UBC Social Ecological Economic Development Studies (SEEDS) Sustainability Program

Student Research Report

Design Document for Dynamic Parking Project

Saurav Choudhury, Anthony Csikos, Graham Judd, Alicia O'Sullivan, Nicholas Windt

University of British Columbia

ELEC 491

Themes: Transportation, Community, Wellbeing

April 4, 2019

Terms Appendix:

Term	***Definition***
User	UBC Parking Admin
Client	UBC Parking
Device	Parking stall display and Monitoring Device
FR	Functional Requirement, (outlined in requirements documentation)
NFR	Non Functional Requirement, (outlined in requirements documentation)
C	Constraint, (outlined in requirements documentation)
IP65	IP rated as "dust tight" and protected against water projected from a nozzle
LoRaWAN	Long Range Wide Area Network
DEVEUI	Unique device ID used to communicate with LoRa devices
SPI	Serial Peripheral Interface
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

Content

The purpose of this document is to specify and justify the design choices made in order to successfully complete the prototype/system.

1. High Level System Description

1.1 High Level System

Based on client requirements the system solution has been broken down into two main components:

- The Web Management System, from which the user can remotely monitor the occupancy, change the display, and in general manage the stalls. The communication link between the web app and the physical display and monitoring devices will be over LoRaWAN, a wide area network technology intended for low powered IoT (Internet of Things) devices.
- The Parking Stall Display and PlacePod (a monitoring device installed in the ground of the parking stall) work together to display various parking restrictions according to the need. The stall also transmits parking data over LoRaWAN, in order for the UBC Parking Administration to view.
- Figure 1.1 below shows a diagram of this high level system.

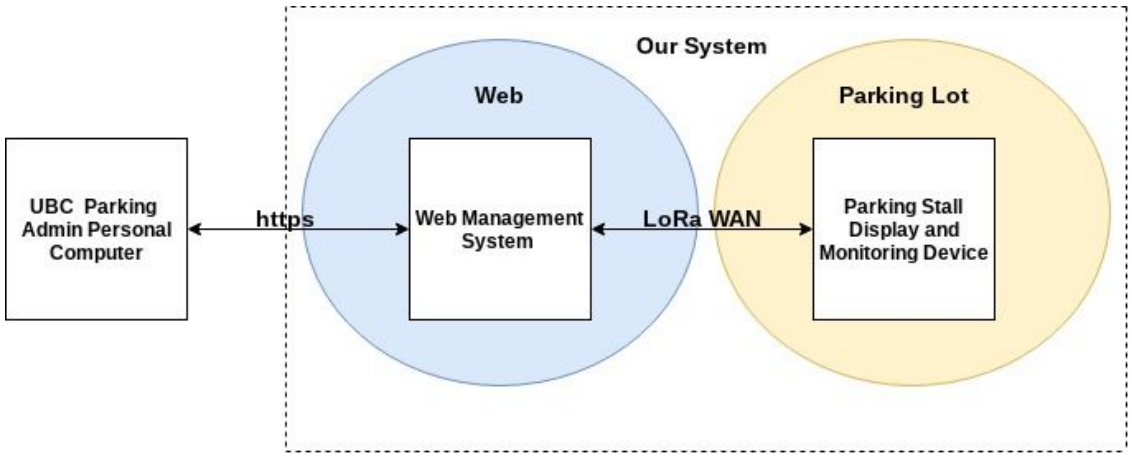

Figure 1.1 - High Level System Diagram

1.2 Web Management System

As previously explained, the web management system will be the interface that the user, UBC Parking Services, will use to manage the dynamic parking stalls. This system will communicate remotely with the Parking Stall and Monitoring Devices to update, set stall restrictions, and receive stall data. The core features include:

- Homepage to display all key stall information so the client can monitor the stalls
- Manual and automatic changes to stall restrictions
- Pre-loaded images and custom license plate restriction options

1.2 Parking Stall Display and Monitoring Device

The Parking Stall Display and Monitoring Device will be the end-system device present at each individual dynamic stall. Its function is to display parking restrictions and monitor the stall. Core features include:

- Displaying configured parking restriction
- Detecting cars parked in the stall
- Detecting vehicle impacts with sign post

1.3 LoRa

LoRa (Long Range) is a long range digital wireless communication technology which utilizes the license-free sub-gigahertz radio bands (915 MHz in North America). LoRaWAN (Long Range Wide Area Network) is the network over which LoRa operates. LoRaWAN enables transmission over very long distances.

LoRa was rated as the best choice for wireless communication in our system design over our second choice, cellular LTE, for the following reasons:

- Low powered - LoRa radio power consumption is much less than LTE cellular. Additionally different classifications of LoRa devices can be implemented, through radio sleep cycles, for even better power consumption.
- Inexpensive - LoRa utilizes unlicensed radio bands so no subscriptions are needed for network use; this scales much better cost wise than LTE
- Infrastructure at UBC - Our client currently has a LoRa system set-up for car detection at parking stalls around UBC. This system features deployed gateways connected to the Eleven-X DASS which acts as middleware between LoRaWAN and the internet. This provides us with a trusted, proven and already deployed sub system to ease integration of the Web Management System and the Parking Stall Display and Monitoring Device.

These advantages over LTE do come at cost when it comes to bandwidth. In practice, LoRaWAN payloads should be limited to a maximum size of only 8 bytes and LoRa enabled devices should not be transmitting more than once every two minutes. Infringing these guidelines can lead to network congestion and dropped packets. For our system high bandwidth is not needed as only small amounts of data need to be transmitted, (~1-7 bytes) at intervals generally greater than 5 minutes.

