THE GARAGINATOR

Group 5 – Summer/Fall 2017

Jonathan Gillis	Computer Engineering
Elliot Rodriguez	Computer Engineering
Sebastian Rodriguez	Electrical Engineering
Jon Staudt	Electrical Engineering

Contents

1 Executive Summary

A common problem among people using parking garages is finding a spot to park when arriving. At the University of Central Florida, parking is such a problem that it is not uncommon for students to be absent or late for quizzes and tests, simply because they could not park their car. This often happens despite arriving on campus with an inordinate amount of time before hand. This is despite students paying over $50 (at minimum) for a parking pass for 1 semester. Since Public transportation in the city of Orlando is severely lacking, most students have no other option than to drive to class every day.

Numerous attempts have been made in the past to solve this problem, but have been too costly to implement at scale. As such, the Garaginator system was designed first and foremost with costs in mind. The Garaginator consists of about 3 main parts: sensors located at the garages entrances and exits, a kiosk, and a mobile application. The sensors are connected to the Kiosk which is able to interpret the data they send. In addition to interpreting signals from the sensors, the kiosk can interface with the webserver, posting information about capacity and recording vehicle locations. The Kiosk has a keypad and a display so that users can input which parking spot their car is in. A mobile application also interfaces with the server, and allows users to query the last recorded location of their vehicle. Lastly, a QR code printed and added to every parking spot can be scanned by the mobile application, allowing them to enter their vehicle's parking information more easily.

The Kiosk will be powered from the power mains, via a wall outlet, and will communicate with the webserver using a standard internet connection. The sensors will be optical sensors which consist of photodiodes and a laser. The laser will be normally activating the photodiode. When a vehicle passes by the laser and blocks it from the photodiode, the photodiode will stop conducting and the pin it is connected to will go low, thus indicating that a vehicle has just entered or exited the garage. The sensors will be hardwired, directly, to the kiosk, to save on the cost of wireless transceivers and microcontrollers.

The webserver acts as a main "back end" component that the user will never interface with. It will handle all requests from both the mobile application and the Kiosk application. It will handle updating the database when the sensors send signals to the microcontrollers, as well as send the location of a user's vehicle to the correct microcontroller when requested. The webserver will also authorize use of the system when a user first tries to login, using an AES encryption technique. Lastly the webserver will also store a user's vehicles location, and send a simple completed message to the user once complete.

2 Project Description

The following section will serve as a brief summary of those things that inspired the group to build the Garaginator. This section also contains the essentials of any project, such as task allocations, estimated budget, how the project will work, and last, but most certainly not least, time and design constraints.

2.1 Motivation

A common problem for commuters of the era of the personal car is finding a place to park. This problem is evident among students at UCF, where many students believe there are far fewer parking spots than there are people who need to park their car during peak hours. Thus, many students may spend an exorbitant amount of time searching for a parking spot, or park illegally to avoid missing a quiz or a test. Often the only reason the student cannot find a spot is because they are looking in a crowded garage/lot, when there may be a less crowded one nearby. This project aims to develop a system that could be installed in parking facilities to provide drivers with enough information to park soon after arriving on campus.

UCF attempts to approach this problem with signs outside of each of the nine garages which display to drivers whether the garage is full. Although it is helpful, such a system does not attempt to aid the driver in finding an available spot, only in avoiding garages without them. This results in students driving around much of campus looking for an empty garage, which is not guaranteed. Furthermore, UCF has many parking lots to supplement these garages, however the parking lots have no such warning, forcing students to manually check each lot if they want to park there. This process is wildly inefficient and frustrating for the student.

Another related problem is the enforcement of parking decals in parking facilities. As it stands it is a very manual process, requiring officers to individually check every parking spot on campus, and every parking decal on those cars. This is a very labor intensive task since the decals can be in many different places on the vehicle, or inside the vehicle. Thus, it is a very slow, and error prone process.

Lastly, at the end of a long day on campus, it is common for some students to not remember where they parked their car. Since there are nine garages on campus and even more parking lots, tracking down one’s car can take an embarrassing amount of time.

Our projects seek to solve these problems. Our system will be able to track the location of registered and tagged vehicles inside of a parking garage, and determine which spot they are parked in. This will allow drivers to view on their smartphone exactly which spots in each garage are available, from anywhere. By detecting spots which are occupied, but not by a vehicle carrying our decal, we can determine which spots are occupied by non-registered vehicles, and are therefore parked illegally. Lastly, by searching for the location of a specific vehicle, users will be able to find their car after a long day.

2.2 Goals and Objectives

- The system will incur a very low cost on installation.
- The system will be able to count the vehicles entering and exiting the garage with minimal error. Cars should generally be counted entering and exiting as close as possible to exactly once each.
- Errors in the capacity measurement should tend to be corrected over time. Since errors in the detection of entering and exiting vehicles is probably inevitable, errors in the counter should be corrected over time by some other means, such as data from a parking enforcement officer.
- The system should be quick and easy to use. Students often wait until the very last minute to drive to campus, especially for ones early in the morning. If it is not quick and convenient many students will not want to use it.

2.3 Project Specifications and Constraints

In this section, we discuss our project specification we initially made for our project idea. These specifications get discussed later on in the document, where we show how we plan to implement and test each specification.

2.3.1 Garage Sensors/Communication Specifications

- The system should be able to determine if a vehicle and only a vehicle entering the garage.
- If the vehicle has been registered, then its location should be recorded and made available to its owner.
- Information regarding whether a parking spot is taken/untaken should be sent out to a web server for processing and update of the mobile application.
- Users will be able to enter in their vehicle information and their parking spot for it to be remembered.
- Users information will be entered automatically, if they own the smart decal.
- The Garage

2.3.2 Mobile Application/Web server Specifications

- The application should be user friendly.
- The application should be able to map out every parking spot in a garage.
- The application should be able to communicate with the embedded components located in the garages via web server.
- The application should contain a secure payment method for one day parking pass.
- The application should be able to tell you the closest parking garages near a building the user selected on campus.

- The application should contain the exact number of cars currently located in a garage as well as how many total spots are in that same garage.
- The Web Server should be able to communicate with a DBMS such as MySQL, Cassandra, SQL Server, or Oracle.
- The Web Server should be able to communicate with the mobile application.

2.3.3 Nice to have Specifications

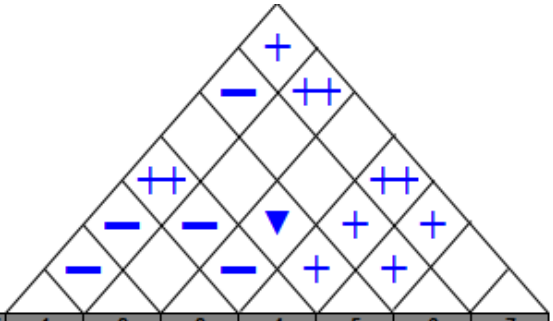
- Analytical data on what garages are full during certain times of the day/week/month.
- Warn users about days where less parking is typically available.
- Add sensors to detect cars that are parked over the line, and alert them using a light/noise.
- If a vehicle has not been registered, then the database should be updated with the number of unregistered vehicles in a garage.

2.3.4 Project Constraints

- Need to keep cost per spot as low as possible.
- Need to keep update time of taken/untaken spots as quick as possible.
- Time constraint of about 5 months to get the entire project working.
- We currently do not have any sponsors, so we must self-fund all the project's costs.
- Possible Sponsors could be UCF, as well as other locations which use many parking garages.

2.4 House of Quality

Table 1 - House of Quality



Row #	Demanded Quality (a.k.a. "Customer Requirements" or "Whats") / Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")	1 Parts Cost for Kiosk	2 Parts Cost for Sensors	3 Parts Cost per Parking Spot	4 Time required to Input Information at Kiosk	5 Time required to scan parking information on mobile app, until	6 Minimum credentials required to access user Parking info	7 Mean time between erroneous sensor readings
	Direction of Improvement	▼	▼	▼	▼	▼	▲	▼
1	Inexpensive	⊙	O	⊙	O			▲
2	Easy to use	O		▲	O	▲	⊙	
3	Accessible	⊙		▲	⊙	O		
4	Convenient	O		▲	⊙	⊙	O	
5	Accurate	▲	⊙	▲				⊙
6	Secure	▲						
	Target or Limit Value	$$	45 s	25s	< 30s	< 5s	NID & NID Password	+/- 3 cars/day
	Difficulty (0=Easy, 10=Hard)	4	4	3	7	9	8	7

Legend		
⊝	Strong Relationship	9
○	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

2.5 Block Diagram

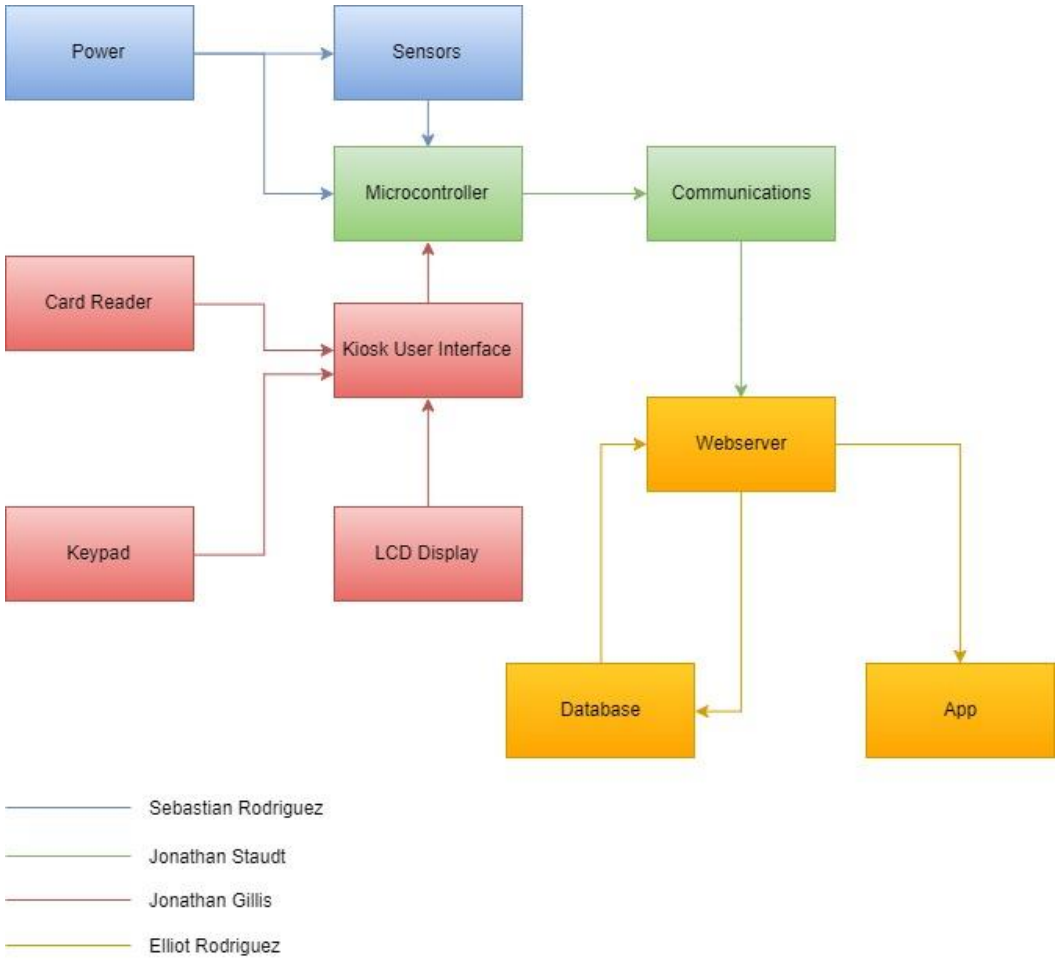


Figure 1 – Task Allocation

2.6 Use Case Diagram Mobile Application

Below in figure 2 is our initial use case diagram for our mobile application. We initially wanted users to open our application and be able to automatically use it without any authentication. This led to some security and ethical issues that we discuss later in our report. Then the user would be able to send requests to check garage availability as well as pay for a day parking pass, and finally be able to store and retrieve their vehicles location.

All of these requests would be handled by a backend webserver that will be able to communicate to a local database hosted on the same server, which would hold information such as the user's vehicle location.

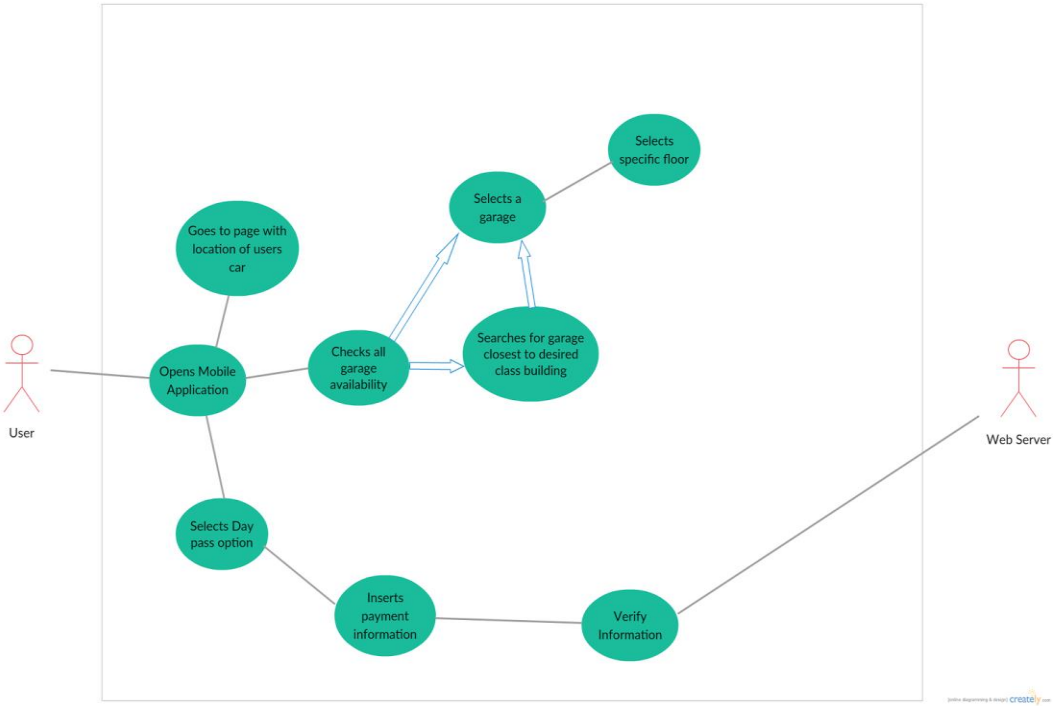


Figure 2 – Initial Use Case Diagram

2.7 Project Operation

The Garaginator is a user self-service system, where the user can store and retrieve their vehicles location. The user has access to all garages hosted on the system, which will be all garages located at UCF. Photodiode sensors are located at the entrance and exit of each garage in a custom designed housing system. Each of these diodes are connected to a microcontroller (Kiosk). Where they will send updates when vehicles enter and exit the garage. Each sensor will send out a voltage signal when a car breaks the beam between the diode and the laser. Then the microcontroller which is connected to the internet via ethernet will send an update request to the webserver.

The webserver acts solely as a back end system that handles requests coming from both the kiosk and the mobile system. It will be used to update the database as well as send information from the database to the kiosk and mobile systems.

The kiosk system will be mounted at the bottom of one corner of the garages stairs, ideally there would be two in each garage, but we are creating one to show as a prototype/proof of concept. It will be stored in another custom-built housing system. This kiosk will allow users to store and retrieve their vehicles location by manually entering a username and passcode or by swiping their UCF id card and entering their passcode. Each kiosk will need to be manually coded, to give each of them a unique id when communicated over to the webserver.

Lastly is the mobile system, which will be deployed on the IOS App Shop and the Android Store. This system acts exactly like the Kiosk system but on a mobile device, giving the user access to store and retrieve their vehicles location whenever they please. They will also be able to store their vehicles location by scanning a QR code located at the garage spot that their vehicle is currently located at.

3 Research

The only way we can be absolutely sure that we have designed the system which best meets the customer requirements, we must thoroughly research every component, to make sure that we have considered every possible option. We research not only individual components but the technologies behind them, to guide us on a path towards the optimal solution.

3.1 Existing Solutions

"*If I have seen further than others, it is because I have stood on the shoulders of giants." - Sir Isaac Newton*

In the design of anything new, it is vital to look backwards at the successes and failures of others to ensure the product will be viable. To ensure the product will be demanded on the market one must find a need that is left unsatisfied by existing options in the market.

3.1.1 Siri/ Google Now

Certainly, the most widely available solution to this same problem is that by Google and Apple. Both Siri and Google Now are supposedly capable of detecting exactly when you turn off or exit your car, and recording the location on internal memory. These solutions have the advantage that they are proactive.

These services then allow you to locate your vehicle through maps, in the case of iOS and through the Google Search on android phones. Siri and Google Now both remember the location of your vehicle without you having to do anything. However, these mostly rely on noticing when your phone disconnects from your car's Bluetooth media player, meaning that if your car does not have a Bluetooth media player, it will not work for you. Additionally, these services utilize mostly GPS and Wi-Fi networks to determine your car's location, which is, in ideal conditions only accurate to a radius of several meters, making these services more useful for finding out which parking lot your car is parked in, not finding its exact location.

This deficiency is highlighted in parking garages due to reduced signal strengths as opposed to parking lots. Lastly, these services to not seem to store any information about which floor you are parked on. As a result, when you park in a garage, these services are typically only able to help you find which particular garage you parked in, still leaving you to scour the whole garage to find your vehicle.

3.1.2 Denmark automated parking system

Denmark has implemented a completely different approach to parking garages. The goal is to completely automate parking from beginning to end. Drivers drive onto a special platform, and exit their vehicle. Next the garage itself manages

parking the car, using a robotic system the car is automatically placed in a parking spot. The system stores the location of that vehicle in a database, which is recalled when the driver wants their car back.

The goal of this approach is really to solve a problem entirely different from the one which we are tackling, although it does solve it as well. This system aims to solve the whole problem of parking, from finding a spot to doing the legwork of finding your car. Thus, the system is prohibitively expensive to be implemented on a large scale.

3.1.3 UCF Garage Capacity Displays

As it turns out, UCF does seem to recognize the problem which we are attempting to solve. At some point UCF decided to put large displays outside of each parking garage which read either “FULL” or “OPEN” depending on whether the number of cars is close to the garage’s capacity or not. It appears as though this system uses metal detectors under the pavement to detect cars in the entrance and exit to determine how many cars are in the garage.

UCF’s current implementation still has deficiencies compared to our project. First, it does not tell drivers where they *can* find a spot, only where they *cannot.* Since drivers do not have the guarantee of a spot should they look in a less desirable location, many choose to try their luck anyways.

3.2 Sensors

Contained within this section, one will find clear and concise research which has been performed in order to further the goal of choosing the proper and perfect sensor for vehicular detection. Many sensors will be considered, and the operation and how each sensor may fit in the scope of the project will be laid out in detail.

3.2.1 General Description

Sensors are devices that can detect and respond to various forms of input. The inputs can vary significantly in their form, varying from radio frequencies, light, heat, even to only detecting certain patterns. Most sensors these days provide an output of electrical signals. Sensors are heavily used around the world, so their practicality is clear. The technology of sensors is constantly maturing and developing, allowing more and more sensors to be developed and in turn, these sensors are more sophisticated than their older models and in turn allow for more sophisticated devices to be built. This allows technology to impact our lives in a wider variety of ways as well as expanding on the ways in which it already impacts our live.

Choosing the sensor which best suits a project can be a very harrowing affair. Not only are there an incredible amount of ways in what type of phenomenon can be sensed, e.g., optical, mechanical, chemical, electrical, but there are also an almost

overwhelming number of sensors which can be used for each of these. To be able to choose the right sensor for a project, there are many factors which must be taken into consideration.

The overarching factor which takes precedence within this project is cost. The cost of the sensors must be weighed against the power and features of the sensor and justified. Other important factors are range of detection, accuracy, power consumption, lifespan, and speed.

Cost is among the most important factor, because the project is self-funded and has no sponsors. Lowering the cost of the overall project will also make the project itself more appealing to any interested parties when compared with any relevant competitors. Although there will be only be one to two sensors placed at the entrances and exits of the parking garage, when this is multiplied across many garages, the cost could grow to be very great and thus a low-cost sensor should be chosen, which also has adequate capabilities and functionality in the other criteria outlined within this paper.

Range of detection must be taken into consideration due entirely to the fact that the sensor must be able to detect the relevant inputs from either the ceiling of the entrance and exit or the wall of the entrance and exit. The range of detection is more of a consideration from the ceiling of the entrance/exit as this is a further distance from the car than the wall of the entrance/exit would be from the vehicle which is entering or exiting the garage.

Power consumption ties in directly with lifespan. The more power a device uses, the less time it will be able to operate continuously. Therefore, the power consumption of the sensor should be as low as possible, while still fitting the bill of the other criteria, to allow for the longest lifespan possible.

As can be seen from the criteria above, selecting an optimal sensor will be no easy feat. The following section will explore various sensors which meet the criteria set out for the sensors and remove from the list of possible sensors, those which do not meet the criteria and thus are not viable within the scope of this project.

3.2.2 Sensor Types

The various sensors meticulously and painstakingly researched by the group for the specific purpose of vehicular detection for the Garaginator project shall be laid bare for all to see in the subsections of sensor types.

3.2.2.1 Barcode Scanner

A barcode scanner is essentially an optical scanner. The working principle of a barcode scanner is that a light source, usually a laser or LED, will shine red light upon a barcode. This light is reflected by the barcode back to the sensor in the barcode scanner. An analog signal is then generated which has a varying voltage that represents the intensity of the reflected light. A converter then converts the analog signal into a digital signal that the decoder and computer are able to

process. The decoder takes the digital signal and performs operations on it in order to validate the barcode. Once the decoder has confirmed that barcode is valid and able to decipher, it converts the signal into ASCII text, formats it, and then sends off the signal to the computer which the scanner appended to.

The working ranges of a barcode scanner depend on the method of illumination used within the barcode scanner. If LEDs are chosen as the illumination system for a barcode scanner, then the working range is highly limited, from directly in contact with the barcode to only about one inch away. If a laser is chosen as the illumination system, however, then the working ranges become much more attractive. The range of a laser barcode scanner is between one inch to eighteen inches away. However, by increasing the power of the laser, as well as decreasing the angle of oscillation of the laser, ranges of up to and exceeding twenty feet can be obtained. These ranges can be further increased with the use of special reflective long-range barcode labels. A barcode scanner can scan vehicles travelling as quickly as twenty-five miles per hour.

A barcode scanner could be useful in this project to detect when cars enter or exit the garage. Since it is trivial to make each barcode unique, the barcode scanner could also be used to detect specific user's entrances and exits of a garage. Another more practical application for barcode scanners in this project could come in the form of scanning in at the main microcontroller/keypad within the garage.

The price for a barcode scanning module is between forty to seventy dollars. This price is dependent on various factors. These factors are range of scanning, resolution, and interface compatibility.

The advantages of using a barcode scanner are:

- Barcode labels are inexpensive. Bulk of price comes from purchasing the actual barcode scanner
- Easy to integrate into project overall
- Already implemented in security gates and access control, many cases to study and learn from
- Easy to update parking decals with barcodes
- Passive. Tag does not need to be powered, only the scanner.

The disadvantages of using barcode scanner are:

- If illumination system breaks down, entire scanner is useless. Relies very heavily on one component of system
- Only works at short distances unless modified to work at longer distances by use of higher powered lasers and special reflective barcodes. This drives up cost.
- Can suffer in performance from sun damage or while raining
- Can be cumbersome to implement physically, due to range of scanning

Table 2 – Barcode Scanner Parameters

Parameter	**Value**
Operating Voltage	3.3 VDC
Operating Current	210 mA
Stand-by Voltage	3.3 VDC
Stand-by Current	7 mA
Operating Temperature	-20°C to 60°C
Humidity	5% to 95%
Light Levels	0 to 100,000 lux
Resolution	≥ 5mil
Scan Range	25mm to 380mm
Dimensions	46mm x 32.5mm x 11.5mm
Weight	80g

3.2.2.2 RFID Sensor

RFID stands for Radio Frequency Identification. A tag utilizing RFID is about the size of a postage stamp and can be embedded or attached to an object to be able to track or identify it. RFID tags are small chips that have a unique identifier and have an antenna that receives radio frequencies. When one of these tags comes within the range of an RFID reader or sensor, the reader scans the tag and deciphers the information held within it.

There are three different forms of RFID; UHF RFID (Ultra-High Frequency RFID), HF RFID (High Frequency RFID), and LF RFID (Low Frequency RFID). Out of these three forms of RFID, only two of them are applicable for this project. Those two are UHF RFID and HF RFID. LF RFID is rejected as the range is far too short for the scope of this project.

UHF RFID operates within the 902 – 928 MHz band in the United States. These UHF RFID tags can be scanned by RFID readers between 2 – 80 feet. This scanning range depends upon the antenna size of the tag itself, as well as the power of the RFID reader. The way that these read distances are achieved is using a phenomenon called backscatter. The RFID tags reflect the reader signals in a certain way to boost the range at which the RFID reader can pick up the signal from the RFID tag.

HF RFID operates at 13.56 MHz and has a much shorter range than UHF RFID. A typical operating range of HF RFID is about three feet. This is because HF RFID works on the phenomenon of induction rather than backscatter. While the HF RFID has a much shorter effective range than UHF RFID, it might still be of use in this project if properly implemented.

A rather intimidating problem of RFID is that of security. An RFID tag works with any RFID reader and can be tracked without the knowledge of the RFID tag owner. It is also difficult to make RFID tags secure due to their weak computation and communication power. These faults can lead to RFID inadvertently causing an invasion of privacy.

The price of an RFID reader depends upon whether UHF or HF RFID is preferred. A typical UHF RFID reader costs about 230 dollars, while a typical HF RFID reader costs about 35 dollars. These prices may vary slightly depending on range or design implementation.

