the application was universally decided to be easier to use than the old design by the team, as well as multiple third parties.

Web socket reliability and performance was tested by generating fake messages from The Things Network at specified time intervals, and timing the delay between message creation and the corresponding UI update. During tests of the parking lot view, the update latency due to the server was negligible, and occupancy status of parking spaces was consistently updated well within the 1 minute timeframe specified in the constraints. All messages received by the server were reflected by the appropriate updates in the User Interface.

To test the accuracy of analytics view visualizations, sets of parking data were created, with expected visualization results. These sets of parking data were manually loaded into the database, and results from the analytics view would be manually compared to expected results.

Presentation of the User Interface to Profs. Kennings and Khamesee appeared to yield positive results.

3. Schedule and Budgeting

This section outlines the project management aspects of the final designed system. It details major changes in the schedule and budget as well as the final cost breakdown.

3.1 Schedule

The proposed project schedule consists of project milestones and team goals proposed for the 2020 school year (Table C1, Appendix C). The actual schedule has been updated in accordance with team progression and deadlines occurred in winter 2020 (Table C2, Appendix C). The actual and proposed schedules are different in comparison due to temporary setbacks, longer turnaround time and shipping delays. Nevertheless, the team delivered a fully furnished prototype as promised, and had a successful demonstration in a timely and professional manner.

Mechanical schedule progressed as planned but was held back near mid-February due to hardware modifications and other school commitments. It was further delayed during the construction phase where the turnaround time for 3D printing was longer than anticipated. The total delay for the mechanical refinement was 5 days from the proposed date. Given enough time pre-planned for such scenarios, the temporary delay had no impact on the overall system performance or delivery. The mechanical module was released a week before March 6, 2020, allowing the team enough time to test, integrate and showcase the module for the live demonstration.

Regarding the hardware schedule, there were quite a few setbacks that delayed the overall completion of the pods hardware before the initial target of February 17th. The first setback occurred in November, where the initial hardware order was placed using BangGood. Because the order included a battery and the order originated from China, there was a significant delay in the product reaching the team. In the end, the ESP32s and battery arrived in late December, after the end of 4A, requiring the team to commence the hardware phase of the project in 4B. As stated previously, the ESP32 was replaced with the Feather M0 after the team deliberated on hardware designs for a few days, pushing the final hardware design of the project further back. The final delay the team experienced was in the form of extensive testing of the IMUs magnetometer and LoRa range, with the team testing multiple antennas and ranges with a test pod before final construction commenced. Despite all the shortcomings, using the schedules provided in Appendix C, the hardware portion commenced refining after demo day, meaning it was approximately 2 weeks delayed from the original schedule.

The web application and software infrastructure could be done relatively independent of the other aspects of the projects. Development started earlier than anticipated with a strong focus on getting the base application, data storage, and integration with The Things Network working properly. In this manner, whenever the pods could send data, the information could easily flow into the application without impeding development on other parts of the application. Due to rapid development and changes in the application's data pipeline, the team was required to refactor twice. This slowed down the development of core and additional features. In the end, the software components of the project were ahead of schedule and the team was able to be more detailed and build additional features.

3.2 Budgeting

The project budget is capped at $1,000 CDN, outlined in the constraints from section 1.4.1. The actual project cost is $693.18 CDN in total with tax included (Table D1, Appendix D). The expected cost is estimated at $564.02 CDN outlined in the MTE 481 report (Table D2, Appendix D). The actual project cost differs from the expected cost but is still within the budget cap outlined in the constraints. The actual cost has a 22.9% increase from the expected cost. Hardware design modifications, rush orders and supplementary expenses are attributable to the cost difference. Majority of the cost goes into the sensor hardware development and the LoRa communication device. Supplementary costs include the poster printing charge and service parking permit prepared for the symposium.

4. Conclusions

The final design meets the objectives and requirements proposed in MTE 481 and, above all, it solves the need for drivers to find parking spaces and for parking managers to track parking trends in a more efficient manner.