2. Low Level System Description

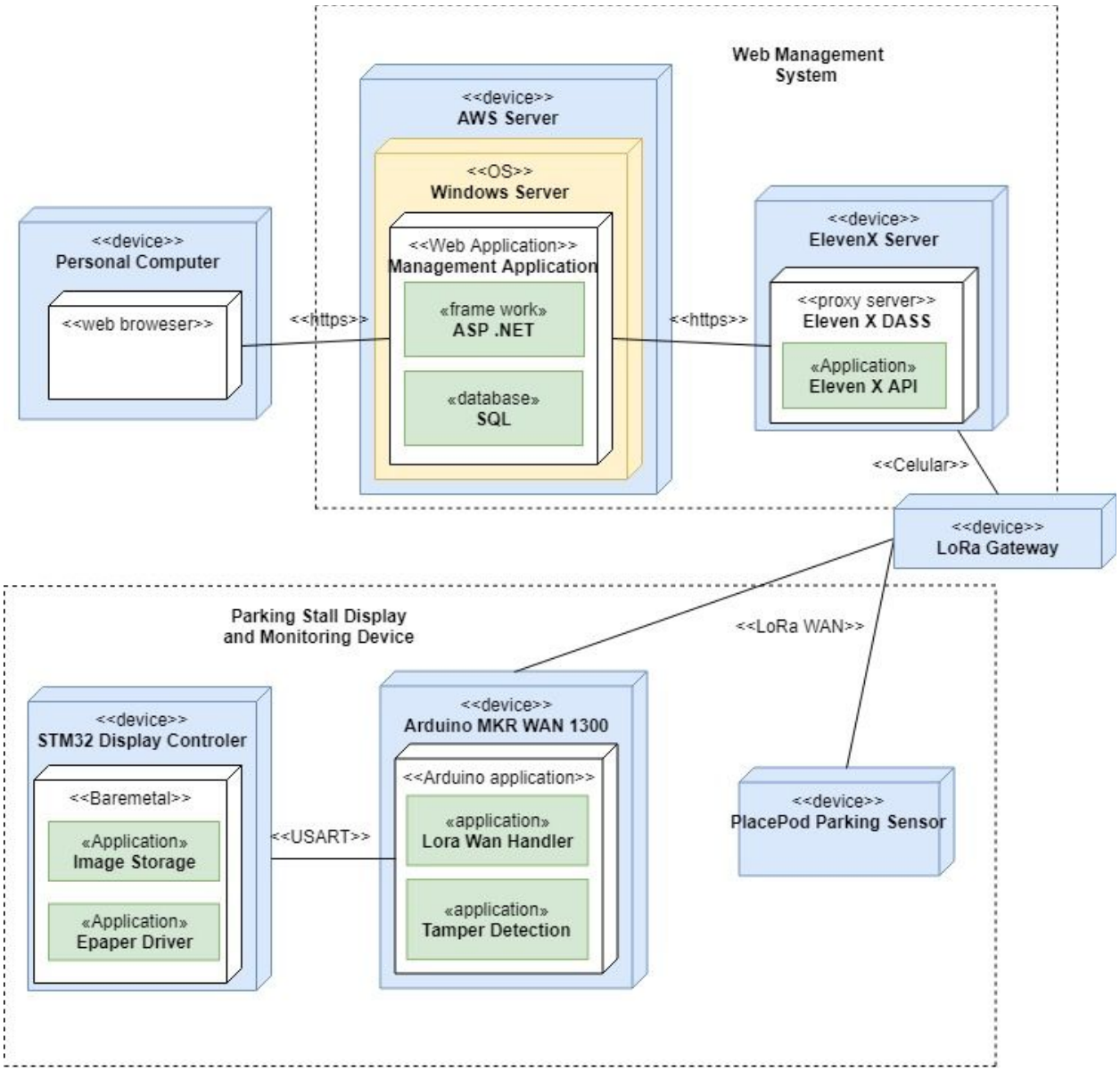

Figure 2.1 - System Level UML

Referencing the System Level UML (figure 2.1), the web management system is broken down into 2 main subsystems, i.e. the Management Application (section 2.1) and the Eleven-X DASS system (section 2.2). The Parking Stall Display and Monitoring device is broken down into 3 main subsystems i.e. Arduino MKR WAN 1300, the PlacePod Smart

Parking Sensor and the STM32 Display Controller. The rationale and function of these subsystems are described in detail in the following sections:

2.1 Management Application

The management application is the interface the user will use to manage deployed dynamic parking stalls. The features and functionality of this subsystem are summarized well by the Web Management System section above (section 1.2). The engine for our web application will be the ASP.NET framework hosted on Amazon Web Services. The client gave full flexibility for selecting the web-app development stack. The rationale for these components are the following:

ASP.NET:

- Active code base, supported by Microsoft
- Readily available documentation
- Mature framework, (stable)
- Powered by C#, (the entire project team has some background in C programing)

Amazon Web Services:

- Very popular, very flexible (offers instances for all popular OS's and frameworks)
- Readily available documentation
- Trendy, good for resumes

The management application will allow for the user to remotely manage the deployed Parking Stall Display and monitoring the devices, required by FR1-5. Further details of the web management design are outline in section 5.

2.2 Eleven-X DASS system

The Eleven-X DASS system acts as a middleware proxy server between the remote LoRaWAN devices and our Management Application. LoRa based devices can be registered with the DASS. LoRa payloads transmitted by registered devices will be received and cached on the DASS proxy server. Applications can then query the DASS to pull payloads over the web or set up DASS to push payloads over the web to the

application. Likewise, payloads can be sent downstream to the remote devices. An application can push payloads to the server where they are cached and can then be fetched or pushed to the desired remote device. This creates a black box interface between our management application and remote device's. As mentioned in section 1.3 the Eleven-X DASS system is currently used and deployed by UBC parking. Several LoRa gateways are set up around campus connected to the Eleven-X DASS over cellular. Using the DASS is a clear choice given the pre-existing infrastructure and use by UBC parking. This subsystem is needed for remote communication with our devices, which is required by FR1.