Advantages of RFID:

- Passive. Tag does not use power.
- No line-of-sight limitations
- Can scan from a distance and through obstacles
- Wear and tear resistant
- Scans quickly

Disadvantages of RFID:

- Costly
- Can have problems if scanning through metal or water
- Unauthorized devices can scan or change information on RFID tags without knowledge of owner

Table 3 - UHF RFID Characteristics

Parameter	**Value**
Operating Voltage	3.5 to 5.25 VDC
Operating Frequency	902 – 928 MHz
Power Consumption	2 to 5.5 W
Idle Power Consumption	0.32 W
Operating Temperature	-20 to 60 C
Scan Distance	30 ft.
Dimensions	46 mm x 26 mm x 4.0 mm

Table 4 - VHF RFID characteristics

Parameter	**Value**
Operating Voltage	3.0 to 5.0 VDC
Operating Frequency	13.56 MHz
Current Consumption	55 mA
Idle Current Consumption	4.4 mA
Operating Temperature	0 to 70 C
Scan Distance	9 ft.
Dimensions	1.65 x 0.735 x 0.17 in.

3.2.2.3 Ultrasonic Sensors

An ultrasonic sensor is a device which is used to measure distances between objects by utilizing sound waves. An ultrasonic sensor consists of two parts: the transmitter that transmits the ultrasonic signal and the receiver which essentially acts as a microphone and reads the signal once it has returned from the object. The working principle of an ultrasonic sensor is that the transmitter will send out an ultrasonic signal and begin timing once it has been sent. Once the ultrasonic signal encounters an obstacle, it will bounce back and be collected by the receiver. The receiver will stop timing once this ultrasonic signal has been received. If the speed of the ultrasonic wave is known, which it should be considering the sensor has been manufactured to a certain specification, then the distance to the object may be calculated.

This is relevant to our project because the ultrasonic sensor may act as a form of car counter. The ultrasonic sensor would send out a wave and detect whether a car was entering the garage. If a car were detected entering the garage, the sensor would send a signal to the main microcontroller telling it to increment the number of cars in the garage. If no signal were detected, then no cars have entered and no such signal would be sent.

Ultrasonic sensors can more accurately track surfaces which are rigid and smooth rather than those which are malleable and rough. It is this attribute of ultrasonic sensors which makes them suitable for our project. The surfaces of cars are rigid and smooth as well, this makes them a perfect candidate for ultrasonic wave reflection.

One defining trait of ultrasonic sensors is that the further an object is from the sensor, the less of the ultrasonic wave that returns to the sensor. This means that the sensor is more accurately able to track those objects which are closer to it than those which are further. This poses a potential problem within the scope of this project, as the height from the top of the entrance of the garage to a car is about

six feet. This potential problem may be fixed however with certain clever implementations, such as mounting the ultrasound sensor on the side of the garage entrance rather than the top. The distance from the wall to a car entering a garage is significantly smaller than the distance from the top to a car. Due to the working range of ultrasonic sensors being 0.07 feet to 10 feet, this aspect of ultrasonic sensors should not be a problem when considered within this project. Specific implementations of sensors will be discussed further in a later section.

The price of an ultrasonic sensor ranges between three dollars to thirty-five dollars. This price depends on the range that the sensor can measure and the power usage of the sensor. The farther away that the sensor can send ultrasonic signals, the higher the price of the sensor.

Advantages of ultrasonic sensor:

- Relatively inexpensive
- Simple, does not introduce much complexity to system
- Sensor does not rely on optical reflection from object
- Sensor works the same for all colors

Disadvantages of ultrasonic sensor:

- Cannot be mounted at an angle. Sensor should be placed in a manner so that it is perpendicular with the object it is sensing, in this case cars
- Certain materials can interfere with ultrasonic waves by absorbing them

The following technical specifications are based on the Parallax Ping Ultrasonic Range Sensor 28015.

Table 5 – Ultrasonic Sensor Parameters

Parameter	**Value**
Operating Voltage	5.0 VDC
Operating Current	35 mA
Communication	Positive TTL pulse
Dimension	0.81 x 1.8 x 0.6 in.
Operating Temperature	0 to 70 C
Scan Distance	1 inch to 10 feet

3.2.2.4 Infrared sensors

Infrared radiation is a form of light which has a wavelength that is longer than regular visible red light. The ranges in which infrared radiation can be found encompass near infrared, mid infrared and far infrared. These wavelengths range from about 710 nanometers (near infrared) to 100 micrometers (far infrared).

To understand infrared radiation, a phenomenon known as black body radiation must be understood as well. Any object gives off light in accordance with its temperature. As the object gets hotter, it will give off shorter and shorter wavelengths of light. The planet Earth gives off infrared radiation in the range of nine to ten micrometers. Animals which are warm-blooded, such as humans, also give off this infrared radiation. This infrared radiation can be utilized in such a fashion as to detect motion or warmth.

Infrared sensors are sensors that are designed specifically to pick up this infrared radiation. The infrared radiation is converted into an electric signal or current and this signal is then detected by a volt or amp detector. One of the many properties of light-emitting diodes is that a wavelength of light is emitted when an electrical current is applied to them, however this property also works in reverse. That, when light-emitting diodes are exposed to a wavelength of light, they also produce an electrical current.

It is this property of light-emitting diodes that allows them to be used as infrared sensors. Two light-emitting diodes are required in order to make a functioning infrared sensor. Infrared sensing via light-emitting diodes is achieved in the following manner: the first infrared light-emitting diode is set up in such a way as to produce infrared waves and the second light-emitting diode is configured to output an electrical signal when it receives an infrared signal. Whenever an object comes into range of the light-emitting diode which is emitting the infrared waves, the object will block the infrared waves and reflect them back to the second light-emitting diode which then produces an electrical signal.

Infrared detectors and emitters are nearly ubiquitous. However, one problem that infrared sensors run into is that water can absorb infrared waves in a nearly absolute manner.

Infrared sensors may be used in this project as a sensor for counting cars when they enter or exit the garage. This would work as described above, where the car would obstruct the infrared wave and reflect it back to the sensor, the sensor would then send an electrical signal and increment or decrement a counter depending on whether the car was entering or exiting.

The range of infrared sensors is between 0.33 feet to 16.4 feet. This range is large enough that there would not be a problem in using these sensors as car counters to count cars entering and exiting the garage.

The price range for infrared sensors is between fifteen to twenty-five dollars. This price increases as the range of the sensor increases. The relative inexpensiveness of infrared sensors makes them an attractive option for sensing.

Advantages of infrared sensors:

- Inexpensive
- Greater noise immunity when compared to other sensors
- Not many regulatory constraints
- Simple design

Disadvantages of infrared sensors:

- Line of sight: transmitters and receivers must be almost directly aligned
- Can be obstructed by common objects, such as people
- Range may suffer at longer distances due to less signal being reflected
- Sensitive to light and weather, rain severely hampers performance
- Not as fast at transmitting data as other sensors

Table 6 – Infrared Sensor Parameters

Parameter	**Value**
Supply Voltage	4.5 to 5.5 V
Current Consumption	30 mA
Output Voltage	3 to 1.4 V varying with distance
Dimension	58 x 17.6 x 22.5 mm
Scan Distance	3 to 15 feet
Operating Temperature	-10 to 60 C

3.2.2.5 Smart Video

Smart video works under the principle of pattern recognition. A preset pattern or shape is programmed into a computer and a computer analyzes footage from a video camera to determine whether this pattern or shape is present within the video.

Smart video may be used as a form of “sensing” when a car enters or exits a garage. A camera films the entrances and exits of the garage and then this data is sent to a computer or processor. The computer analyzes the footage to see if any object in the video matches the profile of a vehicle. If any object in the video does match the profile of a vehicle, then the computer will increment or decrement a counter, the counter depending on whether the car entered or exited the garage.

Prices for this type of system vary greatly as it can be either bought already implemented or programmed and implemented independently. An obvious case of cost versus time is then presented. This system is rather expensive if it is bought from a third-party company but it would take a rather long time to implement if it were to be programmed and implemented independently.

The largest advantage of this kind of system is that it can work no matter what type of material the vehicle is made from and it is not thwarted if the vehicle is covered in liquid, that is to say that this system does not depend on any physical properties of the vehicle outside of the shape or pattern of the vehicle. This means that, unlike most sensors, this system provides more accurate readings and less uncertainty in its results.

Advantages of smart video:

- Works well in any weather
- Allows manual monitoring by an operator
- More robust than other sensors

Disadvantages of smart video:

- Expensive if bought from third-party company
- Time consuming if implemented independently by project team
- Requires more computing power than other sensors

3.2.2.6 Magnetometer

Magnetometers are devices that measure solely magnetic fields. Magnetometers are sensors which measure magnetic flux density. Magnetometers can detect fluctuations in the magnetic field of the Earth, since magnetic flux density in the air is directly correlated to magnetic field strength.

Magnetic materials will change these magnetic flux lines within the magnetic field of the Earth. Magnetometers can sense these changes by magnetic materials when they are caused.

One feature of magnetometers is that they measure flux density only in the area where the magnetometer is situated. Magnetic fields lower in intensity with proportion to the cube of the distance from the object of reference. This trait of magnetic fields means that the largest distance which a magnetometer can detect an object disturbing the Earth's magnetic field is directly proportional to the cube root of the sensitivity of the magnetometer. The sensitivity which corresponding to magnetometers is typically measured in Tesla.

There are two categories which magnetometers may fall under. The first is vector magnetometers; these magnetometers measure the magnetic flux density in a certain direction in three-dimensional space. The second category which magnetometers fall under is scalar magnetometers. These magnetometers measure only the magnitude of the magnetic flux density; this type of magnetometer does not have the capability to measure the direction of magnetic flux density.

One of the problems that magnetometers have within the scope of this project is that, because magnetic field strength drops off in proportion to the cube of the distance from the magnetometer, magnetometers are usually very short range. This means that typically magnetometers must be embedded within the road or buried beneath the road's surface. There are, however, magnetometers which are strong enough to detect changes in magnetic flux density from a relatively far distance. These magnetometers, unfortunately, are almost universally prohibitively expensive.

The main advantage that the magnetometer has over other sensors, however, is that not many objects will falsely trigger it. In this project, the object being tracked is vehicles. Most vehicles disturb the Earth's magnetic field as they travel. Due to this property of vehicles, they can be tracked by the magnetometer. This also means that the magnetometer will only track vehicles that pass over it, rather than other common objects, like humans or animals. This means that the sensor will almost never get a false positive and falsely increment the counter which counts

how many vehicles exit or enter the garage, making it more accurate and reliable than most other sensors considered herein.

Advantages of magnetometer:

- More accurate and reliable than other sensors
- Almost immune to false positives
- Long sensor life
- Low power
- Easy to use communication system

Disadvantages of magnetometer:

- Inflexible in regards to physical implementation of sensor; must be embedded within road or buried beneath it
- Would cost a lot to acquire one with enough strength to measure at distances this project needs
- Being physically located beneath road means communications between sensor and microcontroller would have to be entirely wireless, effectively making entire system more complex, limited, and expensive.

The technical specifications in Table 7 are based on the M100 sensor.

Table 7 – M100 Magnetometer Parameters

Parameter	**Value**
Supply Voltage	3.6V Battery
Sampling Rate	128 Hz
Transmit/receive bit rate	250 kbps
Dimension	74 x 74 x 56 mm
Scan Distance	3 to 15 feet
Operating Temperature	-40 to 85 C
Channel Bandwidth	2 MHz

3.2.2.7 PIR Sensors

PIR sensors are motion sensors which are typically used to detect human motion within its range, however these sensors can be used to detect any motion within its range so long as the motion brings a change in temperature along with it.

PIR sensors are a beast with many names. They can be found under the names of passive infrared sensors, pyroelectric sensors, or even IR motion sensors.

PIR sensors are very similar to the infrared sensors detailed above, but with small, though very important, changes. Rather than emitting a wave and detecting a break in this wave, a PIR sensor measures the levels of infrared radiation of the surrounding area constantly. Almost everything in existence emits some infrared radiation, and the temperature of an object correlates directly with the amount of infrared radiation it will emit. The hotter it is, the more infrared radiation it will emit, and vice versa.

A PIR sensor is designed to detect a change in the ambient infrared radiation levels of the surrounding which it is sensing. For this reason, a pyroelectric sensor is

actually split into two halves. The two halves are designed and wired in such a way as to cancel each other out under normal conditions. The reason for this design is that the sensor is attempting to detect motion, not the average infrared radiation levels of the surrounding area. If one half of the device sees more or less infrared radiation than the other half of the device, then the output will either go high or low. In figure 3 below can be found a visual description of the working mechanism of a PIR sensor taken from Adafruit.

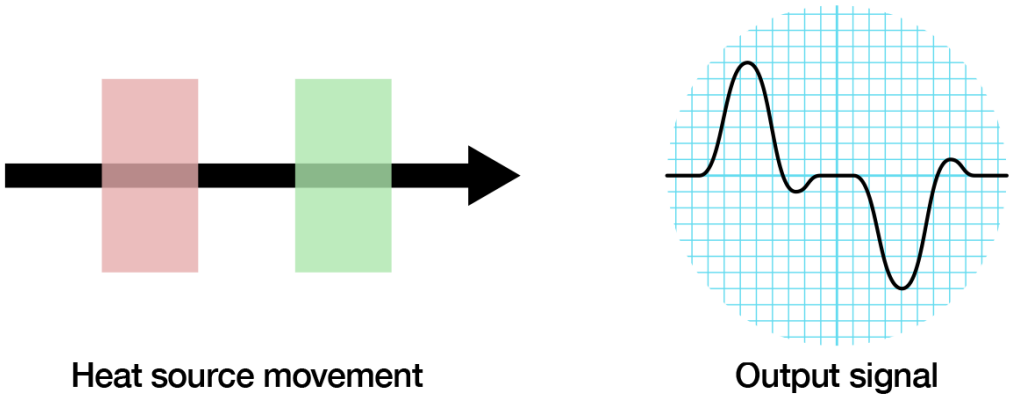


Figure 3 - PIR Movement vs Output [1]

Thus ends the basic description of a PIR sensor. What follows is a comparison of the advantages and disadvantages of the PIR sensor in question.

Advantages of PIR sensor:

- Extremely low cost
- Extremely low power
- Easy to interface
- Extremely durable

Disadvantages of PIR sensor:

- Wide lens range is a detriment to this project as it will sense a wide area rather than the small area needed to detect a car passing
- Cannot discriminate between vehicles and other objects that emit infrared radiation, such as animals or humans
- May not be able to detect vehicles which have been cold started

3.2.2.8 Optical Sensors

Photodiodes and photoresistors will be included under the same section because they accomplish the goal of sensing through almost similar mechanics. Photodiodes generate current and voltage when exposed to light while photoresistors decrease in resistance when exposed to light. The main point being that both photoresistors and photodiodes are sensitive to light and when exposed

to light, change their characteristics depending on the intensity of the light. Therefore, photoresistors and photodiodes can be wonderfully and efficiently utilized as optical sensors for the purposes of this project.

Photodiodes are typically made of silicon. Photodiodes convert light which is incident upon them into electrical current. Photodiodes typically consist of a shallow diffused p-n junction. Most photodiodes manufactured in the modern day are made by the processes of planar diffusion or ion-implantation methods.

The physics intrinsic to a photodiode consist of the following, whenever photons with an energy greater than 1.1 electron volts, which is the bandgap of silicon, are incident upon the photodiode, the photons are absorbed by the photodiode and electron-hole pairs are generated within the photodiode. These electron-hole pairs then drift apart from each other and when the carrier which is a minority within the photodiode reaches the junction between the p and the n sides, they are transferred across by the mechanics of an electrical field.

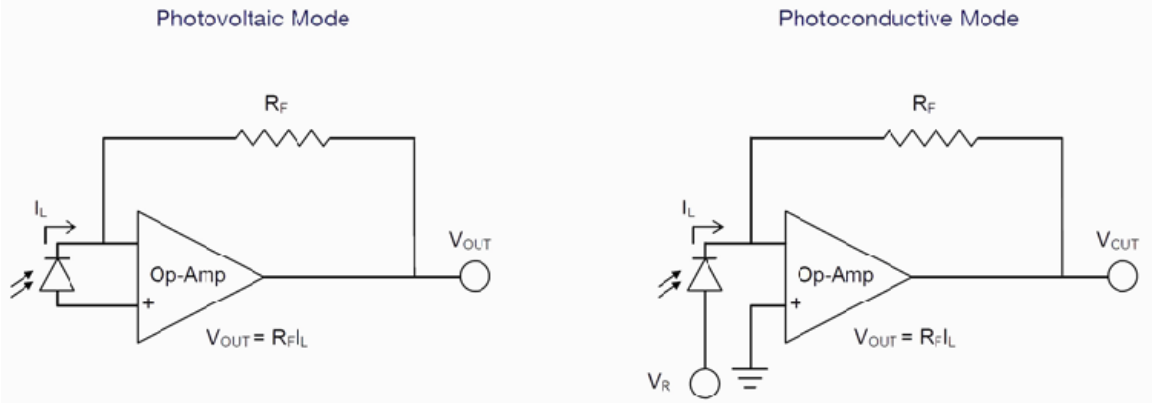
If the p and the n sides are joined together through an electrical connection, then an electrical current will flow through this electrical connection.

Photodiodes are usually sensitive to light within the spectral range of two hundred nanometers, which is near ultraviolet light, to around 1100 nanometers, which is near infrared radiation. The responsivity of photodiodes is measured in amperes of photocurrent that is generated per each Watt of light that is incident. Light levels in the grand majority of areas where photodiodes are applied usually range from picoWatts to milliWatts, which will generate a photocurrent of pico-amps to milli-amps respectively. The responsivity of a photodiode will vary with the wavelength of the incoming light. A silicon photodiode’s responsivity is typically well suited to those light sources which emit light in the range of ultraviolet to near infrared, such as laser diodes.

Silicon photodiodes can be operated in two modes. The first mode is known as the photovoltaic mode. In this mode, the photodiode is unbiased. The benefit of the photovoltaic mode is that the current in the dark when the photodiode is not receiving photons, also known as the dark current, is at a minimum. This means that the current when there is no light shining on it will be very small.

The second mode is known as the photoconductive mode. In this mode, an external reverse bias is applied. The main advantage of the photoconductive mode is that the switching speed of a photodiode is fastest when it is in the photoconductive mode. Figure 4 below details circuits of each photodiode mode.

Figure 4 - Modes of Operation of Photodiode



Photoresistors operate under almost the same principle as photodiodes. Photoresistors are made of a semiconductor material which has a large resistance and few free electrons. When photons are incident upon this semiconductor material, the photons are absorbed by the semiconductor material and the energy from the photons is transferred to the electrons which excites the electrons and ultimately causes the material to have a higher conductivity while simultaneously lowering the resistance of the semiconductor material.

The resistance of photoresistor. is typically between hundreds of Ohms and thousands of Ohms. This resistance depends upon the amount of light shining upon the photoresistor. When the photoresistor is in a bright light, the resistance drops to only a few hundred ohms. However, when the photoresistor is in low light or no light at all, the resistance of the photoresistor is typically several mega-ohms. Photoresistors are typically made of cadmium sulfide, as this semiconductor material is very cheap and performs very well as a photoresistor.

Overall, for this project, photodiodes would be a better choice than photoresistors. This is due to the fact that photoresistors need a power source that would allow current to travel through them and photodiodes generate their own current. This means that any optical sensing circuit that made use of a photoresistor would require not only a power supply for the photoresistor, but also the laser which is used to trigger the photoresistor would also need to be supplied power. This not only makes the overall schematic more complicated, as power would have to be routed to two places, but it also increases cost due to having to design a power supply capable of supplying power in this fashion.

The following lists the advantages and disadvantages of photoresistors and photodiodes as optical sensors.

Advantages of photoresistors and photodiodes

- Fast response time
- Silicon photodiodes are sensitive to common lasers
- High responsivity

- Multiple can be used to increase sensitivity
- Extremely cheap
- Photodiodes are passive

Disadvantages of photoresistors and photodiodes

- Small area, although this can be remedied by using multiple
- No internal gain

3.2.3 Sensors Implementation

Due to the circumstances surrounding the implementation of the sensors described above, specific ways of physically implementing the sensors will be considered within this section.

Our system is designed to be installed within a garage. Taking UCF's garage as a basis for our system and acquiring our measurements from there and then extrapolating, it can be determined that the height from the top of the entrance/exit to the ground is sixteen feet. The distance from wall to wall is ten feet. Embedment in the road is too costly and time-consuming and any implementation upon the floor of the garage or the road would require physical alterations to the road and this is infeasible due to the cost and time, not to mention manpower, attached to such an undertaking. The optimal placements for our sensor would then be either the top of the entrance/exit to the garage or the walls of the entrance/exit to the garage.

When considering any implementation upon the top of the garage, it must be taken into consideration that the average height of a passenger car is about five feet. This means that any sensor implemented upon the top of the garage must have a range of at least twelve feet. However, the distance from the wall to a passenger car is approximately five feet**,** and any sensor implemented upon the walls of the garage would be able to have a shorter scanning range. Therefore, it is the wisest and most economical choice to mount the sensor chosen upon the wall of the entrance to the garage, to acquire the best sensor results for the lowest price. This lower price is due to not needing as strong of a sensor with a large scanning range, which brings down not only cost but power requirements as well.

3.2.4 Sensor Consensus

As should now be evident, there is an incredible variety of sensing technologies that are apt for vehicle detection. All aspects of each sensor listed in the sections above were considered thoroughly by the project group. Every advantage and disadvantage of each individual sensor was meticulously pored over, to make the right choice. There was no clear-cut answer, as each sensor had obvious and sometimes not-so-obvious advantages and disadvantages inherently tied to it, be

it either through the mechanics behind the sensing itself or the physical locations each sensor is restricted to in the case of placement within a garage.

It is through much deliberation, and perhaps just a hint of conviction, that the project group can proudly and confidently declare that optical sensors shall be the sensor type that has been chosen to be used within this project. It was no easy decision, but the reasons for the choosing of optical sensors shall be laid out.

First, the range of optical sensors falls nearly perfectly within the range needed to detect vehicles entering and exiting the garage. Second, optical sensors are rather inexpensive compared to the alternatives, while still being able to provide the quality of detection needed for the project. Finally, optical sensors are entirely passive, notwithstanding the power needed for the laser in the system.

Although there may be some qualms about weather affecting the performance of optical sensors, the project group is quite confident that, using clever placement within the entryways of the garage, weather should not affect the sensors very heavily.

Another point may arise that optical sensors can easily be triggered by common objects, such as humans or animals. The project group is, again, extremely confident in its abilities to collectively come together and put together a solution regarding placement of the infrared sensors that would make this argument null and void.

The specific photodiode and laser that shall be used for optical sensing are the HiLetgo 5 mm photodiode and the GeeBat mini laser dot diode, respectively.

3.3 Displays

The Display is used to give feedback to the user while they are using the system to input their information. Since the display is one of the primary ways that users will interact with the system, it may be worth it to opt for a high-end display, to give a feeling of ease of use. Furthermore, when considering the cost of a display we must also consider what kind of connection is used to connect it to a microcontroller.

A display which requires an HDMI connection should be considered more expensive than one that uses a simple i2c or SPI interface. When considering the size of the display, we will be looking for a display of around 7 inches. Since 3.5” was enough display for all iPhones prior to 2012, 7” should be more than enough to implement any kind of interface we want.

3.3.1 Raspberry Pi foundation 7” touchscreen

At $80, this display is quite pricey, however it comes with a lot in it that we need. It is simultaneously an input and output device which adds simplicity to our design. Furthermore, a touch interface is something that is very familiar to almost

everybody in today's world, lending to a high degree of usability. Lastly this display in particular is well supported with software by the raspberry pi foundation and the broader raspberry pi tinkering community. At around $80, this option does seem to be the most expensive, but it may be worth the investment. This is a decision we must make as a group.

The display connects to a raspberry pi, or other compatible board, via an adapter board and a DSI cable. The adapter board helps to convert power and signal differences between the pi and the touchscreen. Since this kind of connection is not common anywhere except on raspberry pi boards, using this almost certainly means that we will have to use a raspberry pi as our main processor, however the extent that this is a problem is not clear.

3.3.2 256 X 128 Graphic Monochrome LCD (HG25504)

This display sacrifices expressiveness for price, providing a monochrome picture, while coming in at a very affordable $12. It is controlled by SED 1335 LCD controller, which is built into the product. It requires 12 digital input lines, which would take up a significant amount of GPIO space on our microcontroller. Additionally, this display has no backlight, which could make it very difficult to use at night. Alternatively, there are also models available which include a backlight, which are only slightly more expensive.

Having a graphic display, as opposed to a character-line display may allow for some interesting interface designs such as fine-grained scrolling, which is a more pleasurable way of scrolling through data than jumping between lines. This display is pure monochrome, only allowing you to adjust the amount of contrast, but not the actual color of the display. This may result in a drab interface than with a full color screen.

3.3.3 LCD Character DisplayModule 20 x 4, White on Blue (HD44780)

This is a character LCD with twenty columns and four lines, based on the popular HD44780 LCD controller. The display, and others based on the HD44780, supports writing basic characters to a dot-matrix LCD. It uses a simple interface of only 11 digital input pins to synchronously update the display. There is 1 Register Select pin (RS) which selects whether a data or instruction register is updated.

Next there is the Read/Write Select pin (R/W), which selects whether the operation will write to a register or read from it. The Enable (E) pin toggles the ability to write to the registers. Last are the 8 Data pins (D0 – D7), which represent the data that is written to the register. In addition to the digital input pins there are a few pins that are treated as an analog input. First is the contrast pin, labelled as V0, sets the amount of contrast on the display based on the voltage applied to it. Many displays on the market offer a built - in driver that receives information over UART and interfaces these pins accordingly.

Typically, a rotary potentiometer is used to adjust this value on the fly. Additionally, there is a pin to control the brightness of the backlight, which can be adjusted using PWM. The basic process for controlling the display is to put the data in the data registers, and put some instructions in the instruction registers.

This display has many desirable qualities: the large number of rows and columns allow for a variety of different interface styles, which might require more screen space. Furthermore, the LCD controller (the HD44780) is very popular, especially among users of the Arduino platform, and so there is ample software support for it, such as software libraries and code examples which demonstrate most of the functionality available from the device.