The final design details, modifications from original designs, construction, and testing process are thoroughly discussed for the system architecture, pod module chassis, pod module hardware, communication, and web application. Subsequently, the commissioning of the prototype system is investigated. The solution is a connected system of sensor modules, gateways, a network server, and a web application. The pod modules are physically placed in parking spaces and use a BNO055 IMU sensor for vehicle detection. Each sensor module has an ESP32 microcontroller with a RFM95 chip to broadcast parking occupancy with the LoRaWAN protocol to a LoRa Gateway. Each pod module is also equipped with a 1S 1200 mAh battery, expected to run the pod for approximately a year. Each pod module also has a low-profile 3D printed chassis designed to withstand loading from medium-duty vehicles. The waterproof chassis can be installed either on the road surface or underground. Parking state updates are broadcasted every 30 seconds with a spread of 7. The gateway is connected to a WiFi network and sends the parking updates to a network server. The network server hosted by The Things Network handles the data coming from the LoRa devices and routes it to the correct application. An Express and React.JS web-application was developed to visualize and provide analytics for the parking state in real-time. The application maintains the connection to the network servers and continuously listens for any updates received on the network server. A PostgreSQL database is used to store information about the parking lots, devices, and event data to generate analytics.

In terms of project management, the project takes a total of seven months to complete and it exceeds expectations while all the deliverables and goals are accomplished in a timely manner. The project cost totals to $693.18, well within the constraint of $1,000 set forth initially.

5. Recommendations

This section outlines potential improvements that can be accomplished in the future iterations. These recommendations are categorized in: mechanical, hardware, software and web application.

5.1 Mechanical

The mechanical system of the project has some areas for potential improvement. The final prototype chassis allows extra housing space for the hardware components. This space can be further reduced to enable a more compact chassis design if a custom PCB circuit is used for the hardware. By reducing the vertical chassis profile and the contact surface area atop, it would require less material overall for structural support under load.

Next, the manufacturing process can be improved to further strengthen the chassis with less support material while the final prototype chassis meets design requirements and performs beyond expectations in all the tests and demo. For printing wall structures, Current 3D printing process only allows certain infill options such as infill percentage and standard infill patterns. Using more advanced 3D slicer programs such as Cura 4.5 and above would allow stronger infill patterns like cubic, 3D honeycomb or gyroid infill patterns to enhance the strength. However, the printing cost would drastically increase as a result. Alternatively, injection molding would be the best for mass production due to its consistency for producing cheaper and stronger products.

5.2 Hardware

Although the final product performed effectively and reliably, there are a few project hardware improvements that both streamline the setting up of pods and create a better performing system.

One such recommendation would be the use of a better gateway. This addresses two main issues with the current system in use. First, by using the TTN gateway, the amount of nodes per gateway is no longer limited to 300. At the worst case, 1200 nodes (300 * 4) can be supported and high end gateways support up to 62000 nodes [10]. This means that the system can work with larger parking lots, no hassle, without the need for multiple gateways. Second is the fact that the TTN gateway supports OTAA, for more secure packet transmission, has LBT (Listen Before Talk) support which some countries enforce/mandate with some communication devices and easier support remote ssh, meaning that the team does not need to be near the gateway in order to configure it. A final recommendation, for price reasons, is to use a dedicated magnetometer instead of an IMU.

5.3 Web Application

While no part of the web application architecture caused any issues for the demo, it is expected that the increased traffic of a full deployment would have caused two main issues.

First, the current Database architecture has updates to the status of each parking space stored as events in a single database table. The increased number of parking lots from a large-scale deployment of this product would lead to an exponential increase in the number of individual events that would need to be inserted into the table, causing significant decreases in query performance. A solution to this problem would be to move the event tracking from the database to a service designed to track and query time-series data, such as InfluxDB[11]. This provides the benefits of easy scalability and optimized time-based queries.

Second, the current client update architecture has the server send an update to the client each time a parking state changes. For clients with many large parking lots, this could require a significant number of updates. It may be helpful to look into batching some of these updates together, so multiple updates to the same parking lot can all be sent in the same message.

6. Teamwork Effort

John was the project manager and mechanical lead. He was responsible for the commissioning of the mechanical module, and for ensuring that the project delivered on its promises without delay. In addition, he supported and collaborated with Kevlin in the hardware development and testing.

Kelvin worked on the LoRa communication and hardware components. Kelvin led the electrical development of the pod, primarily dealing with the circuit, battery, and integrating sensors.

Hanson and Michael worked on the LoRa communication and the web application. This includes database design, user interface design, and implementation.