2.3 PlacePod Smart Parking Sensor

The PlacePod Smart Parking Sensor is placed underneath a dynamic parking stall. It utilizes a geomagnetic sensor to detect if a car is parked above. This device does not directly connect to our Parking Display as it utilizes LoRaWAN as its form of networking. Detections are relayed from the PlacePod sensor to the Web App. UBC parking currently has PlacePod sensors deployed for car detection, so we have chosen to integrate these detectors into our system. This car detection functionality is required by FR3.

2.4 Arduino MKR WAN 1300: LoRa Handler, Temperature, and Accelerometer controller

One of the biggest challenges to designing our system is power management, as the Parking Display is constrained to being powered autonomously. To design this system we are using the Arduino MKR WAN 1300 which is a very low powered microcontroller (<20 mA operating current) that has a built in LoRa module. However, the device is limited by its computing power. The MKR WAN 1300 will always be running in the system; its tasks include:

- Receiving and transmitting LoRa packets
- Detecting any tampering of the device through an accelerometer

2.5 DESPI-M02 Display Controller

The DESPI-M02 display controller is used to store and update images on our e-paper display (e-paper will be used to display the parking restrictions; see section 3.3.5 for further details on the e-paper display). The board features an STM32 microcontroller which is used to drive the 2 quad SPI interfaces of the e-paper display. Additionally the board features 128MB external SPI flash chip which was utilized to store images on the device, (up to 100 images). Images are preloaded onto the board over USB serial using an image loading script we developed. We went with the DESPI-M02 as it was the recommended development board by the e-paper manufacturer, (GooDisplay).

3. Parking Stall Display and Monitoring Device

This section goes into further details of the Parking Stall Display design outlining: design requirements/rationale (3.1), system overview (3.2), component list/rationale (3.3)

3.1 Design Requirements/Rationale

- Low powered: able to operate power autonomously off of a 20W solar panel
    - Solar panels larger than 20W are dimensionally large for a single stall
- Average power consumption of the device cannot be more than 0.8W
    - In shortest daylight conditions in Vancouver we can expect only 1-2 hours usable sunlight, i.e. direct sunlight. [10]
- Ability to update and display parking restrictions autonomously
    - FR1,2
- Ability to detect parked cars at stall
    - FR,3
- Ability to detect vehicle impacts with the sign
    - FR4
- Enclosure must be rated IP65 to protect against rain, dust, wind and snow.
    - NFR1

3.2 Overview

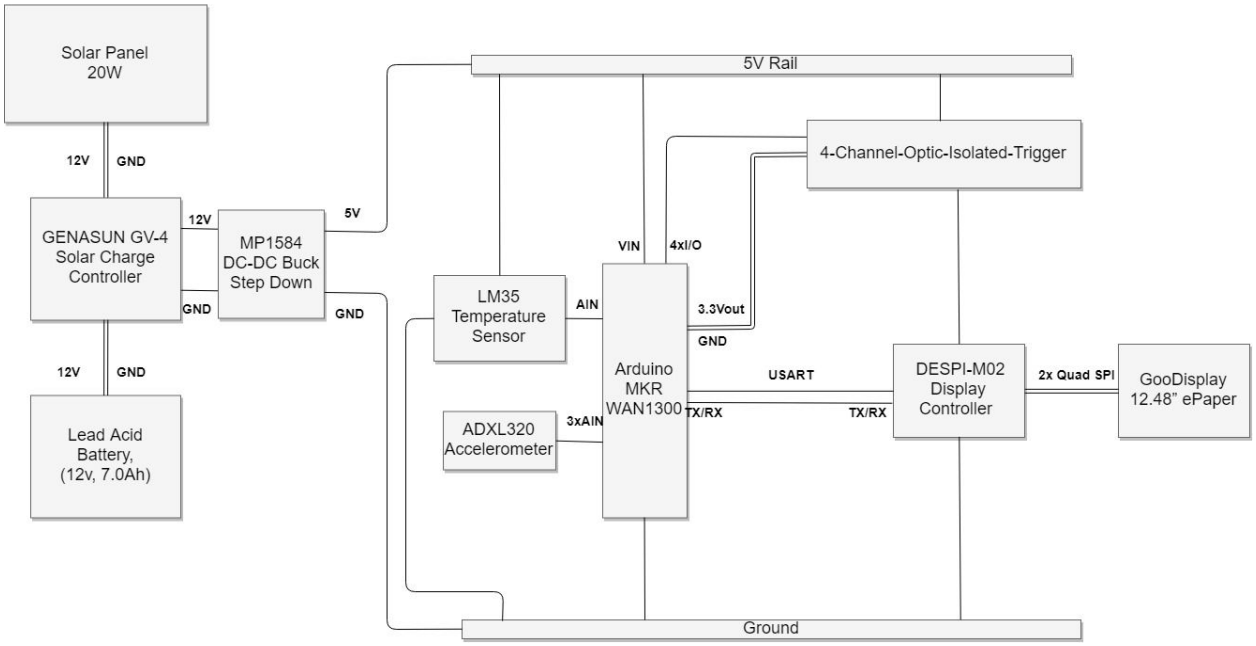

Figure 3.1 - Block Diagram System Overview

Above is the device topology diagram in figure 3.1. The system is powered by a lead-acid battery which is charged by the Solar Panel/Solar Charge Controller. The battery output is regulated by a DC-DC buck converter to step down to 5 V for the system. The Arduino MRK1300 is always be running, its functionality is outlined in section 2.4. Note the Arduino controls the power to the DESPI-M02 display controller which will only run to update the display, (e-Paper does not need power to hold a still image).