3.3.4 Summary of Display Device Options

Below, Table 8 summarizes the tradeoffs of each display option.

Table 8 – Summary of Display Device Options

Device	Price	Connection	Usability
Touchscreen	$60-$80	CSL connection on Raspberry Pi	Best
Graphic LCD	$12	12 GPIO pins	High
Character – oriented LCD	$5	UART	Good

3.3.5 Display Selection

After deliberation, group five decided that the character oriented LCD display was the best choice (Pictured below, Figure 5), being the most economical. Not only does it have the lowest unit cost, but using only a single UART connection places the smallest burden on the controller.

Figure 5 - Character LCD Display

3.4 Audible Feedback

Providing an audible feedback to the user allows for an easy way for the user to interpret what is happening inside the system. Complex sounds are not necessary, since most information will be given to the user on the display, rather than spoken. Therefore, some sort of buzzer or beeper would be perfect for this application.

Ideally, the buzzer could be powered directly from the microcontroller, without any other controlling elements. Therefore, a buzzer with low operating voltage and low operating current would be ideal, as we can always step down an output voltage with a simple voltage divider circuit, but increasing a voltage level is typically more difficult If necessary the control pin may be able to be oscillated using a PWM signal, but it would be easier to not have to do this.

The buzzer should be loud enough to be easily heard by the user through the casing, but not too loud that it is overwhelming. If our requirements cannot be met in a simple and inexpensive way, we can scrap the buzzer all together, since audible feedback is not completely necessary.

3.4.1 Electromagnetic Beeper (Chinasound # CEB12A-05BP)

Per allelectronics.com, a vendor of this item, this beeper operates between 4-7V, at 30mA, which should be easy to achieve directly from a microcontroller. They claim that, when energized it emits a 2.3 kHz tone at 85 dB. 85 dB is probably louder than we need, since, per industrialnoisecontrol.com [2] any noise in the upper 70dB range is annoyingly loud to most people.

3.4.2 Piezoelectric Beeper (FY14-G)

This beeper operates by driving a current through a ceramic element which is susceptible to the piezoelectric effect. What this means is that when an electric voltage is applied to the ceramic, it deforms.

When a square wave signal is sent to the element, it produces a tone of approximately the same frequency. This is what is called a piezoelectric transducer. A piezoelectric beeper is the same device, with a feedback element which is attached to a transistor. When the feedback element is activated it, causes the transistor to shut off current to the main element. This mechanism lets the beeper be driven in a simpler way, requiring only a voltage, not a square wave supply [3].

Per Allelectronics.com, a vendor of the FY14-G, this buzzer can operate at 3-18V, and draws 7mA. Only the low end of the voltage specification will be easily attainable directly from a microcontroller, so depending on how loud the buzzer is at that voltage will decide how effectively it can be used. Allelectronics also claims that the buzzer can generate an 80dB tone, which should far exceed our needs.

3.4.3 Summary

Table 9 compares the operating characteristics two options of audible feedback we considered.

Table 9 - Operating parameters of Buzzers

Technology	Operating Voltage	Operating current
Piezoelectric	3-18V	7 mA
Electromagnetic	4-7V	30 mA

It is obvious from this comparison that the piezoelectric beeper has significantly less demanding operating parameters.

We have decided not to pursue adding this device to the project, favoring simplicity. However, if we were to use one, we would choose the Piezoelectric device, for its more favorable operating characteristics.

3.5 User Input Devices

Users will be required to input their spot number, UCF PID, and name into the system, therefore timely input of letters (uppercase and lowercase) as well as numbers will be required. Basically, a fully-featured keyboard is required. This is unless we decide that only a number is required to access the system, such as the individual's PID or NID. In this case we can use a simple numeric keypad.

3.5.1 Rii i8+ 2.4GHz Mini Wireless Keyboard with Touchpad Mouse, LED Backlit, Rechargeable Li-ion Battery

This device is a full, standard, keyboard complete with a trackpad, arrow keys, and mouse buttons. It also had an LED backlight for use in dark conditions. At $21 it is also well within our price budget. It connects to a computer using a 2.4GHz USB receiver, however a Bluetooth option is also available.

It is charged from a USB cable that connects to the computer. Its main drawbacks are that it comes in only one shape (we would have to fit the case around it to cover buttons that were not used), and that it requires an expensive USB port. Additionally, the keyboard is meant to be a handheld device, and as such the buttons are rather small, and would be difficult to use on a wall-mounted system.

3.5.2 Portable Mini 3.0 Wireless Keyboard, Handheld Remote Control Keyboard for PC smartphone and more Black

This keyboard is like the previous one, being meant to be a hand-held device. This one does not have any extra feature other than the keyboard though, and would likely not require that the case be fit to cover the extra features.

Furthermore, it connects through Bluetooth, not USB saving that port for something else. Instead, this keyboard uses USB only for charging, and can be plugged directly into a powerline, given the appropriate adapter.

3.5.3 Custom made keyboard using tactile buttons purchased online

Small momentary switches can be purchased in lots of places for a few cents each. Thus, this option is by far the least expensive of the input methods. Furthermore, it would also be the most customizable of all physical button based input methods.

3.5.3.1 Construction of custom keyboard

To construct the keypad, we would first design the intended button layout, so that each button is spaced appropriately, and so that they all have a desirable size, etc. Next the layout would be incorporated into a Printed Circuit Board which would be ordered from an appropriate vendor, as well as the Mini-Tactile Pushbuttons.

These buttons were chosen because they are inexpensive (about $1.00 for 3), and provide a tactile feedback. Next, we would design the button-faces, which are the plastic bits that the user would be able to see. The button heads could be made using UCF's 3D printer for plastic buttons, or on the CNC Milling Machine, for metal buttons. Each button face would have a raised portion which displayed the letter it enters.

This raised portion could then be easily painted a different color than the rest of the button face. Alternatively, we could print custom stickers and place them on the button faces.

3.5.4 Button Matrix encoded keypad (4 x 4)

If we relax the requirement that users must enter their name, we can just use a numeric input device, which is an order of magnitude less expensive than alphabetic options. A 16-button keypad is available from all-electronics.com for $12.50. It has numbers 0-9, up/down keys, a clear, help and enter and "2nd" buttons. This combination of buttons would be ideal for navigating an interface of a small number of menus, which may require scrolling.

This device communicates with a micro controller using 8 pins in what is called "matrix encoding". With a matrix encoded keypad, the device uses one pin for each row and one for each column of buttons. When the user presses a button, it creates

an electrical connection between the pins which correspond to that button's row and column. On the microcontroller, inputs are detected by setting one row or column pin to HIGH, and checking if any of row pins have changed from LOW to HIGH. This process is then repeated for every row or column. Figure 2.5.1 shows the idea behind a matrix encoded keypad.

Matrix encoding has the advantage that is uses less GPIO lines to the micro controller. Instead of requiring a dedicated GPIO pin for every button, only the square root of the number of buttons is required. There is an inherent limitation though. If two buttons are pressed at the same time, which occupy a different row and a different column, the microcontroller will not be able to correctly identify which are pressed. This limitation will not prove to be an issue, however, since most interfaces never require pushing more than one button at a time.

3.5.5 12 - Button Matrix encoded keypad (3 x 4)

While a 16-button keypad may provide more different buttons for the user, 12 button keypads seem to be much more common in the market. While 16-button keypads seem common on electronics websites, local hardware stores seem to only stock the 12-button varieties.

3.5.6 2.5.5: Input Devices Summary and Part Selection

It seems to be a trend that stand-alone keyboards with full alphanumeric inputs are not widely available except as fully featured keyboards for use with desktop computers. The only viable option for satisfying this requirement is to either fabricate our own alphanumeric keyboard, or use a touchscreen. Fabricating a keyboard is no simple task, as we have already demonstrated, and would still be more expensive than simpler input methods. A touch-screen, despite being an integrated input and output device, has been shown to be as much as twice as expensive as a dedicated monochrome display and a dedicated keypad. Table 10 summarizes the options we explored.

Table 10 – Input Devices Summary

Device	Price	Capability	Connection
Rii Wireless keyboard	$21	Letters, Numbers, Mouse	Wireless with USB dongle
Portable Mini Keyboard	$15	Letters, Numbers	Bluetooth
16-button Keypad	$12.50	Numbers, up/down, enter, clear	Matrix encoded wires
12-button keypad	$2.00	Numbers, H,P	Matrix encoded

Group five has decided to use the 12-button keypad, pictured below. This is because it has the lowest price of the options. The keypad we purchased (pictured below, Figure 6) was the only one available from Skycraft, a local electronics hardware store in the Orlando area. It has an H and a P where most keypads have a * and #, however we are confident we will find one with the normal arrangement. If necessary, we can add additional separate buttons for additional functionality, such as an enter key or a clear button.

Figure 6 - The 12 -button keypad

3.6 Faster Input Devices

Users who repeatedly park in the garage will quickly get annoyed if they have to enter all their information every time they park. This highlights the need for users to be able to quickly enter their information. We explored several approaches to this, including reading their student ID, incorporating a camera module to read QR codes, fingerprint scanners, NFC tags,

3.6.1 Magnetic Strip Card Reader

Although a card reader could not be used as the single input device, it could be used as an alternative to manually inputting a student's PID, by allowing them to simply swipe their student ID. This would be a good boost to the usability of the system, and could significantly up the process of logging your parking spot. These devices are typically very inexpensive, which is a plus.

The card reader communicates using USB, which is a complicated port, meaning we would need a much more powerful controller. If the USB connection is not an issue, a card reader can be purchased for $20. Another advantage of using Magnetic Stripe card reading is that there is already significant infrastructure supporting it, as UCF uses the technology extensively for a variety of purposes, such as printing stations, testing centers, gym access, etc.

MagStripe card readers work by analyzing a strip of magnetic north and south poles, which represent 1's and 0's. More in-depth technical details are discussed in the relevant technologies section.

3.6.2 Camera Module for QR Codes

QR codes are an attractive option because they cost only the price of a small amount of ink, and can be placed anywhere. Users can attach a sticker to the back of their phone, put it on a small plastic keychain, keep a photo on their phone's internal memory, or any of a host of other options. To use them, they simply hold the code in front of a camera, which detects the code, and reads its data, like a traditional barcode. More in-depth technical information about QR codes can be found in the relevant technologies section of this paper.

To allow users to line up their code with the camera quickly and easily, the camera used must support streaming video as well as taking pictures. Implied is the requirement of some sort of graphic display (either color or monochrome) to show the user the camera's input, so they can adjust the positioning of the code as needed.

3.6.3 TTL Serial Camera

This camera module sold by adafruit.com for $40 is a high-quality camera module that streams video and can take snapshots of that video and transmit them over a TTL serial (UART) link. It supports resolutions of 640x480, 320x240 or 160x120. The use of UART as a communication line makes this device very attractive, since UART connections are simple, well-understood and widely available on a variety of microcontrollers.

The biggest drawback of this unit is that it only streams video over an NTSC connection, which is an analog standard usually used to control televisions. Being an analog format means that connecting to a micro controller.

3.6.4 Zero Spy Camera for Raspberry Pi Zero

This camera is much cheaper than the serial camera, at $20, but imposes the restriction of a CSI interface connected to the MCU. CSI is a connection that can transfer information at high data rates, but is only ever used to transfer pixel data. Most models of Raspberry Pi's have this interface, but many other microcontrollers and microcontroller boards do not. Additionally, using the CSI connector for the camera means that it cannot be used for anything else, including a display, unless

there is another one available. On the positive side is that this camera module requires only one connection, rather than two, in the case of the Serial camera.

The Spy Camera has a native resolution of 5 megapixels, which should be enough to read a QR code accurately from reasonable distances, with reasonable precision. We believe this from having experience with cameras of similar resolution.

3.6.5 Fingerprint Scanner (Sparkfun product # 11792 or #13007)

Fingerprint scanners could be a very convenient way for users to log their information. If their finger is stored in a database somewhere all they would have to do is scan their finger, enter the spot number and go on with their day. Furthermore, users are extremely unlikely to ever misplace their fingerprint, adding to further convenience.

The sensors utilize a smackfinger 3.0 algorithm to identify or verify the finger. The sensors can store the details of up to 20 or 200 unique finger models, depending on which version is used. Additionally, the sensors can transmit the stored finger models to a controller board, which we could then store in a database online and make accessible to systems in every garage.

In order to overcome the limitation of only storing a limited number of finger models on the sensor itself, the microcontroller would have to have a local backup of all known valid fingers, and try them all against the user's finger until it found a match. Another nice thing about these scanners is that they use a simple UART based protocol for control, allowing for easy integration with a microcontroller.

Drawbacks include unreliability, as even high-end fingerprint scanners are known to give false negatives under certain conditions (such as greasy/dirty fingers). If these errors were to happen to users in the field it could cause frustration, however we are confident that this will rarely cause them to not use the system, instead using another method to input their data.

3.6.6 NFC Tags (PN532 NFC reader)

All smartphones these days have an NFC transponder, in order to enable them to facilitate mobile payments. Mobile phone platforms have also opened up this hardware to application developers, since they also make money from app revenue. This provides us with a potentially cheap and easy way to let users input their data from a device they typically carry in their pockets at all times. Small passive NFC tags can be purchased for pennies each, and embedded inside something convenient such as a keychain, a ring bracelet etc. Then by placing the tag in front of the reader, the user's information can be read directly from the tag.

The PN532 is the most widely deployed NFC reader out there, being used in most smartphones. It can be communicated with via i2c, SPI or UART, although i2c is the default.

NFC is an extension of RFID, using the HF band. It typically has a maximum range of about 4 inches, which gives plenty of room for stuffing the sensor behind the project casing.

3.6.7 Nordic Semiconductor nRF52832

This board features a System on Chip, including a 32-bit ARM microcontroller with 32 GPIO pins and built-in NFC tag support, the last one being the reason for its inclusion here.

The chip itself is available from digikey for $5.73 each.

3.6.8 Express Input Scanner Summary

Table 11 below summarizes the various tradeoffs associated with each of the technologies for quickly inputting data, and their tradeoffs.

Table 11 – Express Input Scanner Summary

Product	Price per reader	Price per User	Technology	Connection
Magnetic strip card reader	$7.50	$0	MagStripe	USB/Serial
TTL Serial Camera Module + QR codes	$40	$0	Qr Code	UART Pictures, Video over NTSC
Zero Spy Camera for	$20	$0	QR code	CSI
Fingerprint scanner	$35	$0	Fingerprint Scanning.	UART
Standalone NFC reader	$30	$3	NFC	UART or RS232
nRF52832	$6	$3	NFC	Integrated in MCU

3.6.9 Part Selection

As can be seen from Table 11 in the previous section, the least expensive option is the Magstripe card reader. Although the nRF52832 costs less for installation it has a much cost per user. Since the system is being designed for garages which hold hundreds of cars, the card reader end up being significantly cheaper.

We purchased the component, shown in Figure 7 was purchased at SkyCraft, a local hardware store specializing in electronics in the Winter Park area of Orlando. The device has a simple serial interface, with separate wires for the separate tracks on a card.

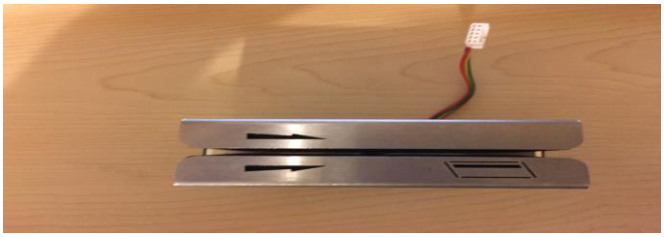


Figure 7 - MagStripe Card Reader With Serial Interface

3.7 Relevant Technologies

3.7.1 QR Code Formats

QR codes (or Quick Response Codes) is a type of barcode where the data is stored in a two-dimensional matrix format. The standard defines four encoding modes: numeric, alphanumeric, byte/binary mode and kanji. The last mode is a remnant of the fact that the standard was first designed for the Japanese automotive industry, in order track parts as they move through the factory and other parts of the supply chain.

The codes can be read quickly by even low end cameras, and have a higher information density than standard UPC barcodes, which are the ones you see on product packaging. The codes are processed with Reed-Solomon Error correcting codes (see that section for details).

A QR code consists of black squares on a white background. The code contains Three squares at the lower left, upper left and upper right corners which are used to detect the position of the tag. Six smaller mandatory squares placed at regular intervals in the code provide alignment information. Additionally, two contiguous lines of alternating black and white squares allow the sensor to figure out the exact boundaries between each data square.

3.7.2 Magnetic Stripe Cards and Magnetic Stripe Readers

Magnetic stripe cards were invented in the 1960s by an IBM engineer who had the idea of securing a piece of magnetic tape to a plastic card. He chose magnetic tape since it was the most widely used storage medium at the time. He became frustrated because every adhesive he tried produced unacceptable results.

The tape strip either warped or its characteristics were affected by the adhesive, rendering the tape strip unusable. After a frustrating day in the laboratory, trying to get the right adhesive, he came home with several pieces of magnetic tape and several plastic cards. As he walked in the door at home, his wife Dorothea was

ironing clothing. When he explained the source of his frustration: he couldn't get the tape to bond to the plastic in a way that would work, she suggested that he use the hot iron to melt the stripe on. He tried it and it worked. The heat of the iron was just high enough to bond the tape to the card. Most modern MagStripe cards have three tracks of data on them, each carrying distinct data.

There are generally two types of magnetic stripe cards out there today: those with High Coercivity and those with Low Coercivity. High Coercivity cards require significantly more magnetic energy to create, and therefore are more difficult to erase. Due to the energy requirements of the card writer, these cards are used in applications where they are not meant to be written to during their lifespan. Most credit cards and Identification cards fall into this category.

The other type of card typically found, those with Low Coercivity require significantly less energy to write. As a result, card writers for these cards are much cheaper, affordable to even hobbyists. the downside of these cards is that because they are easier to erase, their lifespan is much shorter.

3.8 System Housing Design

The housing of the components affects and is affected by, and affects, multiple aspects of the design. Care must be chosen for a material that will not interfere with wireless transmissions, if any are used. The housing must be large enough to hold all relevant components. Additionally, the housing should be resistant to tampering, to avoid people attaching card skimming devices or modifying the system to give them free parking. Additionally, the housing must be aesthetically pleasing.

Before designing the housing, itself, materials be evaluated. Since we do not know at this time all the components that will be inside the housing, we cannot determine its size or dimensions, so that discussion will be added at a later date. Specifically, we will discuss in this section the use of aluminum, plastic, and wood and their usefulness as a housing material

3.8.1 Aluminum

Aluminum is a very strong material, being used for applications such as building airplanes. It is not the easiest material to machine, however one of our group members does have some experience with it, and since UCF provides an extensive machine lab to students it seems to be feasible. Aluminum is electrically conductive, and is known to attenuate some RF signals, as a result care must be taken to ensure that the casing does not cause any short circuits in the electrical components.

3.8.2 Plastic

Plastic is rather expensive material, although it is usually used in small quantities. It comes in many different forms with different properties, but most seem to be equally viable for use in this project. Having access to a 3D printer as UCF students means that plastic can be made into arbitrary shapes on demand.

However, once the piece is created, it is difficult to trim or cut without cause splitting. Furthermore, the 3D printer creates shapes only roughly accurate to the design which does not look nice, so we should not use it for any externally visible components.

Plastic is not a conductive material and does not usually attenuate RF signals in any significant amount. Furthermore, any 3d shape can be made from plastic using the 3D printer UCF makes available to all students, however larger objects can be quite expensive, therefore this option might only be viable if a complex shape is required.

3.8.3 Wood/Lumber

Wood is the easiest material on this list, which is the primary reason for its inclusion here. Cutting wood and attaching it to other pieces of wood is something that can be done with tools owned by most people. Wood is a strong material, used to make houses, but is not as strong as plastic or aluminum are.

Furthermore, it not known to be a significant RF attenuator. Wood can be a fire hazard, especially around electrical components. This fire hazard is an important consideration, as microcontrollers typically do not include cooling fans, and we do not intend to add any to our design. Additionally, sparks which may result from shoddy wiring or power surges raise further concerns.

3.8.4 Pre-Made Housings

Since nobody on this team is a mechanical engineering student, or anything similar, it may be beneficial to consider casings that can be purchased off a store shelf. This section describes options which we are considering.

3.8.5 Project Boxes

So-called project boxes are available cheaply from many retailers, both online and in the storefront. They are constructed of plastic, aluminum, or some combination of each. The largest project boxes I could find available were 5.3” x 3.3” x 1.6”. Clearly these boxes are not very large, and can only be used if we are able to fit all components inside them, which seems likely.

These boxes can be obtained readily at a variety of department stores, including RadioShack, for $4. This poses the advantage of being readily available at any time during the project’s development

3.9 Internet Access

There are several different ways our smart parking system's microcontroller can be connected to the internet in order to communicate with our mobile App. The three best options for this would be either Wi-Fi, Ethernet, or Satellite (also involves Ethernet).

These three options are the most widely used and would be the most practical ways to enable internet connections to our device. The following sections explore these options in more detail.

3.9.1 Landline Internet (Ethernet)

Ethernet is an entire family of networking technology used for local access networks (LAN). It was commercially available in the 1980's and standardized by IEEE in 1983 under 802.3. Since then is has been improved upon to achieve higher bitrates and connect over longer distances.

Ethernet uses a coaxial cable as the shared medium of data transfer. Originally Ethernet allowed for only 2.96 megabits per second of data transfer, today there are speeds available up to 100 gigabits per second. Data transferred through Ethernet is dividing into what are called frames, which transfers the data in smaller sections. Each frame has the destination address, source address and some data for error checking.

Pros

- Fast data transfer rates
- Low chances of interference from outside sources
- IEEE enforced standards
- No extra hardware, only a wire and plugin

Cons

- Wire required (very long distance from Ethernet hookup from parking garages)

3.9.2 Satellite Internet

Satellite internet is available to consumers via communications satellites. These satellites are geostationary and provide internet speeds up to 50 Mbps (Megabits per second), and relay data from ground stations called gateways to a subscriber's antenna. This is done through radio waves in the microwave spectrum.

The main advantage of satellite internet is the removal of a landline to a user's location. Although, an ethernet cable connection is still required unless the internet is distributed from a Wi-Fi module. Using satellite internet for this project would avoid any need for a garage to have a landline internet connection to run this

device, as the majority of parking garages do not have ethernet connection available.

One downside is there are only two companies in the US that provide satellite internet, HughesNet and Exede Internet, and they only provide satellite internet. Meaning that a campus or business would need to pay for their current ISP's (Internet service provider) and also for one of these separate satellite internet providers.

Pros

- Shorter CAT5 cable needed
- Options for solar powered device

Cons

- Weather may knock out transmission
- Monthly subscription needed
- Small choice of internet providers
- Other internet provider needed other than that already used for the campus
- Slower data transfer rates than landline or Wi-Fi

3.9.3 Wi-Fi

Wi-Fi is a wireless data transfer technology which follows the IEEE 802.11 standards. Although IEEE does not test any Wi-Fi device for their standards, a non-for profit organization called the Wi-Fi alliance was made to police the standards in order to help promote WLAN (wireless local access network) technology. Numerous common electronic devices today use Wi-Fi technology, such as cell phones and personal computers.

These Wi-Fi enabled devices connect to the internet via a WLAN access. Coverage of the "hotspot" varies depending on the output power of the Wi-Fi router(s) and if there are multiple access points. Wi-Fi bands are close to the frequency used in microwaves and Bluetooth devices, so they may suffer interference from said devices. If this so happens the Wi-Fi device will switch channels to achieve lower interference.

There are four main variations of 802.11 Wi-Fi; a, b, g, and n. Each have their advantages and disadvantages. "a" operates at 5GHz and has the fastest maximum speeds up to 54 Mbps (megabits per second) but has the highest cost and lowest range of signal that is easily obstructed by walls. "b" has the lowest cost and highest signal range that is not easily obstructed since it has a lower frequency at 2.4 GHz, but has the slowest maximum speed up to 11 Mbps. "g" combine the best parts of "a" and "b" by operating at 2.4 GHz like a for greater range and less obstruct able signal with a high data transfer rate of 54 Mbps like

“b”. It is also much more expensive than the type “b”. Type “n”, also called wireless N, was created to improve on the type “g” variation. It allows up to 300 Mbps speeds and great signal range at 2.5 GHz, but is very expensive and the standard hasn’t been finalized yet.

Pros

- Easy to implement (no cables/wiring from device to device)
- Minimal interference (array of channels available)
- IEEE standards implemented and enforced
- Variety of versions
- Ranges up to 600 feet
- Transfer speeds up to 300 Mbps

Cons

- Security risks from wireless hacking
- Speeds degrade with interference of Bluetooth devices, microwaves, and obstructions
- Still requires ethernet cable to be run to parking garage for the Wi-Fi router

Table 12 - Internet Access Comparisons

Internet type	Transmission speed	Cost	Security risk	Hardware cost
Landline	Up to 100 Gigabit/s	~$0	Low	$12
Wi-Fi	Up to 300 Megabit/s	$9	High	$9
Satellite	Up to 50 Megabit/s	$100/month	Moderate	<$5

3.10 Microcontroller Device-to-Device Communications

If multiple microcontrollers are used for this project, such as two for the ingoing and outgoing sensors of the garage and one as the main hub for direct communication and updating of the parking garage phone app, there are a number of different technologies that can be used so the microcontrollers can talk to one another.

Some options are Bluetooth, Wi-Fi ad hoc, UART (Universal asynchronous receiver/transmitter), I2C (Inner-integrated circuit), SPI (Serial Peripheral Interface), GPIO (general purpose input/output), and USB (Universal Serial Bus). The following pages will explore these options more in depth.

3.10.1 Bluetooth

Bluetooth is like Wi-Fi in that it uses radio waves to communicate wirelessly. Communication of two Bluetooth devices is over a short range with a maximum of 100 meters, and so isn't well suited for long distances. Bluetooth devices need to first pair with one another before they can begin communicating. Bluetooth works as a master and slave communication system. Such as, if there are multiple devices paired like a computer paired to a Bluetooth keyboard and a Bluetooth mouse, then the computer takes the role as master and the other as slaves.