References

[1] “Vehicle registrations, 2016", Statistics Canada, (2017, Jun 29). [Online]. Available: https://www150.statcan.gc.ca/n1/daily-quotidien/170629/dq170629d-eng.htm. [Accessed Oct 28, 2019]

[2] "Searching for Parking Costs Americans $73 Billion a Year", Inrix, (2017, Jul 12). [Online]. Available: http://inrix.com/press-releases/parking-pain-us/. [Accessed Oct 28, 2019]

[3] "Lamborghini Aventador Ground Clearance". Autoportal (n.d.) [Online]. Available: https://autoportal.com/newcars/lamborghini/aventador/feature/ground-clearance.html [Accessed Oct 11, 2019]

[4] J. Silva, J. Rodrigues, A. Alberti, P. Solic, “LoRaWAN - A Low Power WAN Protocol for Internet of Things: a Review and Opportunities”, In Proc. International Multidisciplinary Conference on Computer and Energy Science, (2017, Jul.). [Online]. Available: https://www.researchgate.net/publication/\318866065_LoRa WAN_-_A_Low_Power_WAN_Protocol_for_Internet_of_Things_a_Review_and_Opportunities [Accessed Sep 15, 2019]

[5] “SLAP.,” SLAP. [Online]. Available: https://slapadot.weebly.com/. [Accessed: 27-Mar-2020].

[6] “PLA vs. ABS: What's the difference?,” 3D Hubs. [Online]. Available: https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference/#strength. [Accessed: 27-Mar-2020].

[7] “Thermoplastic,” Polylactic Acid (PLA, Polylactide) :: MakeItFrom.com, 20-Sep-2018. [Online]. Available: https://www.makeitfrom.com/material-properties/Polylactic-Acid-PLA-Polylactide. [Accessed: 27-Mar-2020].

[8] Makerbot PLA and ABS Strength Data, Makerbot Available at: https://eu.makerbot.com/fileadmin/Inhalte/ Support/Datenblatt/MakerBot, PLA_and_ABS_Strength_Data.pdf [Accessed: 27-Mar-2020]

[9] “LG02/OLG02 Gateway User Manual”, Dragino, (2018, Sept 11). [Online]. Available: http://www.dragino.com/downloads/downloads/LoRa_Gateway/LG02-OLG02/LG02_LoRa_Gateway_User_Manual_v1.5.1.pdf. [Accessed: 23-Mar-2020]

[10]“LoRa Module FAQ,” *How many LoRa nodes can be connected to one gateway? Can LoRa nodes talk to each other? Is mesh topology possible in LoRa? | Murata Manufacturing Co., Ltd.* [Online]. Available: https://www.murata.com/en-us/support/faqs/products/lpwa/lora/hardware/0009. [Accessed: 23-Mar-2020]

[11]S. Naik, R. Tomer, D. Brazil, F. Rampp, D. McLean, and R. Allen, “InfluxDB: Purpose-Built Open Source Time Series Database,” InfluxData, 27-Mar-2020. [Online]. Available: https://www.influxdata.com/. [Accessed: 28-Mar-2020].

Appendix A: As-built Drawings

This section contains the technical drawings of the mechanical module.