3.3 Components List

The following section provides details and justification of the parts that were chosen for the device. The quantity of each part per device is noted in the headings.

3.3.1 Arduino MKR WAN 1300[1] (x1)

Functionality: handles LoRaWAN and device side management.

- A major design choice was LTE vs LoRaWAN, this decision making was highlighted in section 1.3 LoRa. The Arduino MKR WAN 1300 features a built in LoRaWAN module to ease prototyping.
- The arduino MKR series microcontrollers are extremely low powered as it idles at 20mA (at 3.3V), thus was chosen to help meet the projects low power constraint.
- Arduino devices simplify and accelerate microcontroller prototyping with it's extensive, easy to use and well supported libraries. Thus we choice to use Arduino given the 8 month timeframe of the project.

3.3.2 GooDisplay 12.48" Color Large E-Paper Screen[8] (x1) / GooDisplay Communication Board Demo Kit Adapter for 12.48" E-paper Display[9] (x1)

Functionality: Display parking restrictions at stall.

- Implement the project's functional requirement to display parking restrictions.
- Meet the project's low power constraint; this e-epaper display consumes around 0.045W of power to change the display and requires no power to sustain an image. Thus due to the static display nature of our project this display was chosen over other display options such as LED matrix displays and LCD displays which require constant power to display an image.

3.3.3 DESPI-M02

Functionality: Drive the e-paper display and store images

- We went with the DESPI-M02 as it was the recommended development board by GooDisplay.

3.3.4 LM35 Temperature Sensor (x1)

Functionality: Monitor temperature of the device.

The LM35 Temperature sensor is used to monitor the temperature of the device enclosure. This will work in conjunction with the Arduino to ensure that when the temperature is below 0 degrees an image will be uploaded to the display twice to prevent image ghosting (remains of the previous image).

3.3.5 ADLX345 Accelerometer (x1)

Functionality: Detect any tampering with the device.

The ADLX345 Accelerometer sensor will be used with the Arduino MKR WAN 1300 to continuously monitor the device enclosure movement. In the event that a car hits the pole on which the device is mounted or someone is tampering with the device, notifications will be sent to the administrator. This device consumes 58μW when in measurement mode and needs to be powered all the time so that any tampering with the device can be constantly monitored.

3.3.6 Four Channel-Optical-Isolated-Trigger (x1)

Functionality: Interface to control power of the DESPI-M20.

The Four Channel-Optical-Isolated-Trigger will be used by the Arduino to control the input power to the DESPI-M20 display controller. These are the components that consume the most power in the system and will only be used when needed. Note this particular relay switch was selected because its I/O on range accepts 3.3V as ‘ON’ which is the output voltage of the I/O pin in On state of the Arduino MKR 1300.

3.3.7 Lead-Acid Battery (x1)

Functionality: Power the system.

We decided to go with lead-acid batteries as our client has an abundant supply of these batteries. We are using a 12V, 7.0Ah battery which should be more than enough to power our system 24/7, (see our power budget in section 4)

3.3.8 MP1584 DC-DC Buck Step Down Converter

Functionality: Step down the battery output voltage to 5V.

We needed to step down our battery output voltage from 12V down to 5V. Buck converters are the most efficient way to step down a DC voltage.

3.3.9 20W Solar Panel/GENASUN GV-4 Solar Panel Control

Functionality: Recharge the devices battery

This solar setup was provided by our client. A 20W panel is more than enough to keep our battery charged, and hence keep our device running continuously (see our power budget in section 4).

3.4 Housing Design

The parking stall display is to be housed in an adapted off-the-shelf casing to help meet our unique size and weatherproofing requirements. The materials chosen for the housing are outlined below:

- PVC casing
    - PVC casing (JB12126 PVC box) rated to IP66 was chosen as it is an off the shelf solution that has already been validated for weatherproofing.
- Acrylic display cover

- Acrylic was chosen for the display cover as it is low glare and UV resistant making it perfect for outdoor display applications.

All of the casing materials were chosen to meet the project requirement of being weatherproof up to IP65. A rating of IP65 indicates that the casing must be completely protected against dust and that it must be resistant to a water jet with a pressure of 30 kPa, at a distance of 3m, for 15 minutes.

The housing consists of an aluminum faceplate with an acrylic screen cover set into the centre to protect the e-Paper display. A PVC box sits on the back of the faceplate, connected to the acrylic screen cover, to house all of the electrical components for the device.

4 Power

4.1 Power Budget

Due to the project goal which states the signs must be stand alone units and C4 which states the system must be self-powered, the device will be solar powered and therefore cannot consume more than 20W per day.

4.2 Measured Power Consumption

Scenario	Time (hr)	Current (A)	Power(W) 5 V system
IDLE *	23.8	0.04	0.20
Update Screen	0.2 **	0.20	1
Average Per Hour	1	0.06	0.3

Daily power consumption = (23.8 x 0.2) + (0.2 x 1) = 4.96 Wh

Assumptions

* LoRaWAN module transmitting and receiving consumption are negligible compared to IDLE therefore are not considered on their own

** The update screen time was calculated as follows: screen updates take 30 seconds, and assume screen updates once an hour (this is a very large assumption), thus

hours = (30 seconds / 3600 seconds/hr) * 24 hr = 0.2 hr

4.3 Power Analysis

Given our worst case scenario of only 2 hours of usable sunlight in vancouver [5] and 15% solar panel efficiency [6] our panel should charge our battery:

Charged Watts/per day = 2 Hours * 20W * 0.15 = 6Wh

This value exceeds our absolute worst case scenario power consumption of 4.96 Wh.