Recent Bluetooth sensors have become very power efficient and are able to run off a coin sized battery for months. Technology such as Bluetooth Smart, BLE, and GATT (Generic Attributes) allow for Bluetooth sensors which require a very low amount of energy. Making them easily feasible to run off of purely solar or even kinetic energy.

The ISM bands of Bluetooth are from 2.4 to 2.485 GHz (similar to Wi-Fi). The technology was invented by Ericsson in 1994 as a wireless alternative to RS-232 data cables which was the standard of serial communication at the time. The standards for Bluetooth are controlled by the Bluetooth Special Interest Group (SIG), because IEEE no longer maintains their standard of 802.15.1 for Bluetooth technologies.

There are four different classes of Bluetooth. Class 1 Bluetooth uses the most power at 100mW and has the furthest range up to 100 meters. Class 4 uses the least power at only 0.5 mW with a range of only about half a meter. Most devices today such as cell phones use class 2 which uses 2.5 mW of power for a range of up to 10 meters. If Bluetooth is used in this project for wireless communication between two microcontrollers, class 1 or 2 will be the most likely choice for this application. Although class 1 requires 100mW of power and class 2 only has a range up to about 10 meters.

Bluetooth also uses a UART connection to allow data transfer and receival, shown in figure 8. Using a total of 4 pins on a MCU.

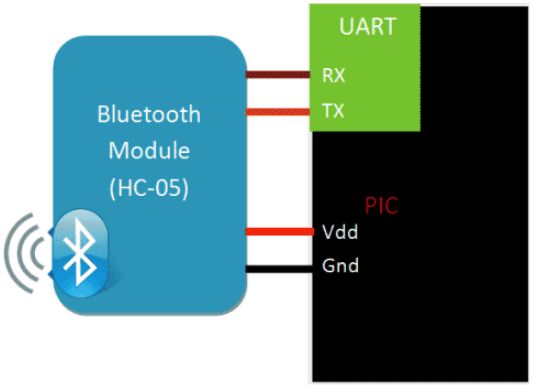


Figure 8 – Bluetooth Module with UART

Pros

- Low power consumption
- Small sensors
- Direct device to device communication
- Cheap compared to Wi-Fi ad hoc technology
- Uses UART to communicate
- Only 4 pins used

Cons

- Interference of other Bluetooth devices
- Considerably short ranges
- Interference from Wi-Fi's in the same ISM
- Intended for indoor use (our project is outdoors)
- Easily obstructed signal by physical objects

3.10.2 Wi-Fi Ad hoc

Wi-Fi also has the capability to allow two devices to talk directly to one another without the need of traveling through the internet, this is called Wi-Fi ad hoc. This has been used in a number of consumer electronics, typically handheld gaming systems.

This technology would be of use if multiple microcontrollers are used for this project, to allow them to remotely talk to one another without the use of direct wiring. An ad hoc network is considered a decentralized network because of this direct device-to-device communication, so no routers or access points are required. Wi-Fi ad hoc can use many types of radio frequencies, from UHF (Ultra high frequency) 300-3000 MHz to SHF (Super high frequency) 3-30 MHZ. Ad hoc uses unlicensed ISM (industrial, science, and medical) 2.4 GHZ radio frequencies, and also 5.8 GHZ radio frequencies.

Pros

- No expensive equipment needed for communication (routers and access points)
- Operates in unlicensed frequency spectrum
- Fast direct distribution of information
- Easy setup
- Uses UART for communication
- Longer distance of connection than Bluetooth

Cons

- Expensive compared to Bluetooth
- No central hubs

3.10.3 UART (Universal Asynchronous Receiver/Transmitter)

UART is a device for serial communication with variable data formatting and transmission speeds, controlled by the user. It takes data that is parallel and translates it to and from serial form for transmission. This type of device is very common in microcontrollers and is typically included in them.

The UART takes byte of data and transmits them bit by bit in a serial manner from one device to another. This serial transmission through one wire is more cost effective than parallel transmission through multiple wires. Communication with UART can be either simplex, meaning in one direction only, full duplex; both devices can transmit and receive data at the same time, or half duplex; the devices take turns transmitting and receiving.

Advanced UART technologies are capable of holding received data in a buffer until the microcontroller comes to read it. Most UARTs release the buffered data in a first in first out basis (FIFO). These buffers can be as small as bit and as large as kilobytes.

The majority of microcontrollers offered today have at least one UART onboard, and may carry as much as four UART connections in larger microcontrollers. It's intended use for this project would be to allow hard wired communication between two microcontrollers, but it has many other possible uses too. Such as GPS modules, modems, Bluetooth, wireless communications, and many more. Applying this method to microcontrollers would involve hard wiring their RX (receiver) and TX (transmitter) pins together for a simple hard-wired communication.

Pros

- At least included in just about every microcontroller
- Simple to program
- Simple to set up hardware
- Options to have many UARTs on one microcontroller for flexibility

Cons

- Hard wired UART communication may have very long wiring from microcontroller to microcontroller when implemented (>10 feet)
- Many wired connections may be required in order to have successful handshaking

3.10.4 I2C (Inner-Integrated Circuit)

I2C is a serial protocol with only two wires used for an interface to connect low speed devices. Devices such as microcontrollers, I/O interfaces, EEPROMs, and other likewise peripherals inside embedded systems.

The original design was created by Philips and is used just about all major integrated circuit manufacturers. I2C is very popular due to its simple use. It uses

master-slave functionality where there can be multiple masters controlling many slaves. Each slave has its own address and can be called on by a master to either read or write, and the master device(s) require no address since they generate the clocks.

The interface uses only two wires, one for the serial clock and the other for the serial data, both require a pull up resistor to +Vdd. In order to start communication, the master with generate the start condition followed by the address of the slave it wants to use. If the 0 bit of the address was a 0 then the master wants to write to the slave device and if it's a 1 then it wants to write.

Once all the data has been read or written the master will send the stop condition to signal the communication has ended. I2C is capable of having virtually unlimited number of slave devices connected in this configuration and it is simple to do so, the only requirement is that there are enough bits available in order to have unique address for each slave.

Pros

- Simple to implement
- I2C communication can be used on a MCU even if there is no I2C interface on board
- Large number of possible slave devices
- Only two wired connections needed for communication
- Multi-master capable

Cons

- Slow data transfer rates
- Data transfers are only controlled by master devices

3.10.5 SPI (Serial Peripheral Interface)

SPI is a common interface bus that is used to transmit data between microcontrollers and other similar technology. It is typically used for short distance communication in embedded systems. Originally created by Motorola in the late 1980s and became a standard form of communication thereafter.

Communication happens in full duplex with a single master and a number of slaves. If there are many slave devices, they are supported by having many slave selection lines (SS lines). There are four connections used for SPI.

The first of which is SCLK (serial clock), which is controlled by the master device. MOSI (master output slave input) and MISO (master input slave output) are on way data transfer lines for reading or writing the data to the from the slave devices. SS (slave select) as mentioned before is used by the master device to control which slave it wants to operate. To start communications the master configures its clock to a supported frequency of the desired slave device to be used. The master then selects the slave device begins it operation. During each cycle a full duplex

of data is transferred. Data transmission typically consists of 8-bit words, but other sizes may be supported such as 12 or 16 bit words.

Pros

- Full duplex data transmission is default
- High throughput of info compared to I2C
- Very simple hardware interfacing
- Only four pins required for use
- Simple software implementation

Cons

- Requires more pins than I2C
- Only supports one master device
- Having many slave devices make SPI impractical compared to I2C

3.10.6 GPIO (General Purpose Input/output)

Another option for cross-microcontroller communications is use configure their given GPIO pins to communicate in either 8-bit or 16-bit format. GPIO pins are generic pins on integrated circuits and computer boards with no intended initial purpose and in most cases, go unused by default. The intent is that a device integrator/designer who is building some sort of system may have these pins as an option to use instead of having to go through the trouble of adding extra hardware to fulfill a tasking requiring more inputs or outputs. GPIO pins have many capabilities, such as normal input or outputs to control an LED or a simple beeping speaker.

They also can have input values that are readable, and output values that are writeable and readable. The intent would be to use them for communication between two microcontrollers in a sort of parallel custom made UART by attaching up to 20 GPIO pins on each microcontroller. 16 of the pins will be the 16-bit information, one pin for the clock, another for starting the transmission, another pin to say if it's reading or writing, and the final pin to indicated the stop of the transmission.

This will allow for parallel communication of data transfer in 16 bit chunks. The data transmission only need to be unidirectional, as the master microcontroller only need data from the microcontroller controlling the motion sensor and there is no reason for the sensor's microcontroller to receive data from the master microcontroller.

Pros

- Perceived higher bit rate of data transfer with 8 or 16 bits of parallel data transfer as opposed to serial communication of UART or I2C

Cons

- More time dedication needed due to building from the ground up rather than using built in UART or I2C features

- Cost more than UART or I2C because of the larger amount of wires needed to connect 12 or 20 pins
- At most 20x50ft of wire needed which is about $60 of wire
- Greater risk for user error on hardware and software side

3.10.7 USB (Universal Serial Bus)

USB is a standard that dictates the connections and communication protocols between computers and other devices. It has become the most common type of connector and has replaced many older connectors such as, barrel connectors and serial ports. It was developed in the 1990's by the USB implementers Forum (USB IF), a non-for-profit organization created to promote and support USB. Its members include many large name semiconductor and computer companies including, Microsoft, Apple Inc., Intel, and HP.

There are four different levels of USB's, each one is an improvement on that latter version of a USB. USB 1.1 was released in 1998 which allowed for transfer speeds of 12 Megabits per second. USB 2.0 was then released in the year 2000 that allowed for far greater transfer speeds of 480 megabits per second. Eight years later USB 3.0 is released allowing for an improved 5 gigabits per second of data transfer, it's also known as USB 3.1 generation 1.

Five years later USB 3.1 is released in July 2013, also called USB 3.1 generation 2, allowing for data transfer rated up to 10 gigabits per second. Because USB has been around for nearly twenty-five years, most microcontrollers have software already offered for them to use USB communication easily, such as the Microchip PIC microcontrollers. This will be a great help for quick programming and debugging, as there are also forums and FAQ sites giving a hand in implementing this code correctly and efficiently.

Pros

- Widely used and supported
- Allows for huge amount of data transfer, up to 10 Gbit/s
- Very cheap, some cables costing only $15 a 50 feet male to female USB cable
- Easy to find and pre-established support for its use

Cons

- Typically, only USB 2.0 is supported in microcontrollers

Name	Cost ($)	USB	UART	#Pins	Self Programing	Ethernet	Debugger cost ($)
Atmel	210	Y	5	144	Y	Y	N/A
Rasp. pi 3 B	35	Y	0	40	Y	Y	N/A
nRF52832	6	N	1	48	N	N	33
PIC32MZ17	15	Y	6	176	N	Y	50
PIC32MX64	2.16	Y	4	64	N	N	50
PIC32MX51	5.66	Y	6	64	N	Y	50
Rasp. Pi A+	20	Y	N	40	Y	N	N/A
MSP430F67	5	N	3	72	Y	N	90

Table 13 – Device to Device Communications Summary

3.11 Microcontroller

For this project, there are a large variety of microcontrollers which can be chosen to get the task done. Not only by capability but also by added features. It's simple to choose a MCU which can handle all the tasks thrown at it, but it's harder to find one that can handle everything required while also having things such as low power modes, or a built in wifi module. Although many options can be added on externally, many features can only be found inside an offered MCU.

3.11.1 Main Microcontroller

Our group is still exploring many options to use just one powerful microcontroller to do all the processing and data transfer. Which will involve reading all the sensors detecting cars leaving and entering the garages, relaying data through the internet to the App, and controlling the parking kiosks for manual user entry of data.

This of course will require a very powerful and flexible microcontroller with room for a lot of interfaces, whether that be USB, Ethernet, GPIO, or HDMI. There are many microcontrollers on the market which could handle such a task, but at a hefty price. Below I will explore the pros and cons of a few chosen microcontrollers which may be up to the task.

The other option is to have two MCUs in total, where one controls the car sensors at the entrance and exit of the garages and then relays that data to the main MCU which will control the kiosk and also relay all the needed data to the mobile App.

3.11.2 Atmel SAM V71 Xplained

This microcontroller uses an ARM Cortex-M7 microcontroller, the most powerful version of this microcontroller comes with 2MB flash, QSPI, 384kB of SRAM, 10/100Mbps Ethernet with traffic shaping, dual CAN-FD, SD/MMC, high-speed USB host and device, CMOS camera interface, 1 SSC, 3 USARTs/SPI, 2 SPI, 3 TWI, 5 UARTs, low-power backup with 1KB SRAM, external bus interface, SDRAM controller, and 144 pins.

The powerful capability of this evaluation board microcontroller is reflected in its high price of $210. This microcontroller isn't a standalone chip, but is instead a full evaluation board with many of the connectors already attached to it, such as the power and Ethernet cable slots. This allow for quick and easy installment.

One problem is its lack of USB connectors. It only has two micro USB ports and those already have dedicated uses for debugging and for controlling a target device. Another downside is that the board contains a few different types of connectors and capabilities that are unnecessary and will go unused in this project. Such as a separate microphone and headphone jack, and a media LB connector.

Our project will not use any audio or any audio recording and neither a Media Local Bus connector, rendering those extra and unused. But the extension 4 LCD connector will most likely be used for the garage kiosk, for a simple display of inputs and outputs to the user.

If this board is chosen to be used its many UART connections (5 of them) will have to be used in order to receive data from the car sensors at the entrances and exits of the garage, and its LCD connector for display at the kiosk. The Ethernet port is also the only option to connect the microcontroller to the internet as there is no option for a USB connected Wi-Fi dongle.

If separate microcontrollers are used to control the entrance and exit car sensors, then they too will have to be communicated with using the built in UART connections on the Atmel evaluation board since there is no option for Bluetooth or Wi-Fi. Another option is to use its many GPIO pins for communication, but that is very unlikely as UART in most cases is superior to custom made GPIO communication, as well as cheaper.

Pros

- Very powerful ARM microcontroller
- Many UART connections (5 total)
- Option to add micro SD card for more storage space
- Many pins allowing for flexibility (144 pins, 114 GPIO)
- Ethernet cable slot onboard
- Self-programming (no external debugger needed to program)

Cons

- No HDMI ports or support
- No USB ports

- No options for wireless connectivity
- Very expensive when compared to other powerful microcontroller evaluation kits
- Extra ports that go unused

3.11.3 Raspberry Pi 3 B

This version of the Raspberry Pi is the latest release with the most capabilities. The microcontroller uses a 1.2 GHz 64bit Broadcom ARM v8 CPU, which is powerful enough to run retro videogames from the 90's at 1080p. It also has 1GB of ram, built in 802.11n Wi-Fi, built in Bluetooth 4.1, 40 GPIO pins, 4 USB 2.0 ports, stereo output, composite video port, HDMI port, Raspberry Pi camera port, Raspberry Pi touchscreen display port, Micro SD slot, and a micro USB port for power supply.

This microcontroller is immensely powerful, and can easily function as a low powered computer. It also comes at a very low cost of $36. This type of microcontroller is a great candidate to run a program called Node.js, which is a CPU intensive program that allows for the execution of JavaScript code server-side and allows for our microcontroller to easily communicated with, and update the parking App.

One downside to the board is the built in Wi-Fi 802.11n, because developers also took out the Ethernet port to make room for this module, so the only option for internet connectivity is to use its Wi-Fi capabilities coupled with a nearly Wi-Fi router. That removes the option to use purely ethernet connected internet, or to use satellite internet using and ethernet cable to the satellite dish. Another minor problem with the Raspberry Pi 3 B is there are a lack of GPIO pins as compared to the previously mentioned Amtel SAM Xplained which has+ 112 GPIO pins, the R-pi only has 40.

This restricts the flexibility of its capability to support many other I/O devices, like an LCD screen, keypad, and motion sensors. Also, the two UART capabilities of the Raspberry PI 3 B are already pre-connected to its Wi-Fi and Bluetooth modules. That leave no available UART connections for use by the user, a big downside as UART as far as this group is concerned is the best type of data transfer to be used in this project for the motion sensors.

Pros

- Cheap compared to many other options, ~$35
- Powerful ARM processor for the price
- Self-programming (no external debugger needed)
- Able to run programs compatible with Linux machines (like Node.js)
- Built in Wi-Fi
- Built in Bluetooth
- HDMI port

Cons

- No Ethernet port
- No free to use UART connections
- Low GPIO pin count (only 40)
- Isn’t a true microcontroller and is more like a weak computer

3.11.4 nRF52832 Microcontroller

The nRF52832 microcontroller is a powerful and very low power machine with built in Bluetooth. It has a 32-bit ARM Cortex-M4F CPU with 64 kB of RAM, 512 kB of flash, 3 SPI, 2 two-wire interface I2C, UART, 2 PWM, 48 GPIO pins, and a digital microphone interface.

The device uses Bluetooth 5, the latest version release of Bluetooth and is a great improvement from Bluetooth 4.0. Bluetooth 5 has higher data throughput, at about 1600 kbps (kilobits per second) it is nearly five times faster than Bluetooth 4.0. It also has greater range at about four times the range of Bluetooth 4.0 capabilities reaching up to 400 meters, whereas 4.0 has a maximum range of 100 meters.

The device in itself is very low power, requiring only a voltage source between 1.7 and 3.6 volts to function correctly. Most other microcontrollers recommend using a source of at least 5 volts, so the nRF82832 can run at less than two fifths the voltage of other microcontrollers.

The chip doesn’t have any capabilities to connect to in internet on its own, which will be a complicated problem to solve as the only option will be to use a separate Wi-Fi or Ethernet dongle that will have to be connected to the MCU through its available GPIO pins.

This chip is also a stand-alone chip and not a evaluation/development board, meaning that it doesn’t have self-programming capabilities so a development kit is needed to program then single microcontroller using spy-by-wire.

The kit cost about $33 and the microcontroller costs about $6. Costing about $4 more than the Raspberry Pi 3 B, with less capabilities but containing Bluetooth 5.

Pros

- Very low power usage
- Built in Bluetooth 5

Cons

- Development kit needed to program ($33)
- No built-in internet capabilities
- Less processing power and costs more than Raspberry Pi 3 B
- Only one UART connection (at least 2 wanted for this project)

3.11.5 Microchip PIC Microcontroller

The Microchip PIC Microcontrollers offer an enormous library of hundreds of different MCUs with different capabilities and applications. All of their offered microcontrollers are stand-alone chips, and are not built into an evaluation/development board. Because of this they require an external debugger to be connected to it in order for the user to program it.

There are a few different debuggers offered in order to program the PIC MCUs, the one that would be chosen for this projects application would be the PICkit 3 In-circuit Debugger. The PICkit 3 costs about $48 and $60 after shipping. It has all the basic functionality needed in order to program, reprogram, and debug a PIC MCU. It works by plugging into a program designer's PC using a USB interface, then attaches to the PIC MCU with two I/O pins and a reset line. It's then run by the MPLAB X Integrated Development Environment (IDE). The PICkit 3 is the most basic debugger offer for Microchip PIC MCUs, the others offered have added functionality that would go unused, they are also much more expensive at 200+ dollars.

One of the most powerful PIC MCUs offered from microchip in the PIC32MZ2064DAH176 and is only $15. It's a 32-bit MCU with a 200 MHz MIPS32 core, 2MB of flash memory, 640kB SRAM, 32MB DDR2 DRAM, integrated graphics controller, graphics processor, Ethernet MAC interface, Hi-Speed USB host device, Crypto Engine for data encryption, 6 SPI modules, 5 I2C modules, 120 GPIO pins, and 6 UART modules.

The MCU has many other capabilities, but the ones mentioned above are the highlighted most likely to be needed by this project. The many interfaces and ports of this device give it a lot of fixability for large variety different hardware arrangements. But its lack of processing power and flash memory leaves it unable to run the planned Node.js program in order for our devices to talk to and update the phone App.

Although, with its many UART connections (6 of them) this MCU is a great candidate if it is decided that multiple microprocessors will be used for this project. This one would be good at reading the relaying the data collected from the sensors detecting cars coming and going from the garages, while another MCU being the "central MCU" which will collect and relay the data through the internet to the App.

There will possibly be another controlling the Kiosks that will be placed in the garages also, and it too will compute and relay its data to the central MCU. The PIC32MZ2064DAH176 will make a possible choice for the kiosk's MCU as it has an integrated graphics controller and a graphics processor.

Pros

- 6 UART connections
- 120 GPIO pins (144 pins total)
- Ethernet connection

- Graphics controller and processor
- $15
- USB host device
- 32-bit code
- A lot of DRAM, 32MB

Cons

- $48 - $60 debugger required
- Only a 200MHz core
- Lack of Flash memory, only 2MB
- External power supply needed (it's a stand-alone chip)

Another PIC microcontroller that is more specialized for the application of controlling the two sensors at the entrance and exit of the garage is the Microchip PIC32MX120F064H. It is a 32-bit MIPS MCU that runs at 50MHz, with 64kB of flash memory, 8kB of SRAM, two I2C modules, 4 SPI modules, Low-power management modes, 4 UART modules, 53 GPIO pins, and runs at 2.3 to 3.6 volts. The chip on its own only costs $2.16, but because it too is a PIC MCU it requires the $60 debugger hardware; which brings the cost up to about $62 for this MCU.

This version of PIC MCU is a scaled down version of the previously mentioned PIC32MZ2064DAH176. It has a lot less peripheral support, but the only modules needed for this task are to have at least 3 UART modules to connect it to two sensors and then the main MCU, and this device has 4.

There weren't any with exactly 3 UARTs that are in the PIC32MX MCU category, which are the simple bare bones MCUs offered from Microchip. They either offered 2 or 4 UART connections, and this was the only PIC MCU with 4.

This MCU also has a selection of different low-power modes; deep sleep, sleep, and Idle. Those will be in use during low traffic hours in the garage's operation and during the nights. Although the MCU's core only runs at 50MHz, the MCU will only be used to relay data from the sensors to the main MCU using UART connections, and the low clock rate MCU is good for saving energy.

Pros

- No extra peripheral connections wasted
- Low-power modes
- 4 UART modules
- $2.16 for the chip
- 32-bit code
- 53 GPIO pins

Cons

- $60 debugger required
- One extra UART connection
- Only a 50MHz core

The PIC32MX675F51H microprocessor offered by Microchip is another great candidate. Although is only have a max core speed of 80MHz, it should be just enough power to run every device we wish to have for this project. This MCU has a memory of 512 kB, 128 kB RAM, 12 Kb of auxiliary flash, operating voltage of 2.3 - 3.6 volts, 3 SPI modules, 3 I2C modules, 6 UART modules, built in USB host controller, 64 pins with 53 of them being GPIO pins, and a built in Ethernet controller. This PIC32 microcontroller with be easier to handle than the PIC32MZ2064DAH176 mentioned before because it only has 64 pins with about the same form factor rather than 144 pins. Meaning that each of the pins are spaced out a lot more, allowing for less precise soldering to be used.

One downside they both share is that their pins are flat form factored, so they don't insert into a breadboard on their own. A breakout board will have to be made in order to allow for easy prototyping and testing.

There is only one breakout board offered online that would fit this form factor which is only $3, but has terrible reviews of one out of five stars because of its apparent poor quality. If this breakout board doesn't work, then one will have to be made. two important qualities the PIC32MX675F51H microcontroller has is the built in USB and Ethernet host modules.

The USB is needed to run a card reader that will be housed in the kiosk, and the Ethernet is needed to connect the MCU to the internet to talk to the App that will be created in unison with the hardware in the garage. The 53 GPIO pins will be plenty to run the LCD screen and keypad that will also be used in the kiosk to take and display user input. Both the keypad and LCD screen will need about 20 GPIO pins together.

Pros

- 6 UART Modules
- On-board USB host controller
- On-board Ethernet controller
- Larger pin spacing
- Only $5.66

Cons

- Low profile pins (breakout board needed)
- 80MHz may be too slow to run all peripherals effectively
- Harvard Architecture
- $60 Debugger needed to program the MCU

3.11.6 Raspberry Pi Model A+

This version of the Raspberry Pi is the low cost and smaller form factor variant of the Raspberry Pi products. It only costs $20 and is self-programmable so no external debugger is needed.

The Model A+ has 40 GPIO pins, a micro SD slot, 1 USB port, a mini HDMI port, Raspberry PI camera connector, Raspberry PI LCD connector, audio jack, and 256MB of RAM. The Model A+ lacks any sort of Wi-Fi or Ethernet connection port, so in its basic state is unsuited for use as the kiosk MCU. Also, because it doesn't have any UART ports either, it's also unfit for use as the entrance/exit MCU. Because it does have the Raspberry PI LCD connector, it could be used as a kiosk MCU even though it has not internet capabilities.

In that case there would then be a third MCU in use that would receive all the data from the two other MCUs and send it over the internet to update the mobile App. The biggest attraction to the Model A+ is it's lower power consumption and improved audio quality compared to other Raspberry Pi's. Although, audio capabilities aren't needed for this project.

Pros

- $20
- Raspberry Pi LCD connector
- HDMI

USB port

Cons

- No internet capabilities
- No UART ports
- Project would need 3 MCUs total if this one was to be implemented

3.11.7 Texas Instruments MSP430F6720

The MSP430 family of MCUs offered from TI are a large variety of ultra-low-power microcontrollers that feature many different sets of peripherals for use in specific applications. The MSP430F6720 is the best fit of the MSP430 family because it has minimal extra features and 3 UART connections. This MCU would be used as the MCU controlling the two sensors at the entrance and exit of the garages with two of its UART connections, and the last UART connection to send the sensor's data of cars entering and exiting to the central MCU located inside the kiosk which will then relay that data through its internet connection to the mobile App. There are many variants for the MSP430F6720, numbering from MSP430F67(*20)* to MSP430F67(*36)*. The sixteen different variants only differ in their combinations of non-volatile Memory sizes and Ram sizes. Ranging from 16 to 128 kB of non-volatile Memory, and 1 to 8 kB of RAM. Other than their differences in memory and RAM, all other features of each version are identical.