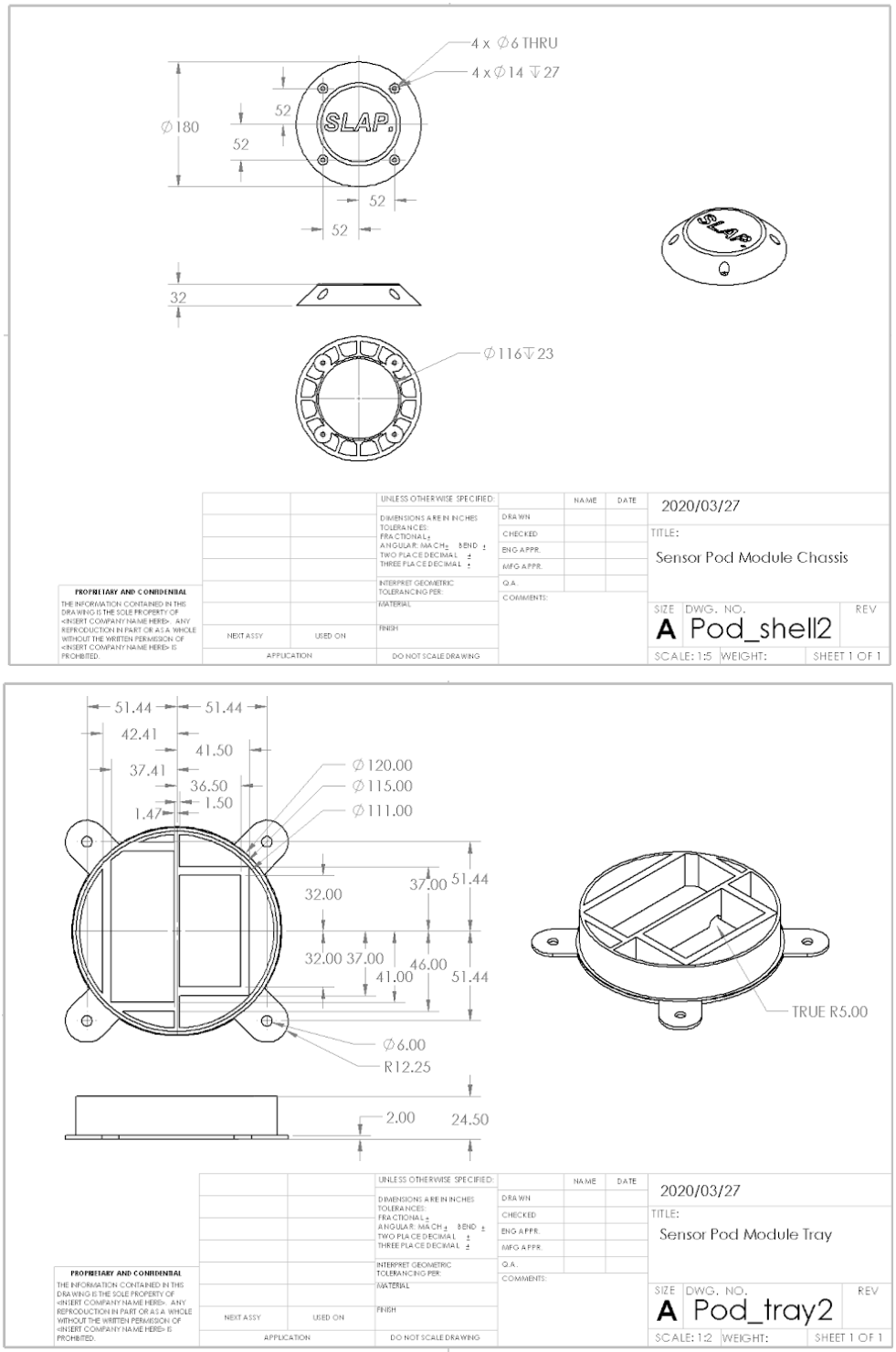


Figure A1: Sensor Pod Module Chassis Drawings

Appendix B: Simulation Results

This section contains the final and original static analysis results for comparison.