Additionally, the battery that we are using is rated at 7Ah (84Wh, 50.4Wh effective considering worst case discharge) fully charged, thus based off our calculated worst case scenario average power consumption our system can run for ((84Wh*0.6)/0.3W) 168 hours straight without a charge. This is equivalent to 7 days.

5 Web Application

This section details the requirements and tools used to develop the web application that UBC parking will use to control and monitor the displays.

5.1 Design Requirements

The following are design requirements for the web application as identified by the team:

1. Device list views.
2. Stall management
    a. Add/Delete
    b. Set display and restrictions
3. Scale to handle unlimited number of stalls.
4. Hosted on AWS
5. Transferable to UBC server after the project is finished

An overview of the web application system can be seen in figure 5.1.

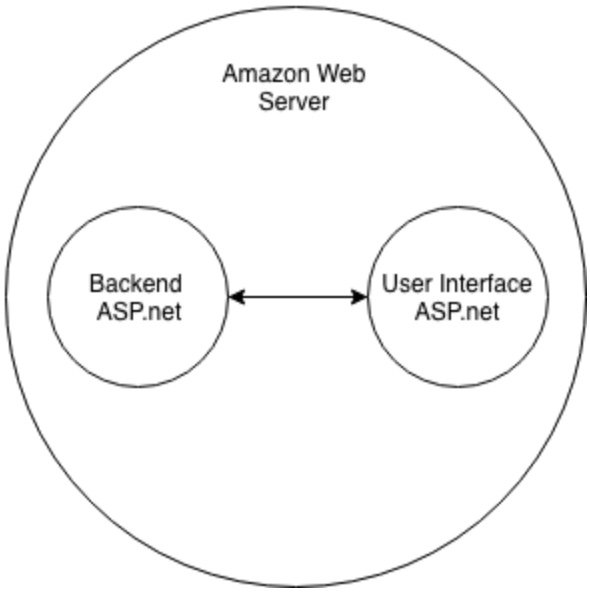

Figure 5.1 - Components of Web Application

5.2 Web Application Components

Amazon Web Services was chosen to host our web application. This decision was made because it:

- Satisfies web application design requirement 4 that the web application must be hosted on AWS.
- Implements design requirement 5 to be transferable to UBC server after the project is finished

UBC can set up a server to host the web application, but due to administrative oversight and approval timelines this option was negated as delays would have disrupted the project's timeline.

ASP.NET was the web framework chosen to implement the application because:

- Implements design requirements 1-6.
- Documentation readily available.
- Smaller learning curve, which allows the team to quickly learn and use the framework.

An alternative that we looked at was using React for the frontend and Node.js for the backend. This was decided against this as there were concerns with the learning curve of React.

5.3 Wireframes for Web Application

Code for the web application can be found on github at:
https://github.com/acsikos/Dynamic-Parking-Web-Application.git

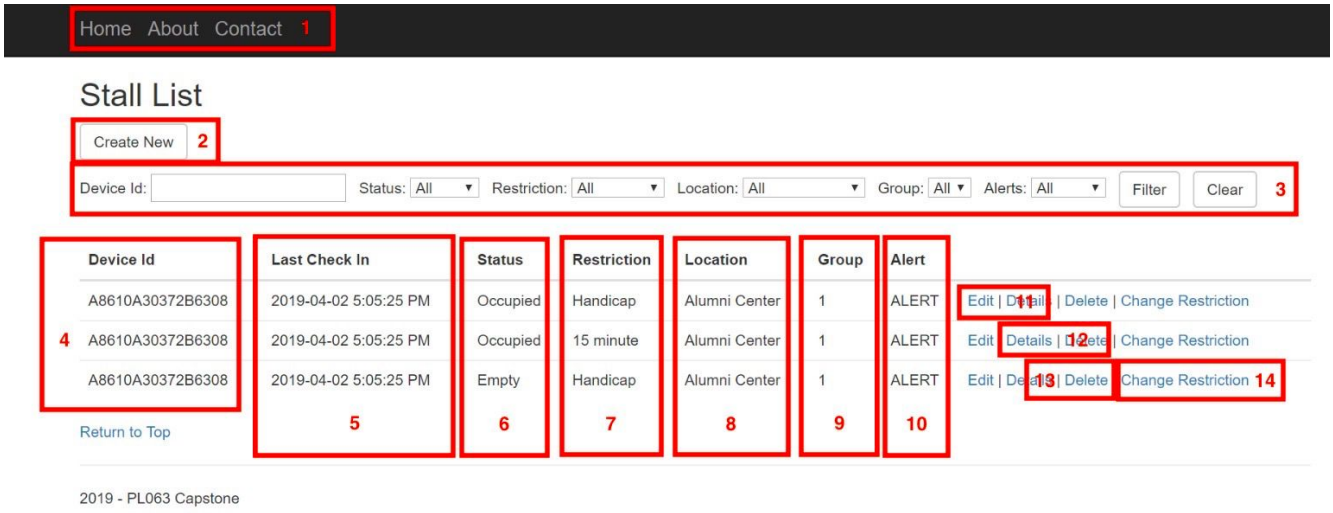

Figure 5.1 - Page 1: Stall List

1. Header bar provides links to the other pages
2. Create New button allows you to add a new stall
3. Filter search options. Options to filter search such as “occupied/vacant”, “Alert”, “Restrictions”, etc.
4. Device ID column gives the device’s ID used to identify the device on elevenX
5. Last Check-in column shows the last time the web application received a check-in from the stall
6. Status column shows the stall’s status of the stall occupied/Empty
7. Displayed Restriction column shows the current restriction the stall is displaying
8. Location column shows where the stall is located
9. Group column that displays the stall’s group
10. Alert column indicates whether or not the stall needs attention
11. Edit button to make changes to the stall
12. Details button opens more details about the stall
13. Delete button removes the stall
14. Change restriction takes the user to the Change Restriction Page