Along with its 3 UART connections the MSP430F6720 contains a 25MHz 16-bit MIPS core, with 72 GPIO pins, 1 I2C module, 4 SPI modules, and 3 DMA. The chip only requires a power source of at least 1.8 volts to a max of 3.6 volts to run effectively. It also has many types of sleep/low-power modes which will be needed during the non-busy hours of operation in the garage. Each sleep mode turns off different clocks and has different parameters for waking up out of the low-power

mode. Such as LMP1 will turn off the DCO and can be woken up by internal or external interrupts.

One problem with the MSP430F6720 is that it requires a 100-pin Target Development Board offered from TI specifically for the MSP430F6xxx MCUs in order to program and debug the hardware. This Development board MSP-TS430PZ100B costs about $90, while the chip by itself is only $5. Bringing the total price of this MCU up to 95$ for this project.

Pros

- 3 UART connections
- Only $5 for the MCU
- Entire group has experience using MSP430s from previous classes
- Ultra-low power consumption

Variety of Memory/RAM combinations offered

Cons

- $90 Development Board required to program the MCU

Table 14 – Microcontroller Comparisons

Name	Cost ($)	USB	UART	#Pins	Self Programing	Ethernet	Debugger cost ($)
Atmel	210	Y	5	144	Y	Y	N/A
Rasp. pi 3 B	35	Y	0	40	Y	Y	N/A
nRF52832	6	N	1	48	N	N	33
PIC32MZ17	15	Y	6	176	N	Y	50
PIC32MX64	2.16	Y	4	64	N	N	50
PIC32MX51	5.66	Y	6	64	N	Y	50
Rasp. Pi A+	20	Y	N	40	Y	N	N/A
MSP430F67	5	N	3	72	Y	N	90

3.12 PCB Circuit Design Program

There are a large number of programs offered for circuit designs. These programs can be used for testing circuit virtually to quickly analyze their stability and function. They can also be used to create gerber files, which are the files used to save diagrams of a circuit that will be printed on a PCB (printed circuit board).

Most programs that are offered for circuit design are not entirely free but offer limited free trials that allow for the the program to be used with the majority of it's tools and capabilities unlocked for use. Below are in depth descriptions of these circuit design and analysis programs.

3.12.1 PCBWeb

This program is a free to use PCB design tool offered by a PCB printing company. Because this program is offered directly from a company that prints the PCBs too, that means there is more security in designing a PCB with this program because you know their PCB printers are capable of carrying out all the functionality of their program. It has a clean interface that isn't too crowded with tool around the editing screen.

The program has a default component selection tab that has access to DigiKey's inventory, this is good for keeping it so that you only use components that are available. Although as read a many user reviews, the program is hard to work with and doesn't have a large library of preset part sizes. But, the custom component creating tool is very simple and effective to use. A disadvantage to the tool's simplicity is that many advanced options aren't available, such as editing preset offered components to fit your needs.

There also aren't any options to import or export any components. It also doesn't allow for the user to see the schematic and PCB layout at the same time. So if this is needed the user will have to print out the schematic layout to use as reference while editing the PCB layout. Although, the program has an auto generated bill of materials (BOM) that keeps track of all the components you use in an easy to read list that also totals the prices of every component.

Pros

- Easy to use
- Free to use
- Component database connected directly to DigiKey
- Automatic bill of materials

Cons

- Limited advanced editing
- No importing or exporting components

3.12.2 DesignSpark PCB

This PCB program has been around for a long time and thus is very stable as it has gone through many updates and bug fixes over the years it has been active. It's owned by the Electrocomponents PLC which also owns Allied Electronics and RS Components, the joint father companies of this program and community.

This program also doesn't have very many limitation on the sizes and the number of the components. The program uses standard industry gerber files to save the PCB layouts, so you can have any PCB manufacturer print out your PCB. Unlike PCBWeb, which forces you to use their services to print out your PCB because they don't allow you to export your design as a gerber file. DesignSpark PCB is also completely free. Although DesignSpark PCB allows for no options for simulations, that won't be necessary for this project anyways.

Pros

- Free to use
- Professional grade
- Very stable program
- Standard gerber files used

Cons

- Extensive login information needed each time program is opened

3.12.3 PROTEL (Altium Designer)

PROTEL is a CAD circuit design program that has been around since the 80s and many universities today use it as it's go-to program for circuit design and even offer classes centered around using the program.

The user interface and program is best suited for creating boards with less than 4 layers, but has been proven to become very complicated if more than 4 layers are needed. The program's price point is intended for large companies at its $7000 license price point. Making it nearly unobtainable for this project.

There is a short free trail offered for 15 days, but that would make the PCB design too time constrained. It also has proven to have a difficult user interface for new users, and has a steep learning curve.

Pros

- Time proven to be stable and well updated
- Used by many universities

Cons

- Short 15-day free trial
- $7000 license fee
- Unfriendly user interface
- Not good for boards greater than 4 layers

3.12.4 CadSoft EAGLE PCB Design Software

EAGLE PCB is a subscription only base PCB design software. The pricing is either $65 a month or $500 a year which will save you $280 in the long run. There is a lot of support offered when using this program, such as the Wurth Electronics company provides extensive libraries for Eagle and their parts that can be used in designed PCBs. This option helps a user make less guesses about their design flow and it reduces the iteration of their design. It also is compatible with MAC operating systems. That's a convenient feature to include, as one of you group mates has only a MacBook.

Creating new components is streamlined and made easy in this program, and is well suited for beginners. There's also a lot of support from many forums throughout the internet to help you figure out any task you would need to do in the program. The design blocks in the program allow for widely distributed workflows with a real-time sync option, but the real-time sync is buggy for large teams of people working on a project at the same time.

Pros

- Online real-time team editing of a single program
- Monthly subscription is cheaper than other PCB designer tools
- Large support base online

Used by many large companies

Cons

- Monthly subscription can cause this program to be more expensive than other PCB programs

3.12.5 Kicad

Kicad is an open-source software PCB development program abiding by GNU's GPL copywrite agreement. It has a GUI (graphical user interface) with the ability to be run on Windows or Linux. The software is completely free to use with no limitations.

A problem with Kicad is the fact that it's not owned by any one company, meaning that there is not support page offered for this program. If a user comes across any problem, then they have to turn to online searches and forums in order to solve their problem. Other users are the only support a designer has in this PCB program.

Based off of online reviews from past users of Kicad, the program is very hard to use for inexperienced users. Kicad at certain points will display and error message if something is wrong when a user try's to save the program, then Kicad will freeze and delete all the saved progress.

Pros

- Completely free to use
- Graphical user interface
- Open-source

Cons

- No official support
- Not new user friendly
- Doesn’t run on MAC OS

3.12.6 Osmond PCB

This printed circuit board design program run only on Macintosh computers, and is not supported by Windows or Linux devices. Making it only a candidate if our group mate that has an apple computer wishes to do all of the design their self. Other than that one major flaw, the PCD program is very well designed and allow for many features.

Osmund lets the user create virtually unlimited board sizes with unlimited layers and parts. It also supports hole and surface mount parts. It gives options for both imperial and metric units, with no penalty for using either. There’s also the possibility of using both types of units within the same PCB layout.

Built within the program is the Osmond editor tool, allowing for the user guided ability to create new parts from scratch. It also lets the user edit existing parts for their own design. This comes in handy if a developer would need to create/edit a part to fit a mechanical constraint within their project. It allows the user to create these part types in many typically unsupported shapes such as, round, and oval pad shapes. Osmund also supports the ability to look at two layers simultaneously as silkscreen layers, with two solder mask layers, and two auxiliary layers.

Pros

- Built in custom design tool
- Free to use
- No limitation constraints in designs
- Imperial and metric units supported

Cons

- Only supports Mac OS
- Only free to use for small designs

Software	Free trial limit	Price	Ease of use	Uses Gerber Files	Online group editing
PCBWeb	No limit	Free	Moderate	No	No
DesignSpark	No limit	Free	Easy	Yes	No
PROTEL	No trial	$7000	Hard	Yes	Yes
EAGLE	15 days	$65/month	Easy	Yes	Yes
Kicad	No limit	Free	Hard	Yes	No
Osmund	No limit	Free	Moderate	Yes	No

Table 15 – PCB Comparison Summary

3.13 Ethernet PHY

The PIC microcontrollers that have been chosen for this project do not have built in ethernet PHY. Mean that an external Ethernet PHY chip will be needed to have the Ethernet to work properly. Figure 9 below shows the setup with an external ethernet PHY chip.

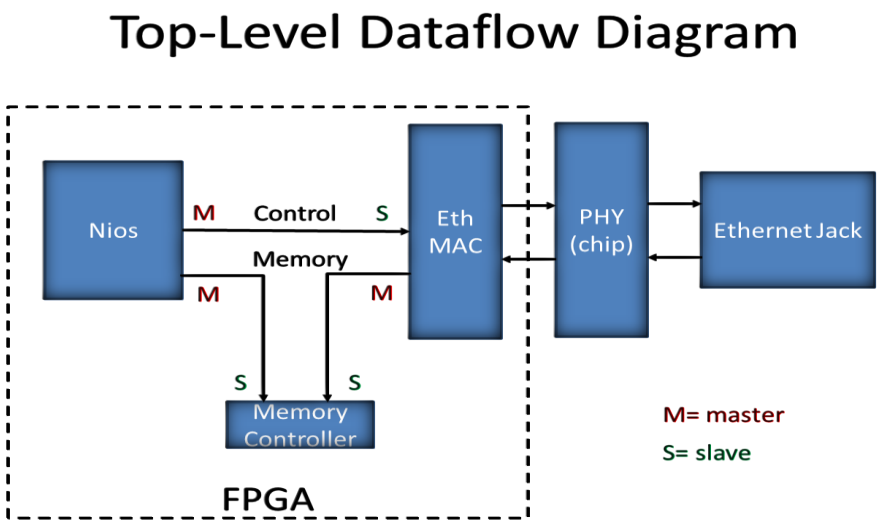


Figure 9 – Top-Level Dataflow Diagram

3.13.1 Microchip KSZ8041

This chip is a single-supply Ethernet physical-layer transceiver for the transmission and receival of data through a CAT-5 unshielded cable. Using this external chip reduces the cost of the MCU since the MCU won't require a more complex layout in order to fit an onboard ethernet PHY. This chip is able to support MII (media independent interface) and RMII (reduced media independent interface)

connections. It operation voltage is at 3.3 volts, which is the same voltage the MCU runs off of. Allowing for a simple power supply to power both chips. This chip only costs $0.67, making it the cheapest and most simple of all the external Ethernet PHY chips available from Microchip.

Pros

- $0.67, very cheap and affordable
- Runs off 3.3 volts, same as the MCU
- Both RMII and MII connections supported

Cons

- Surface mounted chip

3.13.2 Microchip KSZ8081

The KSZ8081 chip is very similar to the KSZ8041 chip, but has a few added features for improved performance and compatibility. The KSZ8081 chip also has 100base-TX/FX, which allows it to be compatible with using fiber optic cables for internet access, fiber-optic internet allows for transfer speeds up to 100 Gbps.

Such speeds will be very excessive for this project, as only small amounts of byte will be needed to be sent through the internet to communicate with the mobile app. The chip is still under a dollar at $0.97, making it very affordable. The chip not only supports MII (media independent interface) and RMII (reduced media independent interface) but it also has SMII (serial media independent interface) support.

SMII is what allows this chip to use fiber optic cables to have transfer speeds up to 100 Mbps. All other features on this chip are the same as the KSZ8041 chip.

Pros

- $0.97
- Fiber optic cable compatibility
- SMII interface

Cons

- Surface mounted chip

3.13.3 Microchip KSZ9031

The KSZ9031 is yea another higher-grade Ethernet physical-layer transceiver when compared to the latter Ethernet-PHY chips offered by Microchip. It offers RMII, MII, and GMII (Gigabit Media Independent Interface) allowing for connection to a gigabit ethernet processor.

This connection will allow for data transfer rates up to 1000Mbps. There is also a RGMII interface allows for connection with a reduced gigabit media independent interface.

The standard version of the KSZ9031 has 48 pins that are lead-free. But it also comes in a 64-pin variety if having more pins that are less divided between tasks is better for the designer. This version of their Ethernet PHY chip also runs on only 3.3 volts, the same voltage levels as the microcontroller; allowing for a simple power supply that only needs to give out a steady 3.3v to power both devices.

The chip also has the option to attach LEDs to display when the device is receiving/transmitting, an option to provide a quick debugging solution.

Pros

- Fast data transfer up to 1000Mbps with fiber optics
- Same voltage level as MCU
- 48 and 64 pin options

Cons

- Twice as expensive as the past Ethernet PHY controllers at $1.87

Table 16 - Ethernet Comparisons

Ethernet PHY	Cost	Transfer speed	Pins	RMII	MII	GMII	RGMII	SMII
KSZ8041	$0.67	10 Mbps	24	Y	Y	N	N	N
KSZ8081	$0.97	100 Mbps	32	Y	Y	N	N	Y
KSZ9031	$1.97	1000Mbps	48/64	Y	Y	Y	Y	N

3.14 Power Supply

In order to make the absolute best possible use of our funds, and to ensure our product is as un-costly for garage owners to install, we plan on creating our own power supply for the Kiosk. This section outlines our research into the subject.

3.14.1 Power

In this section power requirements and power design will be discussed. The requirements in regards to power will be considered for each component individually in the section entitled power requirements and the design which may best fit these requirements will be discussed in complete detail in the section entitled power design.

3.14.2 Power Requirements

In this section, the power requirements of each major component within the overall system will be discussed, as well as the implications of these requirements within the overall system. Most of the components within the project will actually be supplied power via the microcontroller and this will be expanded upon further in a subsection of the power requirements section.

3.14.2.1 MCUs

The MCU used in this project has a requirement of 2.3 to 3.6 V with a maximum of 1 amp draw. The MCU also has many pins which are voltage regulated. This means that certain pins on the MCU are rated for up to 5 volts, while others are rated for up to 3.6 volts. This is a very important detail to note, as the MCU will be driving, powering, many other components within the system. So long as these components do not exceed the rated voltage for the pins they are connected to then there should be no problems in that regard.

3.14.2.2 Lasers

The lasers which drive the optical sensors shall be independent of the microcontroller and thus shall have their own section. The lasers used within this project are a specific type of laser dot diode which are run off of five volts at five milliWatts. Any power supply designed for the laser dot diodes should be able to supply at least these five volts.

The power supply must be able to supply this voltage so that the laser dot diodes may be run at the highest capacity. This means that the lasers would ideally run at five volts at 5 milliWatts so that the highest intensity beam can be delivered from them onto the optical sensors.

3.14.2.3 Peripherals – Card Reader, LCD Display, Keypad

The card reader, LCD display, and the keypad, collectively known as the peripherals, will be powered through the microcontroller. This means that the most important consideration when designing the power supply should be the microcontroller unit because it serves as the central hub of the system. Not only does it receive all the inputs and output all the information needed, it also supplies power to nearly every component, due to those components being connected to the microcontroller.

3.14.3 Power Design

This section will serve to discuss the basics of designing a power supply as well as serving to further understand the needs for power within this project.

3.14.3.1 Transformer

The most common pairs of voltage supplied to wall outlets are 120/240 or 110/220 AC. Clearly these voltages are too high to be used by any low power, direct current

project such as the one detailed throughout this paper. So, what is the answer to this problem? Luckily there is an electronic component known as a transformer which can help to begin to rectify this conundrum.

A transformer is an electrical device that utilizes a phenomenon known as electromagnetic inductance to transfer electrical energy between two circuits. A transformer consists of two coils of conductive metal placed close to each and wound around a core, be it metal or air. The coils that make up the transformer do not have to be metallically connected and are never in direct contact with one another. Figure 11 below serves as a satisfactory visual demonstration of an example of transformers as they appear in circuits.

Transformers serve as a simple and elegant solution in terms of lowering the voltage from wall outlets to a more manageable voltage. The way transformers lower the voltage is described in this equation: Vs = Vp(Ns/Np), where V_s is the voltage of the secondary coil, V_p is the voltage of the primary coil, and N_s and N_p are the amount of turns of the secondary and primary coils respectively.

It is clear to see that if the project group in charge of this project were to decide to design a power supply capable of supplying power in the desired fashion to the project, then a transformer would be a near necessity.

3.14.3.2 Half and Full Wave Rectifiers

After transformation of the AC voltage to a lower AC voltage, the current must then be further transformed into direct current, which is the only form of voltage and current which is beneficial to this project, as well as most electronic projects of this scale.

To begin this process of transforming AC voltage to a DC voltage, there are two electrical circuits known as half wave and full wave rectifiers, which serve to begin conversion of AC voltage to DC voltage.

The half wave rectifier is the less useful, but much more simple rectifier of the two. It essentially takes the AC signal and, through clever usage of a diode which prevents negative voltage from passing, cuts it in half. Therefore, the half wave rectifier causes only half of the alternating current wave to pass through and thus comes the justification of the name ‘half wave rectifier’. Figure 10 below is a visual representation of the operation which a half wave rectifier performs upon an AC signal along with a half wave rectifier circuit.

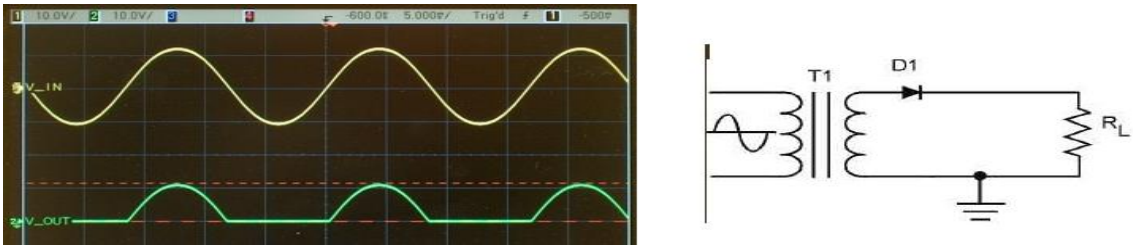


Figure 10 – Halfwave Rectifier Circuit

Quite clearly, a half wave rectifier is inefficient, as it only utilizes half of the power and voltage of the incoming alternating current because it completely negates the negative half of the incoming AC signal and only passes the positive half of the incoming AC signal.

Luckily, there is a solution to solve this inefficiency. This solution comes in the form of the full wave rectifier. As the name suggest, the full wave rectifier improves upon the design of the half wave rectifier by completely utilizing the incoming AC signal. As with any benefit, there must be some trade-off and this trade-off comes in the form of the full wave rectifier circuit being much more complex than that of the half wave rectifier circuit.

There are two main full wave rectifier circuits. The first is a bridge rectifier and the second is a center-tap transformer rectifier. They each have their advantages and disadvantages and the following paragraph will aim to lay out those advantages and disadvantages in a quick but concise manner.

The main advantage of the bridge rectifier is cost. No modifications must be done to the transformer as all components are inserted after the transformer. Cost is also reduced as the only components that must be added are four diodes and diodes are extremely inexpensive, costing about fifty cents per diode. The main disadvantage of the bridge rectifier is that the circuit and design itself is somewhat complex.

The main advantage of the center-tap transformer full wave rectifier is that it is much simpler in design. The main disadvantage of the center-tap transformer is that it is rather high in cost. The transformer itself has to be modified in order to allow for the center tap and transformers are somewhat cumbersome and expensive to modify. One thing to note in the center-tap transformer is that, while it may seem like it would affect the transfer of voltage across coils, through the use of clever minds and even more clever design, the center tap does not actually reduce power or voltage transferred across transformer coils.

3.14.3.3 Smoothing capacitors

A smoothing capacitor is a capacitor with the sole purpose of smoothing out a signal. It can be used to either even out fluctuations in a noisy signal or, in an application which is more relevant to this project, smoothing capacitors can be used to smooth out the ac components of a signal and turn it into a dc signal.

The reason a smoothing capacitor is able to “smooth” out an ac signal and turn it into a dc signal is because of the basic working principle of capacitors. Capacitors are not able to instantly change voltage and thus when capacitors are fully charged, capacitors will slowly discharge rather than immediately change voltage.

3.14.3.4 Voltage Regulators

A voltage regulator is an electrical component that is designed to keep the voltage in a system at a constant level. These voltage regulators essentially work as voltage stabilizers.

A voltage regulator will only allow the voltage to reach a certain predetermined level. A voltage regulator will hold the voltage at this level by rerouting any current that is in excess of the current required to maintain the voltage to a ground point. The current usually also goes through a low value resistor on its way to the ground point.

The power supply design that the voltage regulator is being integrated into should be designed so that the maximum amount of current that it can supply is within the operating range of the voltage regulator. If the regulator receives current that is above the maximum amount of current that the voltage regulator is able to safely divert to a grounding point, then critical failure is almost a certainty.

There are two types of voltage regulators that have application within DC circuits such as the one within this project. The two types of voltage regulators are linear regulators and switching regulators.

The design of linear regulators incorporates devices that operate within their linear region. These regulators typically make heavy use of transistors. The main advantage of a linear voltage regulator is that the output of the linear voltage regulator does not introduce a lot of noise into a circuit in the output of the linear voltage regulator. The largest disadvantage of linear regulators is that they are usually not very efficient and cannot step-up or invert the voltage which is input into them, which a switching regulator is capable of doing. Another benefit of a linear regulator is that they are easy to obtain and are simple to integrate into a power supply design due to the fact that they are usually available as integrated circuits.

Linear regulators also require a higher input than they output. If the input voltage for a linear regulator is very close to the output voltage, the regulator will stop working as the voltage differential between the input and output will be outside of the operating range of the linear regulator. The lowest voltage that this differential voltage can reach and not stop normal operation of the voltage regulator is known as the drop-out voltage.

Switching voltage regulators operate by switching a series device on and off so quickly and at such a high frequency that it doesn’t even seem like the device is being turned off. This makes the switching regulator extremely efficient because the constant switching on and off of the series device means that the device dissipates nearly no power at all. The switching regulator can also step-up or invert the input voltage. These features of a switching regulator make it much more efficient overall than a linear regulator. However, switching regulators are somewhat cumbersome and difficult to implement.

3.14.4 Overall Power Design

The overall design of the power supply for this project will incorporate all the power components listed above. The voltage regulator used shall be a linear regulator, due to its ease of implementation within the overall power design. The two main components which will need power within the system are the lasers and the

microcontrollers. The microcontrollers shall deliver power to all the peripherals, as defined above.

The power supply shall be capable of delivering a maximum of five volts. If the power supply were to exceed five volts, a critical failure might become imminent within the component receiving more than five volts.

3.15 Software Languages

Software is a key component in almost every implementation of an embedded system project. Our project is not different, in this section we researched all possible software languages we could use in order to implement our project.

3.15.1 Java

The Java language project, also known initially as “Oak” was first developed in June 1991 by James Gosling, Mike Sheridan, and Patrick Naughton. The initial plan for Java was for it to be integrated into the digital cable television industry but at the time it was too complex. Java was designed to be like C/C++ when it comes to syntax. It was used mainly in Embedded Systems, Android Applications, Web Applications, Software Tools, and Scientific Applications.

Java can be run on pretty much any Operating system i.e. Windows, IOS, and UNIX based system. This is due to the creation of the Java Virtual Machine (JVM). Currently the easiest way to install java on your machine is to download the Java Development kit (JDK), which contains both the Java Software Environment as well as the Java Runtime Environment.

Pros of using the Java Programming Language:

- It is free to use.
- Contains a tremendous amount of community support when it comes to developing applications and solving problems.
- There are a lot of very helpful IDEs that help you develop your application with less hassle. i.e. Eclipse, NetBeans, IntelliJ
- It can run on almost any platform.
- The amount of written code could be considerably less than using another language because of java’s object oriented nature.

Cons of using the Java Programming Language:

- There is not a lot of community support when integrating java with a raspberry pi.
- We will need to consider the multiple ways our application can be attacked since Java is a versatile Language.
- The Garbage collection used in Java could cause memory leaks, as well as slow down the application itself over time.
- Many of the User Interface libraries of Java are not up to par in comparison to other languages that offer more robust UI control

3.15.2 Python

Guido Van Rossum was the first to publish Python code in February of 1991. The version number as 0.9.0, it included exception handling, functions, and many core data types. It was designed to be an object-oriented language. Python 1.0 was released in January 1994, which included programming tools like lambda, map, and filters. The latest python is 3.0 which was released in 2008, the main change of 3.0 was to reduce the number of duplicate features by designing using newer techniques.

Pros of using Python

- Lots of community and native support when using python with a raspberry pi microcontroller.
- Open Source, can be used on over 21 different platforms including Windows, Linux, and MacOS
- Multiparadigm brings lots of benefits unlike java where you need to create a class to make a OO object, in python everything is an object
- Python is easier to understand when trying to write asynchronous tasks

Cons of using Python

- Slow execution speed
- Design Restrictions because the language is dynamically typed
- Will need more debugging due to errors only showing up at runtime.
- Group familiarity with Python, only one of use have used it prior to doing this research.

3.15.3 C++

The history of C++ goes all the way back to 1979 where Bjarne Stroustrup was working on his Ph.D. While writing his thesis, he began working on the first implementation of C++ called “C with Classes”. In 1983 the name of the language changed to C++. Up to 2014 there many publications and books written on the language, as well as new versions of C++.

Pros of C++ are a huge amount of community support with microcontrollers, it combines the low-level power of C which also works well for microcontrollers, and it contains several layers of abstraction such as OO style, functional, and imperative programming. The cons of using C++ are that it is a very extensive sized language since it is based off C, Memory management is also difficult since it is also based off C but they have made updates to help ease it.

3.15.4 C

Development on C began in 1972 at Bell Labs by Dennis Ritchie and Ken Thompson, when they wanted to port the operating system they wrote, UNIX, to the PDP-11. The name was chosen as homage to a previous language written by

Thompson, called B, which served as a predecessor to the language. C first appeared in Version 2 of Unix, and would eventually be the sole language used in the operating system's codebase, save for the occasional assembly.