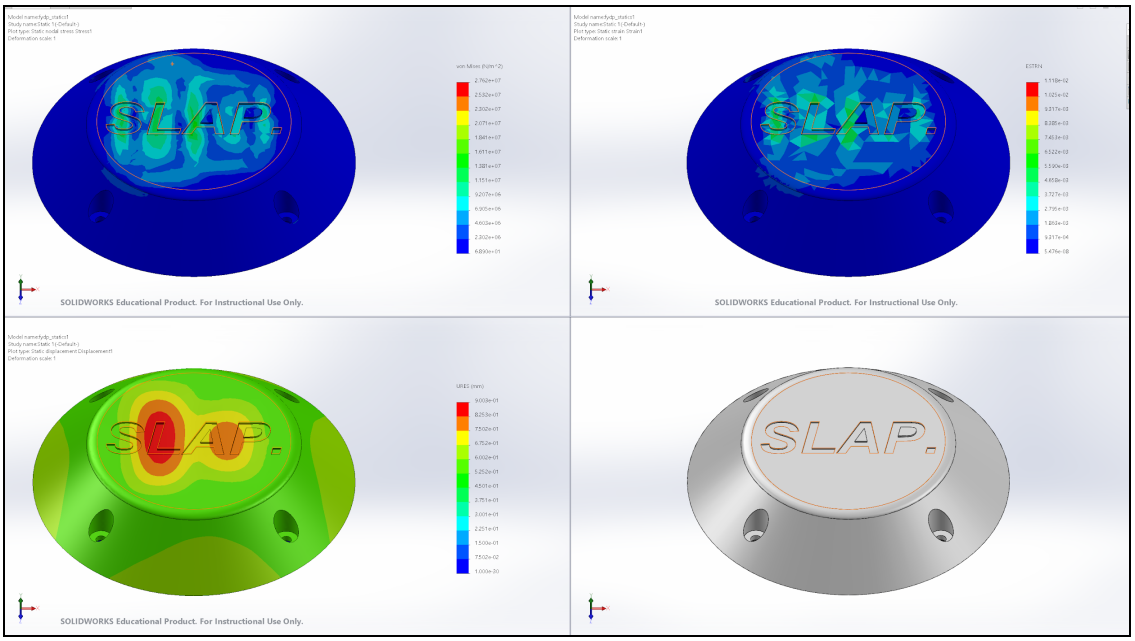


Figure B1: Von Mises Stress, Strain and Displacement of the Final Design

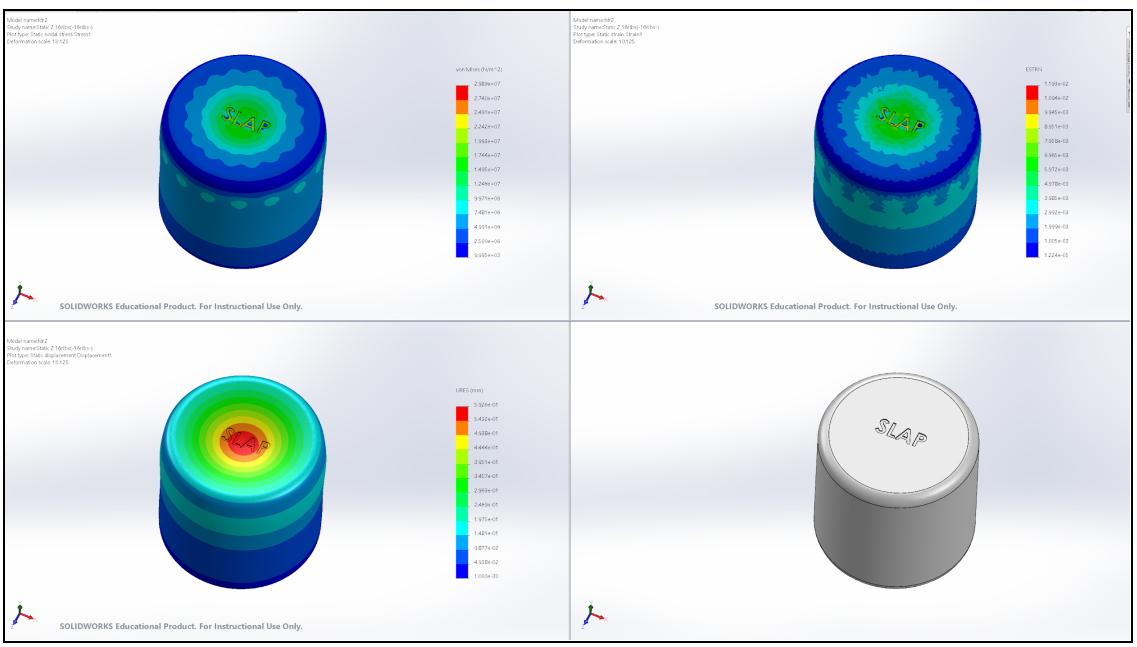


Figure B2: Von Mises Stress, Strain and Displacement of the Original Design

Appendix C: Project Schedules

This section displays the final project schedule alongside with the proposed schedule for comparison.