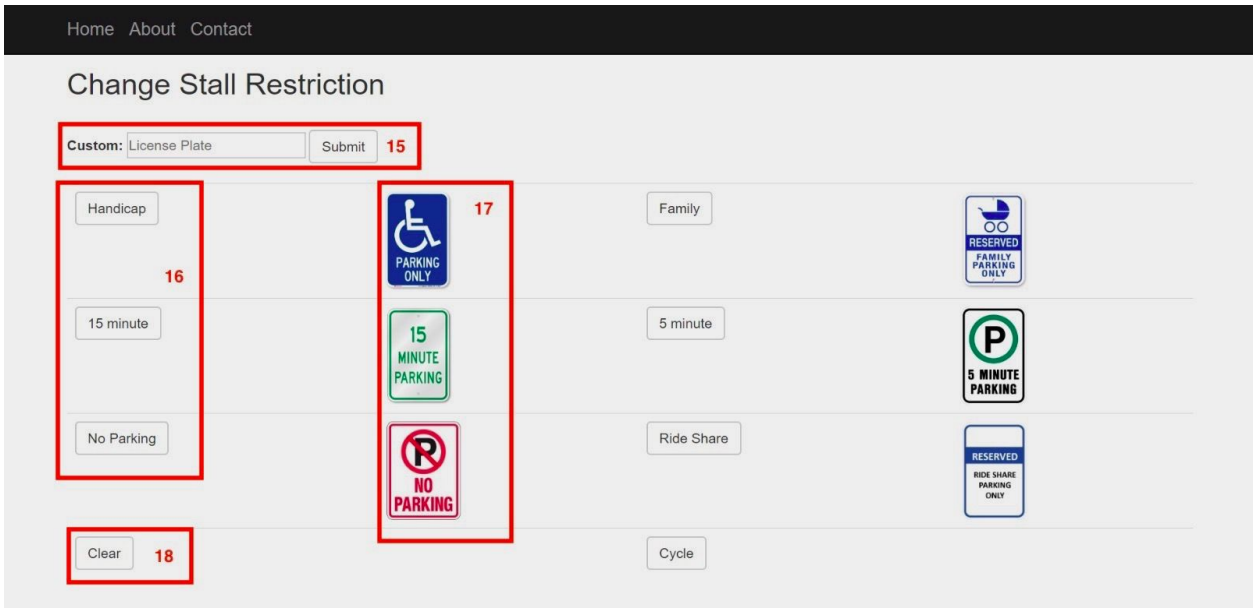

Figure 5.2 - Page 2: Restriction List

15. Add a customized restriction to reserve a stall for a given licence plate
16. Selection buttons that submit the restriction change to the stall
17. Image of the displayed restriction
18. Clear the display

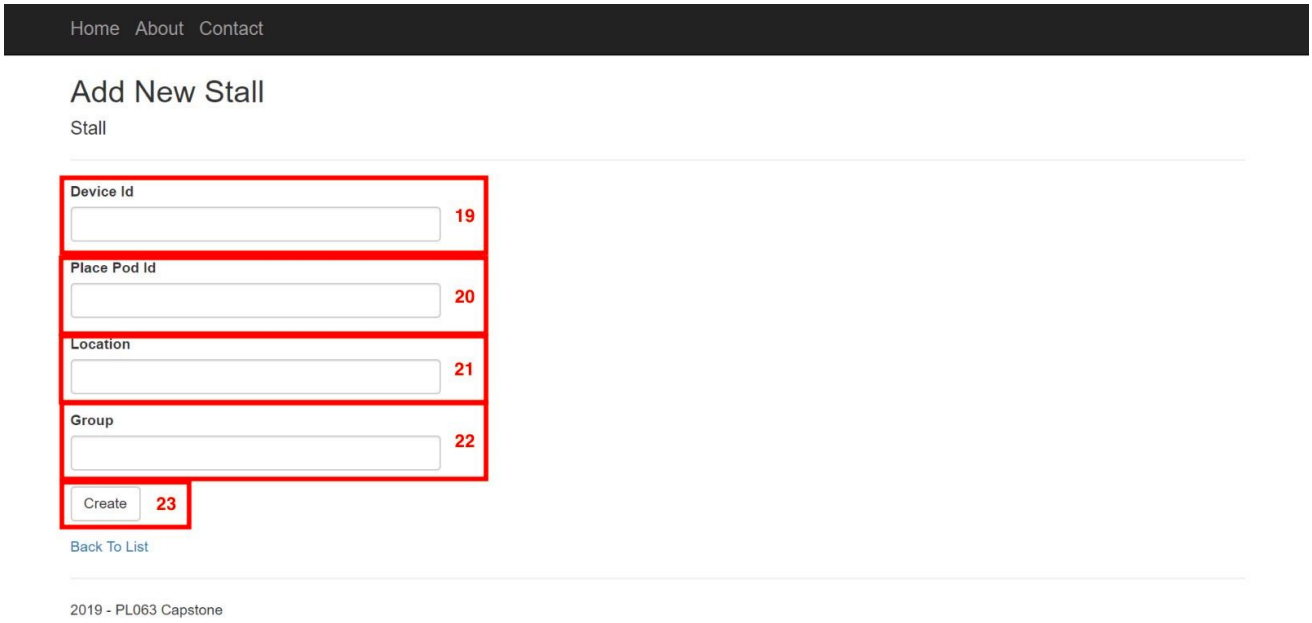

Figure 5.3 - Page 3: Add New Stall

19. Device ID which represents the ID of the arduino of the stall
20. PlacePod ID which represents the ID of the PlacePod of the stall