In 1978, they published "The C Programming Language by Kernighan and Ritchie," colloquially known as "K&R C", which served as the first description of the language other than a compiler. Still, to this day, many people still consider this book to be the go-to reference on the language. Later, the language received formal standardization, first by ANSI, then by the ISO. Interestingly, this book contained the first true "Hello, world\n" example. In fact, many believe that the trend of using the phrase "Hello world\n" is a direct consequence of this book's wide popularity.

As a language, C has very few features, causing many people to refer to it as "UNIX assembly" by many, even going as far back as the 1970's when high-level programming languages were not nearly as advanced as they are today. C requires the programmer to work with raw memory addresses, in the form of pointers where other languages usually introduce the concept of a reference, and always adds bounds checks to array accesses. When C programmers forget to add these checks themselves it can often cause errors known as buffer overruns, which are often a security vulnerability.

Despite C's deficiencies, it is the premier language for use in embedded and systems programming. Many consider C's frequent use of raw memory addresses as an advantage, since interfacing with peripherals often requires reading or writing to a specific memory address. Furthermore, data structures in C are almost always as big as you specify them to be, so predicting memory usage is easy. Since it is still a high-level language, there is no need to memorize archaic instruction sets, and programmers are able to leverage decades of knowledge and libraries in writing their code.

In our project, we will certainly utilize the C programming language in writing code for the embedded microprocessor embedded in the garage. Its ability to operate at a low level, as well as its compiler support for nearly every architecture and platform on the market is a combination nearly unmatched by any other option.

3.15.5 Haskell

Haskell is a Functional Programming language, that is meant to be much more oriented towards declarative programming, rather than imperative programming. Declarative programming is a style of programming where the programmer creates constraints and definitions for the program, and the compiler creates a program that satisfies those constraints.

With Declarative programming usually has no explicit restriction on what order things execute, only the implicit restrictions created by dependency of information. Imperative programming, however, is akin to creating a step-by step recipe for a meal: every step is intended to execute one after the other, and it is up to the

compiler to find sections of code where this contract can be ignored. In a declarative language, such as Haskell, it is common to not have any kind of explicit control flow statements, such as loops, available. Instead, programmers must make use of recursion instead.

The mark of a Functional programming language, such as haskell, is the ability to treat functions as first class values. This mean functions can be passed as parameters to other functions, or stored in a data structure, as a normal object. The second mark of a functional programming language is the distinct aversion to explicit statefulness. most functions in haskell are what is known as “pure functions” which are functions which have no side effects and which will always return the same result when given the same inputs.

Pure functions are easier to understand, and also allow the compiler to perform interesting optimizations, such as automatic memorization. Another nice feature of haskell is the extremely powerful pattern matching built into the language. This pattern matching is not to be confused with regular expressions. Haskell’s pattern matching allows programmers to write the base case of their recursion in a very literal form, and write the general form more generally.

Functional programming in Haskell is generally considered a more elegant, error - free way to write code. Rather than trying to think in terms of the steps required to arrive at the solution, functional programming encourages programmers to think about what the solution is.

Unfortunately, Haskell is not used very widely, and support for it is lacking, compared to other languages. Most platforms do not even have a working Haskell compiler, other than ARM and x86. As a result, we will not be using this language for any notable amount of code.

3.15.6 Erlang

Erlang is a functional programming language, rather than Object - Oriented or Procedural. This is because it treats functions as first class values, allowing you to pass function objects as parameters to other functions or store them as a field inside an object. Although Functional programming is not as widespread as other kinds of programming, it is widely considered significantly more elegant.

Erlang was developed initially for the telecommunications industry, and has strong built-in support for distributed programming, where computations are split between processes that may or may not be on the same physical machine. These features are extremely powerful for creating apps that can easily be scaled.

One of erlang’s core features is the concept of “message passing,” which is when one Erlang instance sends a message that is received by another instance which does something in response to that message. The two instances can be separate processes, or be on separate machines connected by a network, and erlang will handle the whole process of delivering that message by itself.

Erlang implementations typically utilize a virtual machine which interprets and compiles bytecode at runtime, in an architecture that is similar to that of Java. Such a virtual machine typically demands minimum hardware capabilities, such as an operating system.

Very few free-standing erlang implementations exist, and as such erlang will almost certainly not be used on our micro controller. Without having an erlang node running on the client side, many of the benefits of running erlang on the server are immediately negated, since erlang's message passing functionality only makes sense when communicating with another erlang node,

3.15.7 JavaScript

JavaScript was created in 1995 by Brendan Eich, who was working at Netscape at the time. The original name was Mocha, but then switched to JavaScript once a trademark was sent from Sun Microsystems. JavaScript was created to make the Web more dynamic. It was made to tie HTML to a logical language to interact with web pages [5].

3.15.7.1 JQuery

JQuery is a free open source JavaScript Library used for client-side scripting of HTML. JQuery makes it easier to navigate through DOMs, create animations, handle input, and design Ajax applications. It was first released in January of 2006 by John Resig, and is currently maintained by a team of developers led by Timmy Willison.

Pros of using JQuery are the use of Ajax requests being simplified in the code, as well as JSON parsing, allowing for easier transfer of information in a set format. The cons of using JQuery is that the code can be hard to understand when not familiar with it. As well as there being a lot more code in general because JQuery requires a lot more code to do simple tasks.

3.15.7.2 AngularJS

AngularJS was created by Miško Hevery and Adam Abrons in 2009. AngularJS was built on the principle of declarative programming, it expands the HTML by connecting it through a two way bind process that allows for synchronization of Views and Models. The Pros of using AngularJS is that it is a very flexible language, as well as easier to understand when looking at the code.

AngularJS is another open source JavaScript Library that some of our group members are familiar with. It excels at building dynamic single page web applications. The cons of using AngularJS include possible errors in error reporting. Also, the learning curve may be steeper for group members who have no experience using it.

3.15.8 Ionic Framework

Ionic is a Mobile Application Framework that was originally developed on HTML, AngularJS, and CSS. It is an MIT Licensed software meaning anyone can use it for commercial or personal reasons, it is 100% open-sourced. It was created in 2013 by Drifty Corporation, founded by Ben Sperry and Max Lynch. Ionic provides many native features for building mobile applications for Android, IOS, and Windows through the Cordova Plugin. It allows you to develop for all said platforms simultaneously, allowing our application to be used by a wide variety of mobile users.

Ionic 2 which is the latest version of Ionic uses TypeScript a version of JavaScript. Since it uses TypeScript it allows the mobile application to be very robust, also it allows for quick development since it focuses on HTML, CSS, and JavaScript which are known for being easier to learn languages.

3.15.9 TypeScript

TypeScript was introduced in October 2012 after around two years of development at Microsoft. TypeScript is a modified version of JavaScript made for large-scale applications. TypeScript has safe automatic refactoring, as well as syntax errors caught at compilation time. It contains module support as well class-based OO with inheritance.

Cons of typescript are some different from the standard type annotations, as well as our group not being familiar to using typescript, only one group member has coded in typescript before. Another con is that it does need to be compiled into interpreted JavaScript, but for purposes of our project it fits well since it will be compiled into our mobile application.

3.16 Database Management Systems (DBMS)

Database Management Systems are often used in applications today to store user information, as well as for application data storage. They are used to allow for the creation, manipulation, updating, and querying of databases. Databases use SQL (Structured Query Language) to manage and manipulate data. DBMS operate by using SQL as well as some sort of ODBC (Open Database Connection) or JDBC (Java Database Connection) Connection.

3.16.1 MySQL

A Swedish company called MySQL AB created MySQL and released the first version on May 23 1995 [6]. They created a new SQL interface while keeping the same API as mSQL a well-known language at the time. It has been open-sourced since 2000, and has had a tremendous amount of support from 3rd party developers. MySQL was eventually acquired by Sun Microsystems which then was acquired by Oracle.

Pros of MySQL

- Pros of using MySQL is that it is focused on web and cloud development.
- Also since Oracle acquired MySQL the development of MySQL has been revamped because of the amount of resources Oracle brings with the other companies they have acquired such as InnoDB.
- MySQL has a ton of Community support, and it is open-sourced for certain editions
- Our Team is familiar with using MySQL

Cons of MySQL

- Cons of using MySQL are that it is not as mature as other relational databases such as PostgreSQL.
- Also for commercial use MySQL requires licensing.
- Limited in certain areas of fault tolerance, and performance diagnostics.

3.16.2 Apache Cassandra

Cassandra is a database management system based on NoSQL (Non-relational) system, which was created to handle large sizes of data across multiple servers, allowing for there to be no single point of failure. Avinash Lakshman and Prashant Malik created Cassandra when they were both working at Facebook. They released it to the public on Google Code in July 2008, it later became an Apache Incubator project in 2009.

Cassandra excels in write speeds, it can handle massive amount of write volumes which can be useful for our application. Other advantages of Cassandra are that it is JVM based which allows for cleaner integration of java applications. Some faults that Cassandra has is that there are no Ad-Hoc Queries, as well as no Aggregations such as SUM, MIN, AVG, and MAX which might hinder performance server side since it is extremely resource expensive on the database side.

3.16.3 Microsoft SQL Server

Microsoft first developed a 16-bit server for their OS/2 operating system in 1989. It is a relational database system based on SQL. It can be bought as different editions, the one we could use would be the “Express” version. Which is a scaled down version of the standard edition, and is free.

Pros of using SQL server are that it contains loads of helpful documentation since it is still supported by Microsoft. Also, it is very simple to install as well as maintain. Over the course of its development it has matured with each new release, but it still has a hefty price tag if you want to use a full scaled version. But for personal or small project use the “Express” version of SQL Server can suffice if you are willing to skimp on the amount of memory and storage that you can use.

3.16.4 Oracle

Oracle as a company was founded in 1977 under the name Relational Software, Inc. Later named Oracle Corporation. They created the first commercially available relational database management system in 1979, with their latest version releasing in March of 2017 as the 12c Release 2 version.

Oracle is the oldest and most tested database management system for large scale systems. It is known for its ability to handle huge amounts of data transfer. The weaknesses of Oracle are the expense of using their DBMS. For our project, we would not need the size or speed of an Oracle database, and it would also bring great expense to our project where we are trying to keep expense as low as possible.

3.17 Web Server Platforms

A Web Server is a computer system that takes in “requests” over HTTP and handles those requests accordingly. The main function of web servers is to send out web pages to clients, and to also manipulate those web pages based off user requests using server-side scripting. Lots of web servers are used in embedded systems today such as printers, routers, and webcams.

We will be incorporating a web server to enable server-side scripting and manipulation of a simple web page on our garage based system as well as the mobile device system, both systems will communicate with the same web server which will handle similar requests from both external systems. We decided to go this route because it seemed to be the most intuitive way of creating our application since it will make most of the heavy work stay on a server instead of the clients application.

3.17.1 Node.js

Node.js was originally written by Ryan Dahl around 2009, it was maintained by Dahl and later sponsored by Joyent. Node was originally released for Linux and Mac OS X by Dahl. In June 2011 Microsoft and Joyent developed a version for Windows and released it in July. Dahl criticized other popular web servers such as Apache HTTP Server for their lack of concurrent connections and how it was based on sequential programming where you might have to loop through the same code for every connection even though you might not need to.

Pros of Node.js

- Node is based on JavaScript which is easy to learn, also our team is familiar with JavaScript.
- NPM (Node packaged modules) is one of the most used packaged module systems and is still growing.

- Asynchronous events drive I/O which helps with concurrent request handling. Both client and server can share the same pieces of code.
- Node has a huge amount of community support.

Cons of Node.js

- Node does not provide great scalability or use of multiple CPU cores
- Dealing with Relational Databases can be rough if you do not use the correct premade library's
- Node is not best prepped to handle CPU-intensive tasks, it is more suited to I/O.
- Node can be difficult to understand if you don't understand core concepts of JavaScript.

3.17.2 Apache HTTP Server

The first version of the Apache HTTP Server was created by Robert McCool who then left the National Center for Supercomputing Applications in mid-1994. When McCool left, it left the HTTP Server with many holes that needed to be fixed. This is where other developers started pitching in, which created the original "Apache Group" of developers that created the HTTP Server. Apache hosts roughly 39.25% of all currently hosted websites.

Apache has low costs due to not having any software licensing fees. It also has program flexibility and community support due to it being open source. It runs on every common operating system. Has enhanced security due to it being originally developed on a non-windows system. It is a process-based server which requires a thread for every simultaneous connection, this creates a significant amount of overhead.

3.17.3 Microsoft IIS

IIS (Internet Information Services) was first published as IIS 1.0 as an add-on of Windows NT 3.51 operating system. IIS is composed of different services such as File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), HTTPS, Network News Transfer Protocol (NNTP), and Simple Mail Transfer Protocol (SMTP). The latest version is IIS10 which is included on Windows 10 and Server 2016. IIS

IIS is consistent and flexible when it comes to website hosting. It has great integration with MS SQL Server. But since it is based off windows, Linux or Mac OS X based systems could run into some compatibility errors. Web hosting through IIS is not free, so this would add another cost to our project.

3.17.4 Nginx

Igor Sysoev first developed Nginx in 2002 to solve the problem of websites getting over 500 million requests a day and not crashing the enter site. Sysoev founded a

company by the same name in July 2011. The original design of Nginx was specifically made to outperform Apache Web Server, by serving static files faster as well as using a great deal less amount of memory and handle way more requests.

Nginx is event based so it can handle more connections with less overhead. It has a low memory footprint, which was a part of the original design. Nginx also has a friendly configuration as well as a modular design. But it does have less flexibility such as override of system wide access on a per-file basis. Nginx also has less community support as compared to Apache, as well as modules and extensions.

3.18 Software Development Styles

Software has been around since the first days of the computer, eventually companies wanted to come up with efficient solutions to developing software application in order to save money and create better products. That's where software development methods began to form into entire entities of their own. In this section, we discus two main development methods used today and the pros and cons of each of them.

3.18.1 Waterfall Software Development

The Software development approach of the Waterfall model follows a non-iterative process where progress is shown steadily streaming downwards, from the initial conception all the way to integration and testing. The first formal presentation of this method was by Winston W. Royce who did not use the term waterfall but described the process of the development as a top-down approach that was flawed.

Since documentation is a must in the waterfall method it allows for new designers and programmers to get up to speed on what the project is. Schedules also have strict deadlines not allowing for the next part of the development to start unless the previous sections are completely finished. But it may be hard for a customer to describe every single requirement their system might need at the beginning, and waterfall does not allow for new requirements to be written in once that phase is completed. Also, projects may take longer to deliver since this method has such strict rules on moving on to the next section of the development cycle.

3.18.2 AGILE Software Development

AGILE describes a set of principles on which software should be developed using an iterative process. It emphasizes on flexibility, adaptive planning, continuous improvement, and constant testing. AGILE in 2001 made its headline with the *Manifesto of Agile Software Development* which was written at a meeting by 17 software developers. Which was only the initial creation of AGILE, it would later be developed into a much more mature method by other developers.

With AGILE comes more flexibility on every aspect of your project, it allows you to integrate and test while still developing. It usually leads to fewer defects in the final product due to that constant testing. Each iteration provides constant feedback on what needs to be improved on in the software lifecycle. But documentation is a low priority when it comes to AGILE development, and it can also lead to the possibility of the customer adding/wanting more and more features since they see constant feedback from testing the software.

3.19 Initial Software Design

This section will go into detail of our initial thoughts and ideas of how to go about designing the software for our project. It discusses the multiple ideas we had for developing our garage system as well as our mobile system. We discuss our initial ideas on how to go about creating the software needed to complete our project.

3.19.1 Initial Web Server/Garage System

Through the research our group has conducted we have decided to choose Node.js as our web server platform, because of its ability to handle great amounts of requests as well as its ability to handle I/O. Our team is also familiar with JavaScript the native language on which Node.js was built upon. This will allow us to develop our web server quickly and can fix bugs that could arise during the implementation phase of senior design II.

Our Web Server will handle requests sent from “clients”. The clients can either be the garage systems, or the mobile application clients. The Garage system will host its own private web server using node.js, which will be used to host a small client application on that specific system.

This client application will send out requests to a master web server hosted on a different machine, that will then process the requests sent by the client. This could be a normal update of number of available spots left in a garage, or from a user inputting the spot they parked using the garage system. As well as a user trying to retrieve the location of their vehicle.

If a user has never used the system before there will be an option for the user to create an account using a simple form system, which will also ask them to slide their UCF ID if they would like to be able to access that feature of the system of being able to swipe their ID instead of always inputting their information every time they want to access the application.

The garage system will also be smart enough to tell a user if their car is in the garage, as well as the specific floor and individual spot. It will also be smart enough to tell the user what garage it is in if it is not located in the garage that the system the user is accessing is in.

We took this approach to allow for scalability of multiple garages and multiple mobile applications. Each garage application will have a user enter their own

unique id when accessing the main web server, just like the mobile application user would. Each garage system would require a user to enter a username (possibly PID for a specific UCF case) and a four to six-digit passcode, which would be encrypted on the client and then authenticated on the server side. From there the user would be able to enter the location they parked their vehicle, and this location would be saved to the database that will be hosted on the same machine as the main web server.

The system will be smart enough to overwrite a user's past location if the user never searched for their vehicle after storing its location during another use of the application. This will fix the duplication issue of having a user store their vehicle's location in multiple spots.

We will also try to implement if time permits multiple solutions of having users enter their information in a faster and more secure way, such as swiping their UCF ID. Another possibility could be giving each user a specific tag that can be scanned at the garage system that could contain their information, but these solutions have their own risk to them which might not be ideal for our project. As well as other tasks, such as learning what is stored on a UCF ID, how to access the data properly and manipulate it to our needs.

3.19.2 Initial Mobile Application Design

The Mobile application will be developed using the same philosophy of the garage system with some more features. The framework that will be used is Ionic, since it gives us the greatest range of mobile devices while also giving us a very similar platform of languages to develop on.

The mobile application is being developed to also give the user access to information such as the number of cars currently in a garage as well as the number of spots that garage contains. This information is specifically only being given to the mobile application because a user would not need to know how many spots a garage currently has if they have already parked and accessed the garage system.

The basic layout of the application will be when a user first opens it, it will give the user three options. The first being to retrieve information on certain garages. This information is constantly updated via our sensor system that sends requests to the main web server which then updates the database.

The mobile application will send a request to the web server which then retrieves the information of all or possibly specific garages with their current capacity and current vehicle amount. This will be displayed to the user in a very simple interface. The user will also be able to easily access the second option of the mobile application which is to find/store the location of their vehicle even if they did not select it on the original startup of the application.

The second option of a user finding/storing the location of their vehicle will be created using a very simple layout, allowing the user to store the location of the vehicle by manually entering the location using a simple form system. Or possibly

given enough time we want to implement a QR system (Quick Response System) which allows a user to scan a QR code that would be a unique code for every spot in a garage, which will then be uploaded to the web server via a request which is then cross checked against the database and then the web server will store the information of the location of that that was just entered by the user.

Each QR code must be generated for each individual spot, which will take some time, but on the bright side it does not cost a lot of money to create a good amount of them. Based on time we will most likely implement this feature for a few spots in a garage to show feasibility.

The last option given to the user will be the creation of an account. The third and final option allows the user to create an account using a simple form system. It will be a quick easy way for them to create an account, they will input a username (we might have them specifically use their PID pending further research of the information stored on UCF IDs) as well as a four to six-digit passcode, which is all encrypted and sent over to the web server to store in the database.

We would also like to implement a day parking pass feature in the mobile application. This is something we believe we can implement if time permits. A user would be given the option to pay for a day parking pass, all they would have to do is make an account using a feature mentioned earlier, then through a secure pay option they are given a day pass they would also have to enter extra information such as their make and model of their vehicle as well as the license plate number. We would handle the purchase on the server end, and generate a day pass code and have an individual table for users who are buying day passes.

From there admins will be able to login via an admin user and access the database table of current day pass purchases and cross reference the table with the vehicle they are checking. We would also like to implement further with a specific UCF system of checking for outdated parking decals. By using a user's login credentials and personal vehicle information to also be stored in our database

Other possibilities we would like to have for the mobile application is to be able to show the user when certain garages are usually full and when they are not full. Also, allow for the user to search for the nearest not full garage that is closest to the class or building they need to go to. These features are not in the core scope of our project and will be implemented if time permits.

3.19.3 Initial Database Design

The database will be hosted on the same machine the web server will be hosted on. We have decided through our research to use MySQL for this project. Due to its easiness to setup as well as our team's familiarity with it makes it a perfect choice.

The database will hold information for every user, basic information stored in one table. The user's information will be used to authenticate them when they login to either the garage or mobile system. We will use another table to store each user's

specific location of their vehicle as well as retrieve their vehicles location. When a vehicle location is retrieved, we will give the user a certain amount of time and then delete their vehicles location, or another approach we were thinking of was to only overwrite a user's vehicle location when they try to store their location again, we would check if the user had an entry in the table and if the user was trying to store their vehicles location again we would delete the current entry and store their new entry.

Below we have created a draft of our database schema in Figure 11. It contains four tables, one for general users, and one for tracking specific spots, one to help setup garages, and lastly one for day parking pass users. We decided to use a relational database because of its ability to create unique primary keys based off of other tables, creating that relationship between tables allows us to design our webserver in a simpler fashion. Making the database do simpler and faster work that the webserver does not need to do.

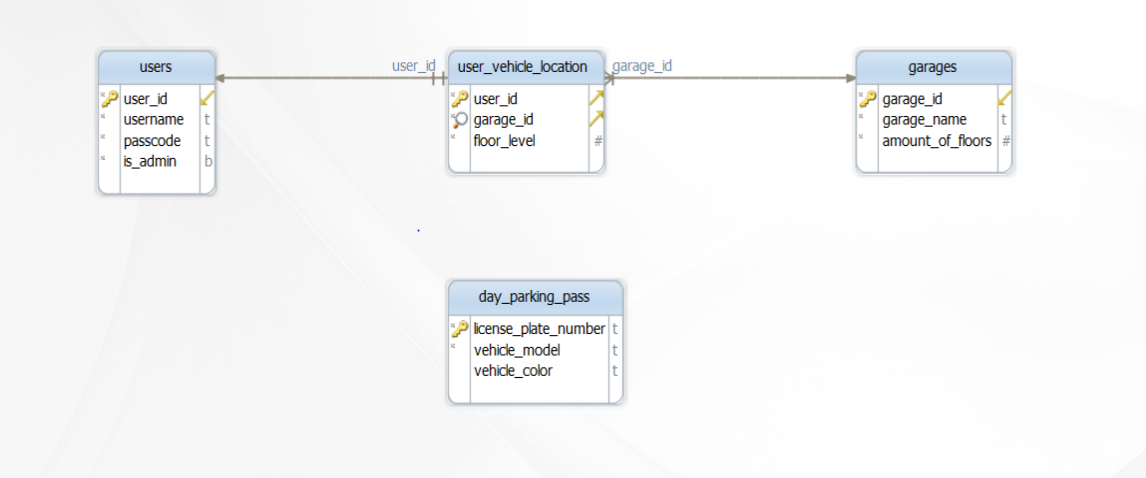


Figure 11 – Initial Database Design

4 Standards & Design Constraints

For our project, we have researched and designed many distinct aspects, in this section we discuss constraints we have seen and overcome. As well as go over standards of technology and algorithms and how we applied them to our project.

4.1 Standards

A well-designed project adheres to standards created and used by the community. In this section, we will describe each standard used in our project, its history, as well was where we implemented said standard in our project.

4.1.1 ISO/IEC 7810

ISO 7810 defines the dimensions of the cards. For cards of the format ID-1, which are the kind that are used for UCF student and faculty ID cards, they are specified to be 3-3/8” x2-1/8” and all formats have a thickness of 1/32 inches. Interestingly the original standard actually defines all these dimensions in terms of the imperial system. ISO 7810 also defines certain physical characteristics as required, such as the bending stiffness, flammability, toxicity, resistance to chemicals, dimensional stability with respect to temperature and other environmental changes, resistance to wear and tear and durability.

4.1.2 ISO/IEC 7811

ISO/IEC 7811 on the other hand is concerned with the recording of data on these cards. It is made up of 9 standards, labelled Part 1 – Part 9, however only part 2 is relevant to this project. According to ISO’s official publicly available free documentation, ISO/IEC 7811-2:2014 “specifies requirements for a low coercivity magnetic stripe (including any protective overlay) on an identification card, the encoding technique and coded character sets. It takes into consideration both human and machine aspects and states minimum requirements.”

4.1.3 Cryptography Standards

Cryptography is the practice of securing communication and information between two parties. With the rise of “hackers” or rather computer programmers who use bugs and exploits to gain access and information from systems not of their own, we as a group must create our project in order to protect our user’s information to the best of our ability. We will be following specific cryptography standards in order to protect our user’s information such as their cars location, as well as their username and passcode.

Data Encryption Standards were developed in the early 1970s at IBM. This standard was the earliest development of cryptography in the computer age. It was heavily scrutinized for its design elements and short key length (56-bit key). Due to its small key size, it was relatively easy as computers got faster to break the cipher, thus Advanced Encryption Standard was made.

AES (Advanced Encryption Standard) was established in 2001 by the NIST (US National Institute of Standards and Technology) as the standard used by the US government. It was created by two Belgian cryptographers Vincent Rijmen and Joan Daemen. AES is a symmetric key algorithm, which means it uses the same key to encrypt and decrypt data being transferred. AES contains three different sizes of keys 128, 192, and 256-bit keys. 256 length keys since 2003 have been used to protect classified information for the US government. To break a 256-bit key using brute force it would require 3x10^51 years using supercomputers that could check 10^18 AES keys per second. This shows how powerful AES encryption is, while it is not unbreakable it is very difficult to do.

For our project, we will be incorporating AES in the transfer of login credentials from the garage or mobile system over to the web server. We will encrypt both the username and password and then send that information over to the web server which will decrypt it and then authenticate the user by checking their credentials in the database. Once a user is authenticated they will be able to receive the location of their vehicle as well as store the location of their vehicle.

4.1.4 Laser Standards and Classifications

There are many standards and classifications for lasers, but the focus of this section will be on the standards set upon lasers by the American National Standards Institute. The current version of the ANSI Z135.1 Standard places lasers into one out of four categories based on their potential to inflict biological devastation.