Table C1: Proposed Project Schedule

Schedule (4A)	Start Date	End Date	Current Timeline/Date	Status	Type
Project Duration	2019-09-06	2020-03-20			
Team Goals					
Brainstorming Ideas	2019-09-04	2019-09-14		Complete	**General**
Problem Definition	2019-09-06	2019-09-19		Complete	**General**
Patent Review	2019-09-19	2019-09-26		Complete	**General**
Concept Solutions	2019-09-26	2019-10-03		Complete	**General**
Design Selection	2019-10-03	2019-10-10		Complete	**General**
PDP	2019-10-03	2019-10-28		Complete	**General**
Design Solutions	2019-10-28	2019-11-07		Complete	**General**
Design Refinement	2019-11-07	2019-11-14		Complete	**General**
FDP	2019-11-07	2019-11-18		Complete	**General**
Report	2019-11-02	2019-12-09		Active	**General**
Procurement	2019-11-01	2019-11-29		Active	**General**
Build	2019-11-30	2020-01-20		Upcoming	**General**
Integration	2020-01-20	2020-01-27		Upcoming	**General**
Debugging	2020-01-27	2020-02-04		Upcoming	**General**
Testing/QA	2020-02-04	2020-02-17		Upcoming	**General**
Hardware Refining	2020-02-17	2020-02-24		Upcoming	**Electrical**
Software Refining	2020-02-24	2020-02-28		Upcoming	**Software**
Mechanical Refining	2020-02-24	2020-03-01		Upcoming	**Mechanical**
FInal Push	2020-03-01	2020-03-13		Upcoming	**General**

Table C2: Actual Project Schedule

Schedule (4B)	Start Date	End Date	Timeline	Status	Type
Project Duration	2020-01-07	2020-03-22			
Project Milestones					
Advisor Meeting 1	2020-01-16	2020-01-16		Complete	Deadline
Advisor Meeting 2	2020-02-06	2020-02-06		Complete	Deadline
Live Demostration	2020-03-06	2020-03-06		Complete	Deadline
Symposium	2020-03-13	2020-03-13		Complete	Deadline
Report	2020-03-19	2020-03-20		Complete	Deadline
Team Goals					
Build	2020-01-07	2020-02-14		Complete	General
Integration	2020-02-14	2020-03-06		Complete	General
Debugging	2020-02-18	2020-03-06		Complete	General
Testing/QA	2020-02-21	2020-03-06		Complete	General
Hardware Refining	2020-02-24	2020-03-06		Complete	Electrical
Software Refining	2020-02-25	2020-03-06		Complete	Software
Mechanical Refining	2020-02-24	2020-03-06		Complete	Mechanical
FInal Push	2020-03-06	2020-03-13		Complete	General

Appendix D: Cost Breakdown Summaries

This section displays the final bill of materials alongside with the initial cost estimate for comparison.

Table D1: Actual Project Cost

Part Names & Descriptions	**Unit Price**	**Currency**	**#Units**	**Total**
ESP32 LoRa, Adafruit Feather	$84.00	1	1	$84.00
ESP32 LoRa, Adafruit Feather	$34.95	1.31	2	$91.57
ESP32 LorRa, LilyGo	$30.20	1.31	1	$39.56
LoRa Gateway, Dragino LG02 915 MHz	$108.21	1	1	$108.21
IR sensor, no brand	$14.95	1.31	3	$58.75
IMU, Adafruit	$34.95	1.31	2	$91.57
3.7V, 1200 mAh LiPo, no brand	$9.95	1.31	3	$39.10
DC Switching Regulator (7.4), no brand	$32.74	1	1	$32.74
Antenna, no brand	$0.95	1.31	3	$3.73
LiPo Charger, no brand	$29.20	1.31	1	$38.25
Filament, Hatchbox PLA	$28.50	1	2	$57.00
Conference poster	$48.69	1	1	$48.69
Parking pass	$5.00	1	1	$5.00
TOTAL (Tax incl.)				$693.18

Table D2: Project Cost Estimate

Parts	**Cost (CAD)**	**#Units**	**Total**
ESP32 LoRa, OLED	$20.10	4	$80.40
LoRa Gateway, Dragino LG02 915 MHz	$95.76	1	$95.76
IR sensor	$11.79	4	$47.16
IMU	$34.95	4	$139.80
3D printed chassis	$5.00	4	$20.00
3.7V, 3000 mAh LiPo	$25.21	4	$100.84
LDO (for 3.7V)	$10.58	4	$42.32
DC Switching Regulator (7.4)	$21.01	4	$84.04
TOTAL (Tax Incl.)			$564.02

Appendix E: Database Schema

Table E1: Database Schema

Table	All Fields	Primary Key	Foriegn Keys
Devices	Device_id, lot_id, active, installed_date, maintained_date, lat, lng, rotation_degrees, updated_date, occupied	device_id	lot_id
Events	Id, device_id, time, detected	id	device_id
Lots	Lot_id, capacity, created_at, lat, lng, address, name, occupancy, notes	lot_id	