21. Location where the stall is located
22. Group that the stall is in
23. Create the stall

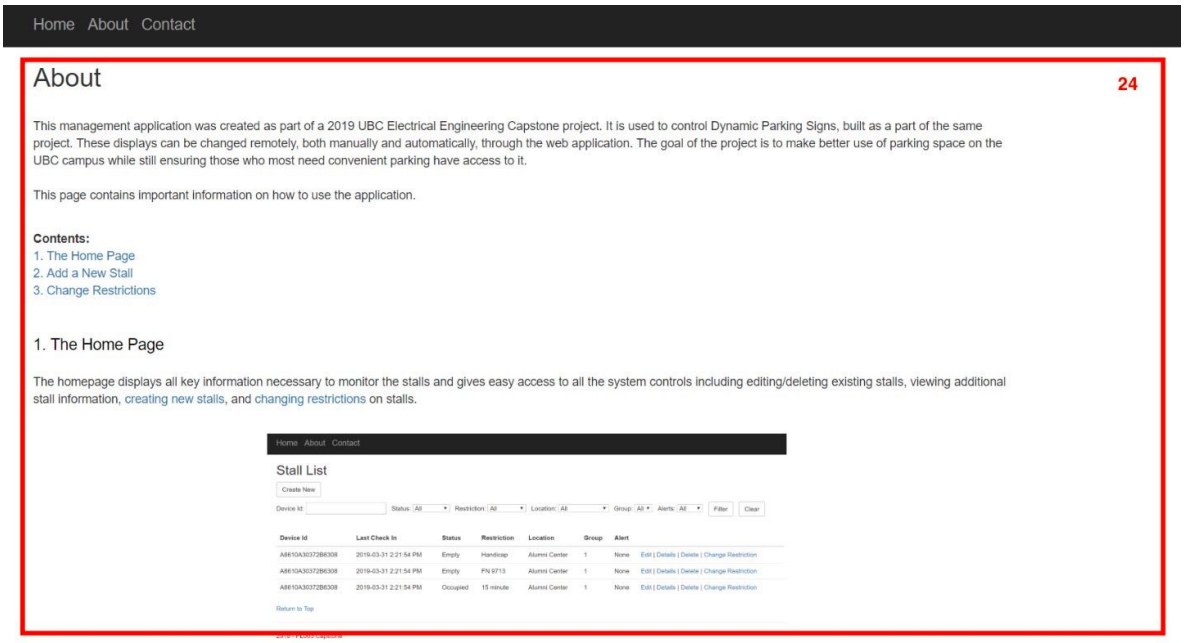

Figure 5.4 - Page 4: About

4. Displays a user manual of the web application

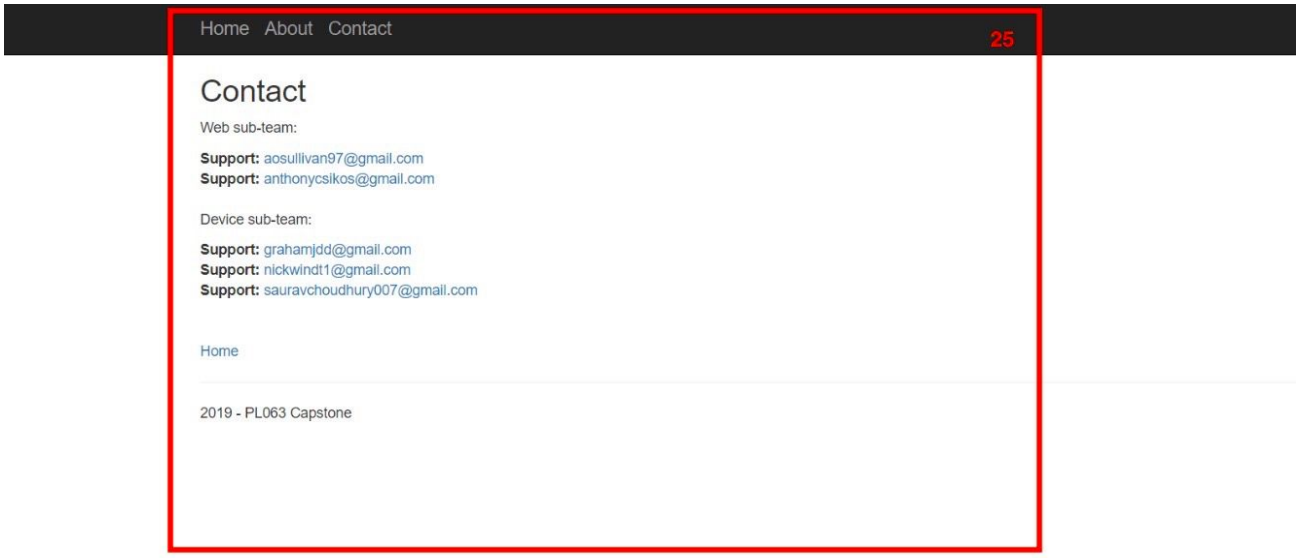

Figure 5.5 - Page 5: Contact

5. Displays contact information of the web developers

6 Software System

This section provides high level flow diagrams of software functionality needed to achieve functional requirements of the system.