These classifications are determined through the use of calculations which take into account exposure time, laser wavelength, and average power of the laser. The calculations are then utilized to define a factor known as the Accessible Emission Limit. The AEL is the product of the Maximum Permissible Exposure limit, which is defined in the Standard listed above, and an area which is defined as the Limiting Aperture. The Limiting Aperture is based on such variables as laser wavelength, fully dilated pupil size, and beam hotspots. [2]

According to the ANSI Standard described above, there are four classes of lasers, with the third class of lasers being split into two subclasses. That is to say, these classes are Class 1, 2, 3a, 3b, and 4. The only classes of laser that this project is concerned with are classes 1, 2, and 3a, as these are the only class of laser which are available for consumer use.

The laser which is incorporated into the Garaginator system as an integral part of the optical sensor is classified as a class 3a laser. A class 3a laser has an output which is between one and five times the AEL for class 1 lasers. The wavelengths for class 3a lasers are shorter than 0.4 micrometers or longer than 0.7 micrometers. The output for a class 3a laser is restricted to less than 5 milliWatts.

The specific laser used in the project is a laser dot diode which operates at a wavelength of 650 nanometers, so it is a red laser. The power of the laser does not exceed 5 milliWatts. Therefore, it clearly falls into the class 3a category. A laser which is categorized as class 3a may cause serious damage to the retina of the eye if the beam comes into direct contact with the eye for an amount of time greater than or equal to two minutes. A class 3a laser may also be dangerous when used in conjunction with an optical tool which enhances the diameter of the laser beam or the power density of the laser beam. However, since no such optical tool will be used within the project, there is no cause for concern from that aspect of the standards. A class 3a laser system must also be labeled with either a “caution” or a “danger” warning label.

The only safety concerns that need be considered within the scope of this project in regard to lasers, are those which state that direct exposure from a class 3a laser

may cause severe damage to the retina. The project team is extremely confident that the laser shall be placed in such a way that the only way the laser will ever come into direct contact with the retina of a human being is through either ignorance or mishandling. It is simply impossible to negate either of these scenarios completely, although the project team shall limit the extent that the lasers may be accessed by the average person.

4.1.5 IEEE 802.3

This IEEE standard is a collection of standards and a working group of people dedicated to laying out the physical layer and data link layer of media access control devices (MAC) for wired ethernet connections. Since the first official standard in 1985 for Ethernet, the IEEE 802.3a for 10 Mbit/s speeds, there have been numerous new 802.3 standards added to encompass the developing technology and improvements over the decades.

Each new 802.3 standard is given a specific designation in order to be identified easily. The latest addition was the 802.3 cd, which covers the new parameters for Media Access Control for 50 Gbit/s speeds, along with the parameters for 100 and 200 Gbit/s transfer speeds of operation. This standard and its many revisions allow for a uniform and reliable connections to the internet via ethernet rj45 connectors for both consumer and professional applications.

4.1.6 IEEE 802.11

The IEEE 802.11 standard convers the use of Wi-Fi technology. The standard was first formed in 1977 by IEEE to create the first standard for WLAN's (Wireless Local Access Network). This first standard only covered the connection speeds up to 2 Mbps, which is far too slow for most user applications. Over the years many new wWi-Fi technologies and capabilities have been added, and new 802.11 standards have been made to keep up with the changing technology. There are now almost as many 802.11 standards as the number of letters in the alphabet, starting from 802.11a, all the way to 802.11y.

The five most important 802.11 standards are 802.11b, a, g, n, and ac. Each one being a greater improvement on the last. 802.11a has very fast connection speeds up to 54 Mbps using 5 GHz frequency for reduced signal interference, but is quite expensive compared to other Wi-Fi versions and has a shorter signal range. 802.11b is the cheapest Wi-Fi version with good signal range that isn't easily obstructed at its 2.4 GHz frequency spectrum.

Although, it has the slowest speed of the five, at just 11 Mbps. Then there is 802.11g, this standard is an attempt to combine the best of 802.11a and 802.11b. 802.11 devices allow for bandwidth up to 54 Mbps in the 2.4 GHz signal range. So it has the best of both worlds with high data transfer rates and great signal range. But because of this it is very expensive. 802.11n is a great bandwidth improvement upon the 802.11g devices, allow for up to 300 Mbps transfer speeds and network bandwidth. The signal intensity is also increased in these devices and allows for

greater ranges to be achieved. But, the standard is still a work in progress and currently costs much more than the 802.11g devices. Then the newest addition to the 802.11 standards is the 802.11ac. “ac” uses dual-bandwidth technology, so it supports signal connections in 2.4GHz and 5GHz frequency ranges. Because of this the 802.11ac devices are all backwards combatable with all 802.11b/g/n devices. 802.11ac also allows for data bandwidths up to 1300 Mbps in 5GHz and up to 450 Mbps in 2.4GHz.

For this project application, any current 802.11 Wi-Fi module will have enough bandwidth and range. The 802.11 Wi-Fi standard also apply to all Wi-Fi ad hoc devices.

4.1.7 IEEE 802.15/Bluetooth SIG

The 802.15 standards are Bluetooth standards that are no longer maintained by IEEE since 2011 and are now maintained an organization called Bluetooth SIG (Special Interest Group) maintain and update the standards for Bluetooth technologies. All manufacturers must meet the requirements of the Bluetooth SIG standards in order to market it as a Bluetooth device. Many revisions and additions to the 802.15 standards have taken place. The first standard for Bluetooth was made in 2002 by IEEE, starting with the 802.15.1 standard which defined the physical layer (PHY) and the MAC (Media Access Control) specifications. The most recent update to the standard was the amendment for Smart Utility Network in 2012 by the Bluetooth SIG organization.

This amendment was for the 802.15.4 standard and allows for very large-scale process control applications such as smart grid networks with millions of endpoints. This amendment was done by the Smart Utility Networks (SUN) task group created by the Bluetooth SIG organization. New standards created by the Bluetooth Special Interest Group are called Core Specifications.

There are currently five active Core Specifications create by the SIG, the most recent being Core Specification Addendum (CSA) 6. CSA 6 was adopted on July 12th of 2017. These Core Specification are the defining features of a Bluetooth device that developers must use to create accepted Bluetooth devices. The CSA 6 is a guide which define what types of devices Bluetooth devices can be and what they must allow and be capable of to be called such Bluetooth devices.

4.1.8 UM10204 - I2C-Bus Specification

Philips Semiconductors, no called NXP Semiconductors, created I2C as a way to have efficient and cheap 2-way control on multiple devices. Only two lines are needed to run this kind of setup for serial line data (SDA) and serial clock line (SCL). With this technology data transfer rates up to 100 kbps can be achieved in standard mode, 400 in fast mode, 1 Mbps in fast mode plus, and 3.4 Mbps in high speed mode. Then there is an ultra-fast mode allowing for uni-directional mode for a data transfer rate of 5 Mbps.

The first release of the I2C Specification by Philips was in 1982, Version 1.0 was released in 1992, and the most recent release called Version v.6 was on April 4th 2014. This was a simple revision of the user manual and applications of the technology. One of the most important revisions of the technology came with the Version 2 revision in 1998. At that time the I2C-Bus became a world standard and licensed to over 50 different technology companies.

This updated version allowed for I2C with faster bus speeds and lower supply voltages, this change was to cover the need of modern technology's demand for power efficiency and faster data transfer rates.

4.1.9 USB-IF

Universal Serial Bus Implementers Forum is a non-for-profit organization that maintains the standards and promotion of USB devices and applications. The organization covers all USB applications and technologies such as USB, Wireless USB, and USB On-The-Go. The group was originally formed in 1995 by a large number of technology companies. Including but not limited to, Apple Inc. Microsoft, and Hewlett-Packard. The most recent addition to the USB collection is the new connection called USB Type-C, allow for data transfer rates up to 10 Gbps and device charging capabilities up to 100 watts of power.

Type-C was finalized in August of 2014, nearly the same time as the USB 3.1 specification was finished. Type-C is special in that it is able to bet plugged-in in two different configurations, right side up, or up side down. Either configuration will function normally. There are a number of licensed USB vendors which offer courses on USB standards and function for a small fee, some of these vendors are Microchip, Texas Instruments, and Silicon Labs.

USB OTG (On-The-Go) is one of the most commonly used universal serial bus standards. OTG technology has been around since 2001 and allows for devices to act as a host to control peripheral devices, such as flash drive, mice, keyboards, and web cameras. USB OTG is also controlled and standardized by the Universal Serial Bus Implementers Forum.

4.1.10 Power Standards

Power standards are divided into two sections: power safety standards and power efficiency standards. The subsection of power safety standards shall touch upon the topic of the standards introduced by various agencies of the world which help to protect the average person from power supply units. The subsection of power efficiency standards shall elucidate those standards which help to make power supply units more efficient.

4.1.10.1 Power Safety Standards

While it may seem that power safety standards would take into consideration only electrical hazards, there are actually a great many other aspects of power supply

design which are guided by power safety standards. This list of aspects includes: electric shock, energy hazards, fire, heat related hazards, and mechanical issues.

The power safety standards, which are provided by CUI Inc., divide power circuits into four different classifications. These classifications are hazardous voltage, extra-low voltage, safety extra-low voltage, and limited current circuits. The power supply for this project will meet the definition of the extra-low voltage classification and thus this will be the classification considered herein.

The extra-low voltage circuit is defined as “a voltage in a secondary circuit not exceeding 42.4 VAC peak or 60 VDC, the circuit being separated from hazardous voltage by at least basic insulation” [3]. The power supply for the Garaginator will output at least 5 volts. Wires which run from the power supply to the components shall be insulated in order to avoid any possible electric shock risks. The printed circuit board itself shall include some form of insulating shielding which will serve to protect the printed circuit board from any metal or other conductive material it may, by happenstance, come into contact with.

4.1.10.2 Power Efficiency Standards

The same organization which defines the standards for safety in power, also lay out standards for power efficiency. These standards define five levels of power efficiency for power supplies. These levels are I, II, III, IV, and V. The power supply for the Garaginator falls within the level II category. This category of power supply efficiency states that “no criteria was ever established” [4] for power supply efficiency. While power supply efficiency can be an important standard to take into account, it is not seen as highly applicable to the power supply unit for the Garaginator project.

4.2 Ethical and Safety Constraints

With our project, there comes some constraints since users will be directly interacting with some of the hardware. Fear of information being stolen is a huge factor that we must address. That is why we are taking a very strict approach when it comes to the transfer of data, we want to ensure our users that we respect their privacy and will ensure that none of their information falls into the wrong hands when using our system.

As for safety constraints, there are very little when it comes to our project. The only safety constraint we have run into is from the installation of our garage system. Through our design, we will be connecting sensors to our microcontroller via wires across a good amount of space inside the garage, but to fix this issue we will be using standard electrical tubing as well as wall mounts to ensure that pedestrians walking about the garage are not in any danger of possibly stepping over any wires with electricity flowing through them.

4.3 Sustainability Constraints

During our initial design brainstorm, we needed to consider how to create a product with longevity. If our product was picked to be installed into a garage system, another factor that would make us better than other products would be how long our product can work without needed some type of maintenance.

This would lower costs in the long term, which is a huge factor for our project. We are still in our initial development phase of the product, so we do not necessarily know what sustainability issues we might run into at this time. But by the end of fall 2017 we will have found all issues and attempted to solve all of them.

4.4 Manufacturing Constraints

Our project has very few manufacturing constraints, our initial idea was to keep the amount of hardware as minimal as possible, in doing so this would drive down cost and maintenance. This in turn also brings down the amount of manufacturing needed.

In theory for our project we would need a system for every entry/exit of a garage, most garages at UCF only have two of each, one on each side of the garage. UCF has 10 parking garages based off their 2017-2018 campus map so we would need around 20 systems in total to fill up every garage on campus. The number of components per system is already substantially low making the cost per system also low.

4.5 Time and Budget Constraints

With our project, we are given two semesters, one being a summer semester (12 weeks) the other being a fall semester (16 weeks). This gives us 28 weeks to come up with an idea, research said idea, and design and create our project. With this amount of time it was difficult to come up with an idea for a project. After the selection of our idea we had to take time to research and flesh our all possible ideas of designing our project.

Our budget is limited to the amount we raise on our go fund me site as well as some personal money we will be using if needed. Our total cost is based off how many garages will implement our system. The cost scales per garage and only per garage. Our cost to implement a prototype will be based off implementing half a system for a garage.

Since most garages especially as UCF contain two exits and entries we decided to implement our garage system for only one exit and entry, but the project we designed in such a way to where we only need to add more hardware as well as program that specific hardware with some slight differences of others, such as the garage id of the garage the system is being implemented in. This brings our

prototype costs down significantly, so we can show proof of concept without breaking the break.

Cost of manufacturing/installation would be the same per garage, if every garage had the same amount of entries and exits. This simplifies overhead that could occur if the garages being implemented with our system consist of different amount of parking spaces, because our system is not based on the amount of spaces it is based on the amount of entries and exits. But we will still need certain information from every garage such as the amount of parking spaces per floor. But this does not interfere with the amount of actual hardware that needs to be bought, programmed, and installed.

As well as the cost of hardware, there will be continuing cost of powering the system as well as communication cost between the system and the web server. This could be via WIFI or ethernet depending on the owner of the garage. We currently do not have a timeframe of how long our sensors/microcontrollers will last without needing any type of maintenance and what that said maintenance would cost.

4.6 Environmental Constraints

It is fairly obvious that the environment will not be significantly be altered as a consequence of the Garaginator, but the environment in which the Garaginator exists will almost certainly affect the design of the Garaginator.

The Garaginator will of course be located inside a parking garage, which is obviously filled with cars. Car must be taken that any wires used by the system are not in locations that are vulnerable to being ripped by car tires. If wires must be run through such an area, they must have adequate protection to prevent them from being destroyed. This constraint is important to keep in mind with our sensors, since they will likely be connected to a kiosk through a hard wire interface.

Parking garages are also notorious for poor wireless reception, due to their all concrete and steel construction. If any wireless transmission techniques are used, care must be taken that they do not need to pass far through the concrete or steel. This constraint effectively rules out any kind of internet connection method that requires a wireless transmission be read which originates from outside the garage, as it will almost certainly not penetrate deep enough.

4.7 Economic Constraints

As previously discussed, we have determined that the primary reason that previous solutions have failed was due to costing too much. Therefore, finding a low cost solution shall be of the utmost importance of this project. Any system which attempts to solve this problem may have to have some sort of hardware at each parking spot, a requirement which becomes costly when there are hundreds

of spots in a single garage. Group 5 must design a system which places minimal hardware at each parking spot, while still achieving its goals.

5 Software Design

In this section, we will be discussing the design of our project. From all the software chosen, down to the exact sensors we chose, with their schematics.

5.1 Server Requirements

To host our database and web server application we require certain specifications to run the applications we want to run on the server. Such as NodeJS, which works well off a single core but needs a decent amount of RAM as well as good I/O speeds. Also, we would run MySQL off the same server to save money, but thankfully MySQL is like NodeJS when it comes to server requirements. For our project, we will be using a virtual machine hosting service owned by a company called Digital Ocean. The current server specifications we will be getting are listed below.

- 4 GB Memory
- 2 Virtual CPU cores
- 60 GB Solid State Disk
- 4 Terabytes of network transfer

These specifications meet the requirements of our project, by allowing us to host NodeJS and MySQL. Which lets us lay the foundation of our project and allow us to develop on it. Using MySQL, we will be able to store parking information such as location, current number of vehicles in the garage, user credentials. NodeJS will allow us to create a “back end” server that won’t be seen by users but will handle all requests coming from both the garage system and from mobile systems.

5.2 Mobile Application

We have chosen to develop the mobile application using the Ionic Framework, because of its multiplatform development as well as the use of common languages such as HTML, CSS, and JavaScript/TypeScript. All of these features have allowed us to create a simple elegant design of our mobile application with clean/crisp response time.

The Mobile application has been designed to be simple, it completes the same tasks as the garage system as well as adds new features. Such as easier ways of storing a user’s parked car location. It will also show users what garages are currently full, as well as the exact number of cars currently in the garage vs the number of spots that garage contains. It will also be able to tell users what garages are the nearest to the building which their class is in.

First a user will need to enter their credentials to use the application, if the user does not have an account there will be an option to create one. All information sent

between the mobile device to our web server will be using AES encryption. Including the initial account creation information. The credentials will match the same credentials used in our garage system. There will be a user ID as well as a 4-6-digit passcode that a user will need to enter to see our system. The mobile system will also have the ability to remember the user's credentials.

To store a user's parked car location, we will have two different input procedures that is up to the user to use. First will be taking a picture of a QR code that will be placed on every spot in a garage, a user will be able to scan the QR code using their smartphone. The other procedure will be the exact same to the manual input of the garage system, the user will in enter their location manually by entering the garage, floor, and numbered spot they are currently located in.

There will also be a bird's eye view picture of the garage they selected to store the location of their parked car, allowing the user to see the number layout of the spots to allow them to find their specific spot easier if they take the manual input route.

Users will also be able to access the garage tab which shows each garage and the current number of cars located in the garage vs the number of spots that garage has. From this tab, it will also allow users to filter the closest garages to certain class buildings so the users can see the current availability of the most convenient garage. Lastly, we would like to implement a Day parking pass system to be used through the application, that will allow the user to store their vehicles information instead of having to use the hardware system located in the current garages.

5.2.1 Mobile Application Interface

Since Ionic is a multiplatform development framework, it allows us to follow one interface for all different platforms using HTML, and CSS. This has allowed us to create a lookalike interface for each platform. During the initial creation of the login page you can see the changes you have made in your code and how your current app looks like in all three platforms simultaneously.

Other features of the mobile application interface will include basic swipe features to switch between tabs once a user has logged in. As well as intuitive user input choices for when the user wants to store their location. Also, a user will be able to start the saving location process from any tab they are currently located on.

5.2.2 Administrative Tasks

We do not have any administrator users, we designed the application so that a user cannot manipulate the system unless they have direct access to our database management system hosted on our virtual machine. Administrators will be able to automatically "close" garages using the database, as well as manually delete cars that are no longer parked in their saved location if needed, but this should not be needed often because the web server has been designed to handle this issue. Administrators will need to be trained in deleting rows in a DBMS, specifically MySQL.

5.3 Garage System Design

Our garage system will implement photodiode sensors in order to track cars entering and leaving the garage. These sensors will be connected to a gateway microcontroller (Kiosk) that will act as a hub between the sensors the LCD Display and the web server. The Gateway microcontroller will receive signals from the photodiodes which will then set off an interrupt that will start our C program to then send information that a new car has entered or exited the garage to the web server.

The other side of the garage system is the user interface side, where users will be able to store and retrieve information of their vehicles location. They will either be able to enter their user id and passcode, or swipe their key card (UCF ID) in order to access the system. From there they choose to store or retrieve their vehicles information, to store the information they will need to input the garages id, floor number, and spot number. Users will be given a guide on the spot numbering of the garages using a picture located next to the LCD Display system.

Below in Figure 12 a flow chart we have created that helps explains our garage system using visuals.

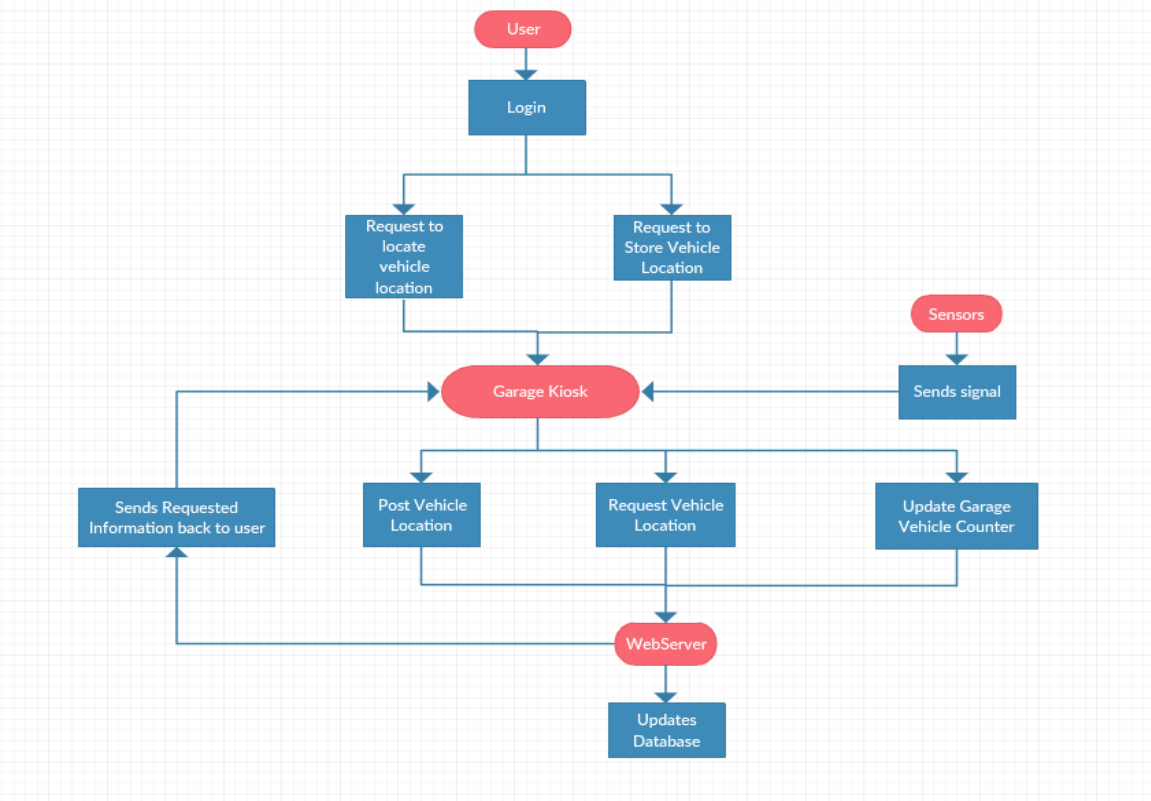


Figure 12 – Kiosk Functional Flowchart

5.4 UML Diagrams

This section contains our final UML Diagrams for our mobile system, as well as our garage system. Unified Modeling Language is a development modeling technique that is good for creating visuals of the design of systems.

5.4.1 Mobile Use Case Diagram

Below in Figure 13 is our use case diagram for our mobile system, it outlines how a user will be accessing our application and what steps they can take once they access it. They will first login, then be taken to the garage availability page, and be given options to filter by class buildings or store the vehicles location, as well as pay for a day parking pass. If a user does not have an account, they will be able to create a new one.

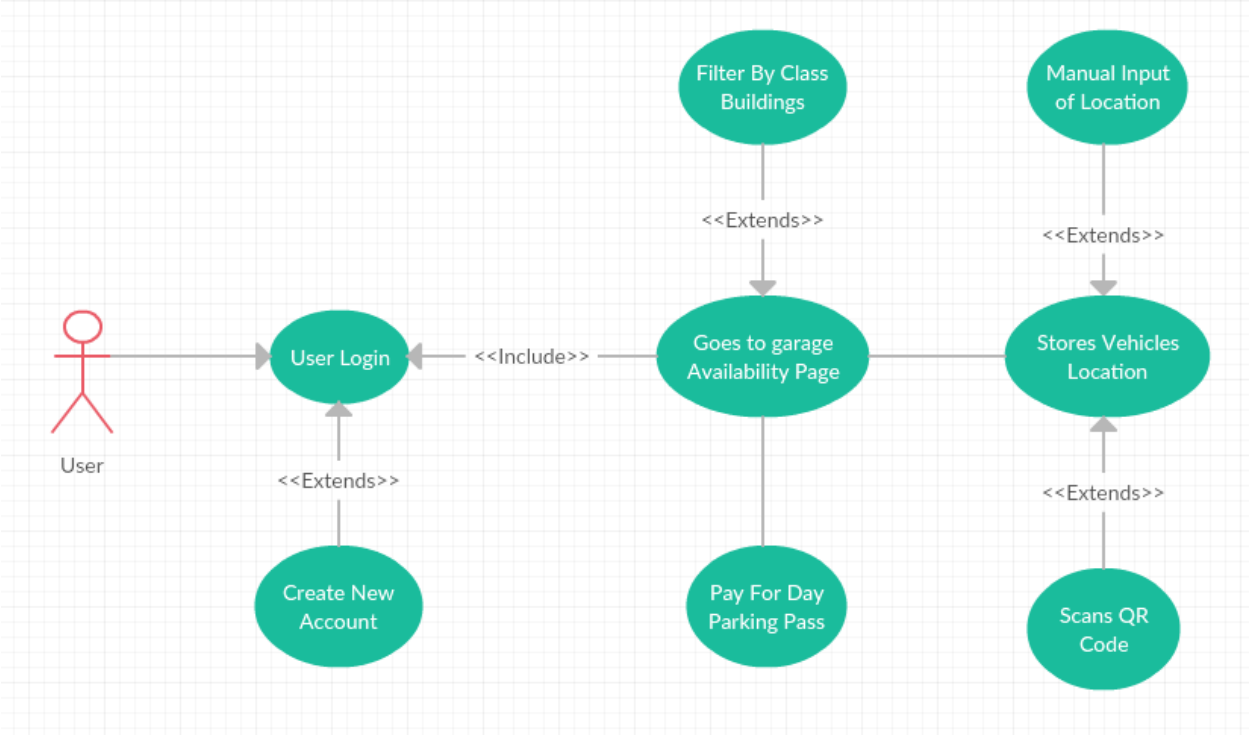


Figure 13 – Use case diagram of Mobile system

5.4.2 Garage System Use Case Diagram

Below in Figure 14 is the use case diagram for our garage based system. It describes how a user will access the systems using the keypad, or by swiping their UCF ID card for our specific UCF garage system. Users will not be able to create an account using the garage system, they will need to download the mobile application, or visit the website on which we will be hosting our entire project, which will also contain a system for creating user accounts. Users will then be given two options, one to find their vehicles location, the other to store it.