6.1 Downstream Payload Flow

Figure 6.1: Downstream Payload Flow

Figure 6.1 shows the downstream flow of data from the web app to the e-paper display. First, from the web app, a packet with a desired command is stored in a JSON object and pushed to the Eleven-X DASS through a HTTPS request. The end device is specified by a DEVEUI which is unique to each deployed device. This DEVEUI is included in the path of the HTTPS request. Next, the packet is forwarded from the DASS to a LoRa gateway where the packet is then transmitted over LoRaWAN. The LoRa packet is received by the Arduino MKR with the specified DEVEUI and decoded by the LoRa handler, (for this example the request is to display a desired image which is currently stored on the DESPI M02). The Arduino powers the DESPI M02 and transmits the image number to be displayed over a USART connection between the 2 devices. The DESPI M02 then reads a single row (163 bytes) of the desired image at a time and writes it to one of the 2 quad SPI interface (each quad spi controls a half of the display). Upon writing the entire image to the display driver, an update command is passed to the display which then updates it to the new image.

6.2 Arduino Initiation Algorithm

```
//Start Device
Display(CONNECTING)          // DIsplay Connecting on e-paper display
Modem.start()                // Start LoRa Modem

While(Modem fails to connect)   // Check if LoRa modem connected to Eleven-X
        wait(30s)               //If modem failed to connect wait restart and try again
        Modem.Restart()

if(Modem connects)              //If successfully connected
        Display (HANDICAP)      // Default Parking Sign
```

6.3 Arduino Flow Cycle

Figure 6.3 shows the Flow Cycle of the Arduino, which is going to be our hub for all the communication and sensor data processing.

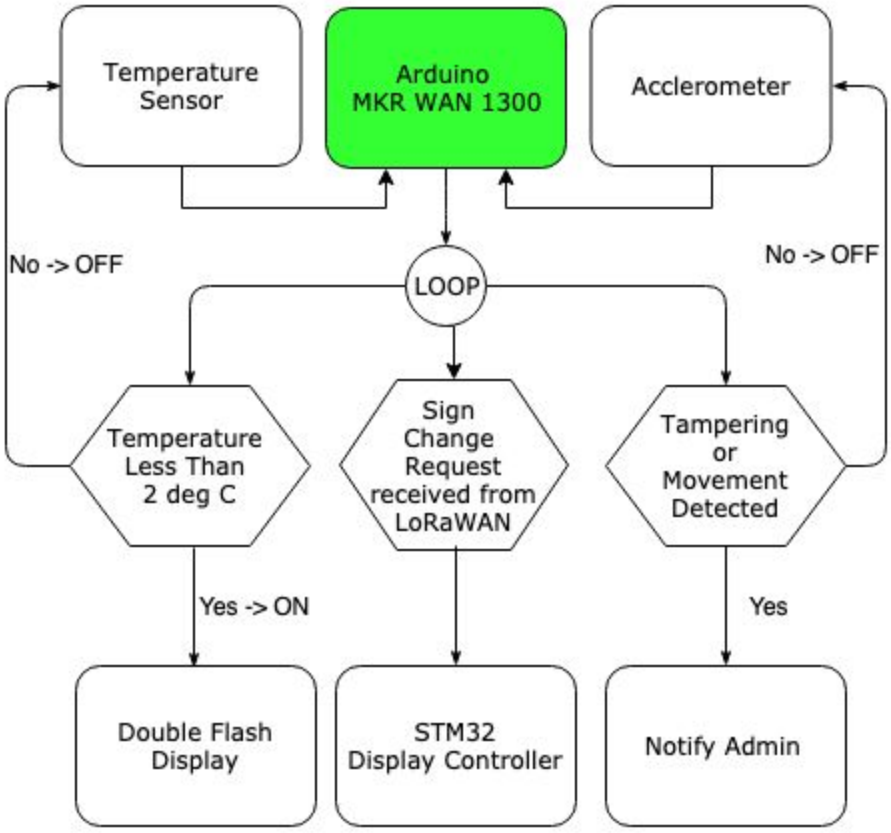

Figure 6.3 Flow Cycle of the Arduino

References

[1]"Arduino MKR WAN 1300 (LoRa connectivity)", *Store.arduino.cc*, 2018. [Online]. Available: https://store.arduino.cc/usa/mkr-wan-1300. [Accessed: 25- Nov- 2018]

[2]"---e-paper display - GOOD DISPLAY", *E-paper-display.com*, 2018. [Online]. Available: http://www.e-paper-display.com/download_detail/downloadsId=693.html. [Accessed: 25- Nov- 2018]

[3]*Amazon.ca*, 2018. [Online]. Available: https://www.amazon.ca/GooDisplay-Paper-Screen-Display-Electronic/dp/B07JG1TN5B/ref=sr_1_2?s=industrial&ie=UTF8&qid=1543125424&sr=1-2&keywords=GooDisplay. [Accessed: 25- Nov- 2018]

[4]*Amazon.ca*, 2018. [Online]. Available: https://www.amazon.ca/GooDisplay-Paper-Screen-Display-Electronic/dp/B07JG1TN5B/ref=sr_1_2?s=industrial&ie=UTF8&qid=1543125424&sr=1-2&keywords=GooDisplay. [Accessed: 25- Nov- 2018]

[5] Current Results, 2018 [Online]. Available: https://www.currentresults.com/Weather/Canada/British-Columbia/sunshine-average-december.php
[Accessed: 10- Feb - 2019]

[6] S. Murmson, “The Average Photovoltaic System Efficiency,” Sciencing, 02-Mar-2019. [Online]. Available: https://sciencing.com/average-photovoltaic-system-efficiency-7092.html. [Accessed: 04-Apr-2019].