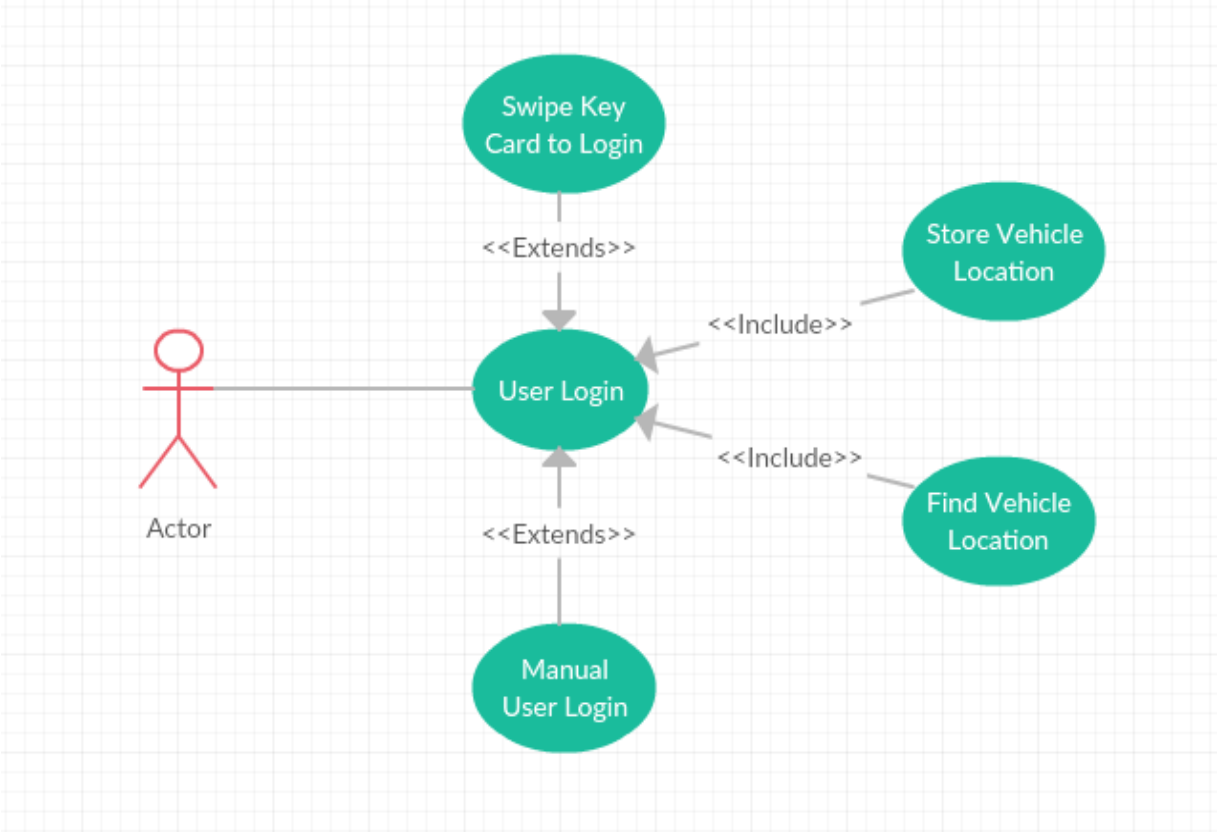


Figure 14 – Use case diagram of Kiosk System

5.5 Database Design

For our database design, we decided to choose MySQL for our purpose. It is the most familiar DBMS to our group, as well as a free one which goes along with our design philosophy to keep cost as low as possible. For our database, we only need to store user credentials, parking locations, some garage information, and day parking pass information.

The database will be updated via the web server handling requests to send/update/delete information in the database. Access to the database must be secure since it contains confidential information for users. The database will contain a counter column for each garage that will be incremented and decremented based on the sensors that send signals to our garage kiosk. For our users, we are only saving their unique id and passcode. For the day parking pass, we must save the user's vehicle make, model, color, year, and license plate number.

In Figure 15 we show the design of our schema that will be implemented in MySQL. With the exact table names, columns, and rows that will be used. The lines between the tables describe the foreign key relationship between the tables, such as for Vehicle_location it uses Account_id which is a Primary Key but also a foreign

key from the Users table, so the Vehicle_location cannot have an entry of a Account_id that is not in the Users table.

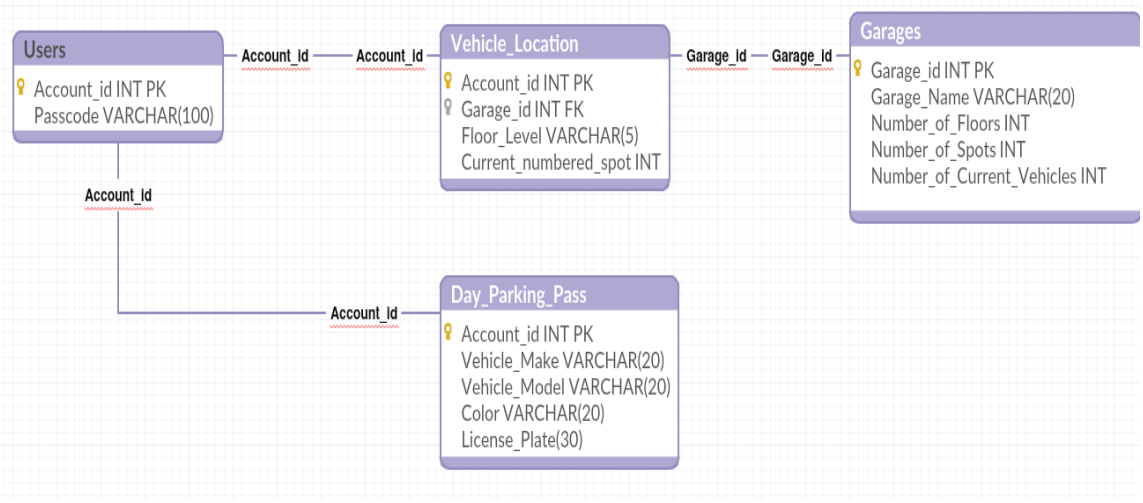


Figure 15 – Final Database Tables

Table 17 – Users table contains each of our user's credential information. Mainly just their username and passcode. But this table can be made to hold more user information if need be in the future. It uses Account_id as the primary key of the table to create uniqueness between users.

Column	Data Type	Description
Account_id	INT	This table uses this column as its primary key to bring uniqueness to each entry
Passcode	VARCHAR	This column will hold the passcode of every user, but it will not hold their exact 4-6-digit passcode, it will hold the hashed and salted passcode.

Table 17 – Users Database Table

Table 18 – Vehicle_Location Table, contains the information of each of our users parked cars location, using our users Account_id as a primary key to ensure uniqueness of every entry in our table.

Column	Data Type	Description
Account_id	INT	This is the primary key of this table, allowing us to create uniqueness between users, by using their user ID.
Garage_id	INT	This is a foreign key which will be used to reference what garage the user has stored their vehicle.
Floor_Level	VARCHAR	This is the current floor level the user's vehicle is located at
Current_Numbered_spot	INT	This is the value of the current spot that the user's vehicle is currently parked at.

Table 18 – Vehicle_location Database Table

Table 19 – Garages table contains all the information of every garage that will be made in the system. Each garage will contain a unqiue id, a name, number of floors, number of spots, and number of current vehicles. All of this information will describe one specific garage and this is how we will be tracking how many current vehicles are located in a specific garage.

Table 19

Column	Data Type	Description
Garage_id	INT	This is the primary key which will be used to give each garage a unique id to differentiate them.
Garage_Name	VARCHAR	This is used for the name of the garage
Number_of_Floors	INT	This is the number of floors this garage contains
Number_of_Spots	INT	The is the total number of spots that this garage contains
Number_of_Current_Vehicles	INT	This is the total number of current vehicles within this garage, this will be updated constanly.

Table 20 – Day_Parking_Pass table will be used for our day parking pass system, this is where we will hold information on users that pay for a day parking pass.

Column	Data Type	Description
Account_id	INT	This is the user id which will be used as the primary key of this table.
Vehicle_Make	VARCHAR	This column will hold the make of the vehicle of the user purchasing a day pass
Vehicle_Model	VARCHAR	This column will hold the model of the vehicle of the user
Vehicle_Color	VARCHAR	This will hold the color of the user's vehicle
License_plate	VARCHAR	This will hold the license plate number of the user's vehicle

Table 20 – Day_Parking_Pass Database Table

6 Hardware Design

The Garaginator system shall include at least one kiosk located inside of the garage, where users can input the spot where their car is, and find out the location of their vehicle. The kiosk will also be connected to the sensors at the entry way, and shall use the information to update the counter of cars inside the garage. Lastly, the kiosk shall be connected to the internet and shall be able to send and receive data to and from the application server. This section describes the hardware we intend to use to accomplish all of this.

The Kiosk is the heart of the hardware of this project. It makes up a majority of the hardware costs of the system.

6.1 Hardware Components

Figure 16 below shows all the hardware components which are to be used in this project. At this point in time, every hardware component has been received and tested and every component is shown in the figure below.

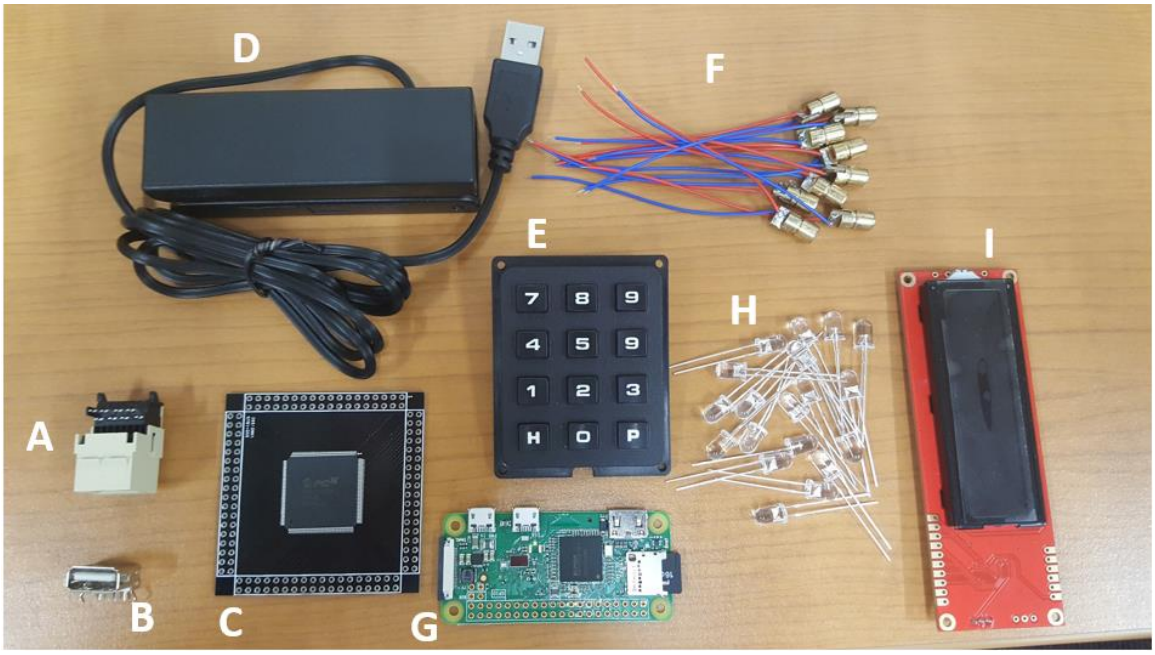


Figure 16 - Selected Hardware Components

The figure illustrates: **A** Ethernet port, **B** USB port, **C** PIC32 microcontroller, **D** magnetic card reader, **E** keypad, **F** laser dot diodes, **H** photodiodes, **I** LCD display. All the components will be integrated onto the PCB and their various signals fed into either the PIC32 microcontroller or the Raspberry Pi Zero W.

6.2 PCB Schematic

The following figure below shows the general schematic of the hardware and their connections. On the top right of the schematic is the Ethernet PHY controller labeled ENC28. To the left of the controller are all the resistors, capacitors, crystal, and inductor needed to allow for normal functionality of the chip. On the top left of the schematic is the DC power barrel jack labeled DCj0202. The three boxes below the DC power barrel jack are the six, three, and four pin headers for GPIO connections to the keypad, LCD display, and photoresistor sensors. On the mid-right of the schematic is the ethernet jack, and below that is the female USB 2.0 header. The large square in the center of the schematic labeled PIC32MX is the microcontroller.

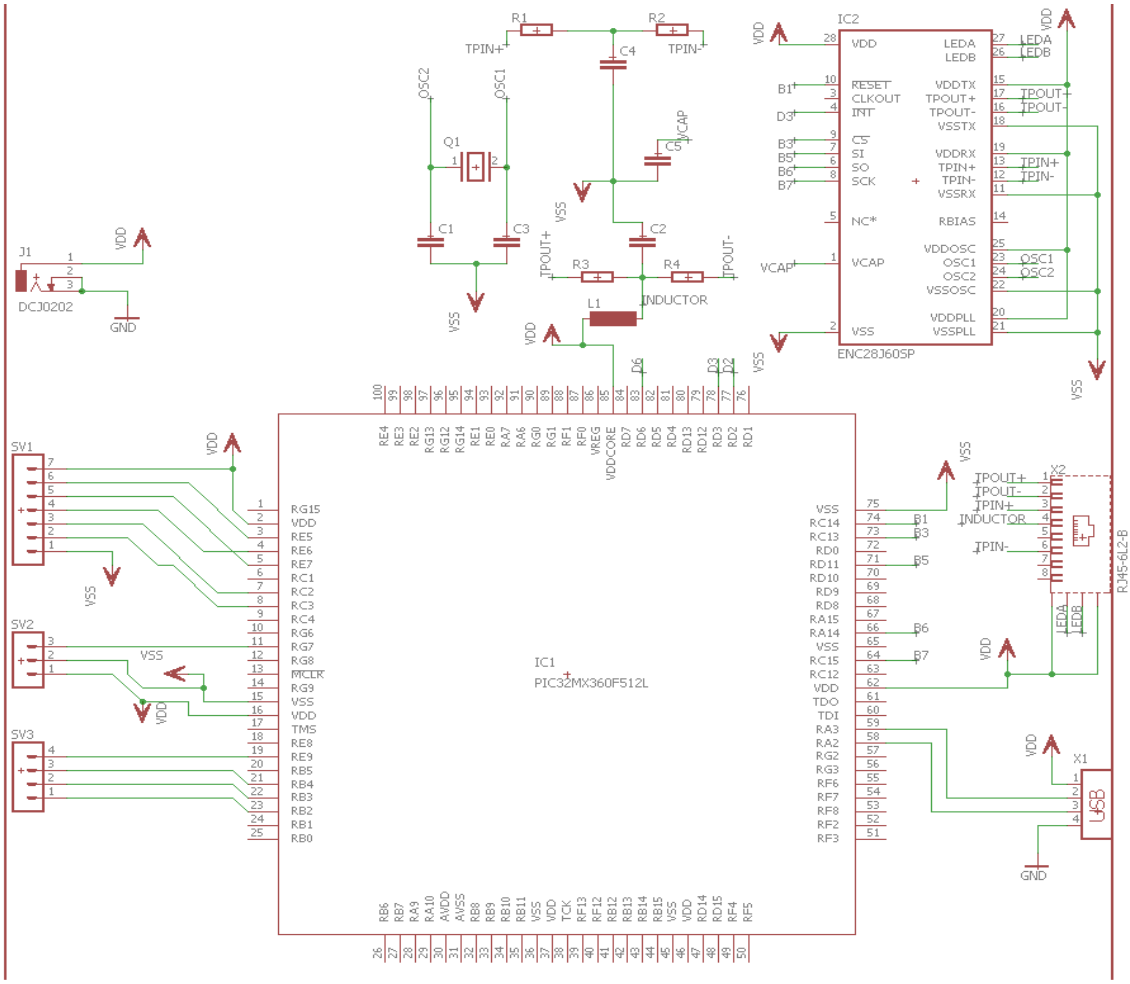


Figure 17 - PCB Schematic

6.3 Hardware Block Diagram

Figure 18 lays out how all of the hardware will interact with each other. As can be seen, the power system supplies power to the optical sensor system, the peripherals, and the microcontroller. The optical sensors send information received about a vehicle entering the garage to the microcontroller. The peripherals send information that a user inputs to the microcontroller as well. The microcontroller then sends this information to the software, at which point the software runs code pertinent to the signals received and sends updating information to the microcontroller.

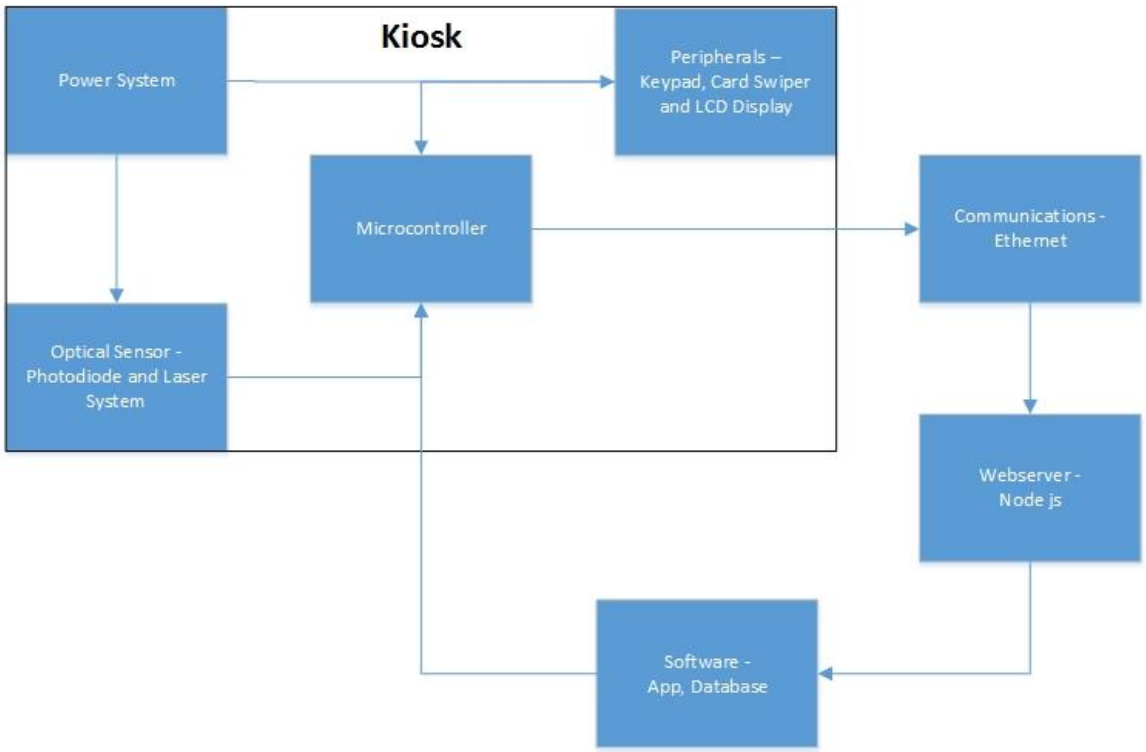


Figure 18 - Hardware Block Diagram

6.4 Sensor Design

The final design for the sensor is laid out in full detail in the figure 19 below. The laser dot diode is powered by a source of five volts. The laser then spans a ten-foot distance to activate the photodiode. This photodiode is normally on and emitting a high signal, but when a car passes and breaks the beam of the laser shining upon the photodiode, the photodiode turns off and goes low.

The signal from the photodiode is fed to a non-inverting amplifying circuit, so that the signal may be of use. The signal from the photodiode is very small, so the op-

amp circuit remedies this shortcoming of the photodiode. The resistor values are chosen so that a high gain is achieved, though this gain may not cause the signal from the photodiode to exceed five volts, due to the five volts that the rails of the op-amp are receiving.

The output of the op-amp is then sent to the microcontroller, so that the microcontroller may process the received signal and increment or decrement the “vehicles within the garage” counter appropriately.

In order to avoid false positives, two of these optical sensors will be fed into an AND gate. Since vehicles are much longer than humans or other typical causes of false positives, the two optical sensors will both be activated at some point during the length of the vehicle. When both of them are activated the AND gate will go high and send the signal to the microcontroller that a vehicle has passed. If a human or other similar being passes beneath the optical sensors, it is highly unlikely that both optical sensors would be triggered at once. This minimizes the amount of false positives received by the overall system.

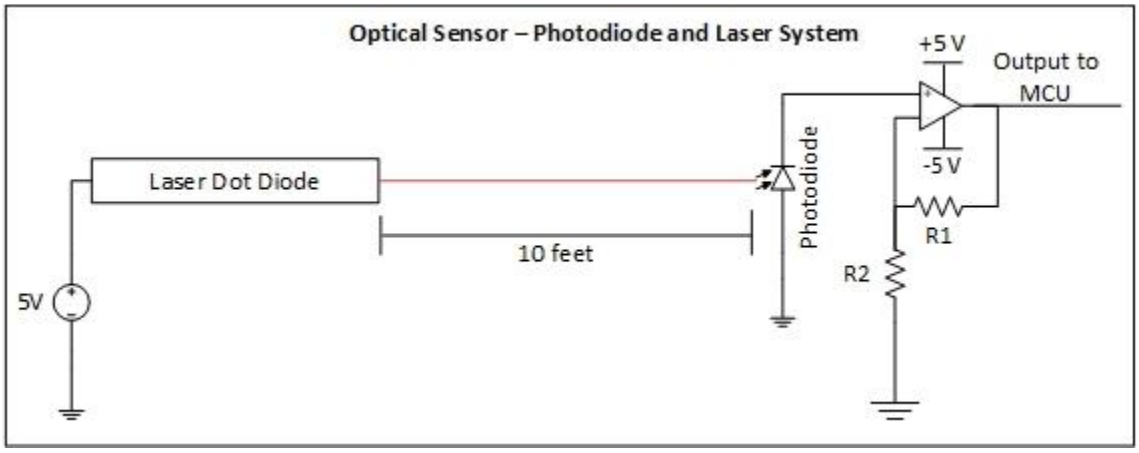


Figure 19 - Optical Sensor Circuit

7 Prototype

To ensure the selected components can operate and implement all the expected functionality, the separate systems were implemented by Group 5 using breadboards or other temporary electronics fixtures.

In order to be able to evaluate various parts without committing to a particular microcontroller, various different MCU's were used as testing equipment, including an Arduino and an MSP430 Launchpad.

These Microcontrollers are not strictly part of the project, but rather were what we had readily available for use at the time. The separate systems and their breadboard prototypes are described in this section.

7.1 Kiosk User Interface System

There are a few major functions of the kiosk, as visible to the user. The user should be able to input information, and the display should be able to show information. A sample interface was created using an Arduino UNO. An Arduino was used because it was readily available and because it has significant software libraries freely available, making writing code very easy.

The test began by connecting the first three pins of the keypad to pins 2,3, and 4, of the Arduino, and the four horizontal pins were connected to pins 5,6,7 and 8. Next the RX pin of the display was connected to pin 1on the Arduino (UART TX), and power was supplied from the +5V pin of the Arduino.

Next a simple implementation of the user interface was created. The Keypad library, which are freely available, made this interface very easy to make. When the user presses on any key, that key shows up on the display, and is written to the USB serial and is displayed on the host computer, if it is attached.

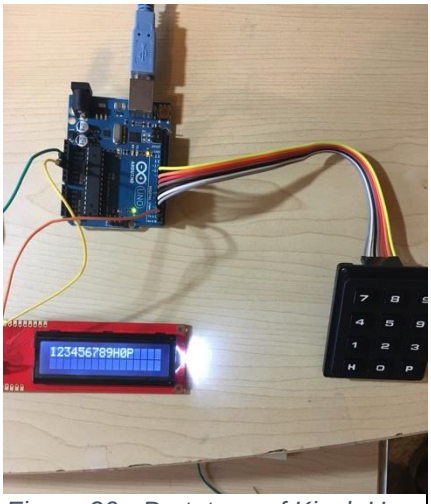


Figure 20 - Prototype of Kiosk User Interface

In all the code used was only 36 lines, including whitespace and comments. Figure 20 shows the result of the test, after having pressed the numbers 1-9, then H, 0, P:

7.2 Optical Sensor System

The main goal of the optical sensor prototype was to test both the laser and the photodiode in conjunction with the pi. To this end, the laser was supplied 5 volts and then shone upon the photodiode. The photodiode was connected to a resistor so that it would have a load that could be measured. The pi was also tested relative to the optical sensors, to see if the pi could receive a signal from the sensors.

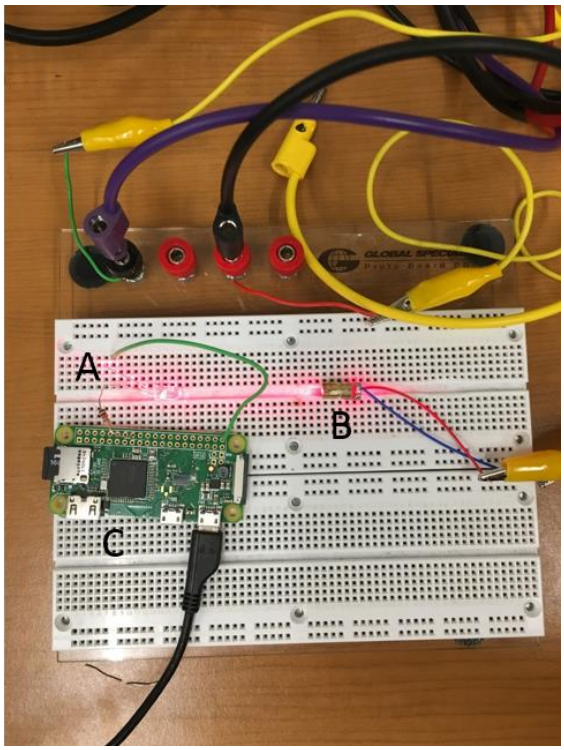


Figure 21 - OPTICAL SENSOR PROTOTYPE

The figure illustrates: **A** photodiode and resistor, **B** laser dot diode, **C** the raspberry pi.

The oscilloscope was also used to read the response of the photodiode when power was supplied to the laser dot diode in the form of a square wave operating at 32 kHz. The 32 kHz was chosen to emulate cars breaking the beam of the laser dot diode so as to emulate a more realistic signal.

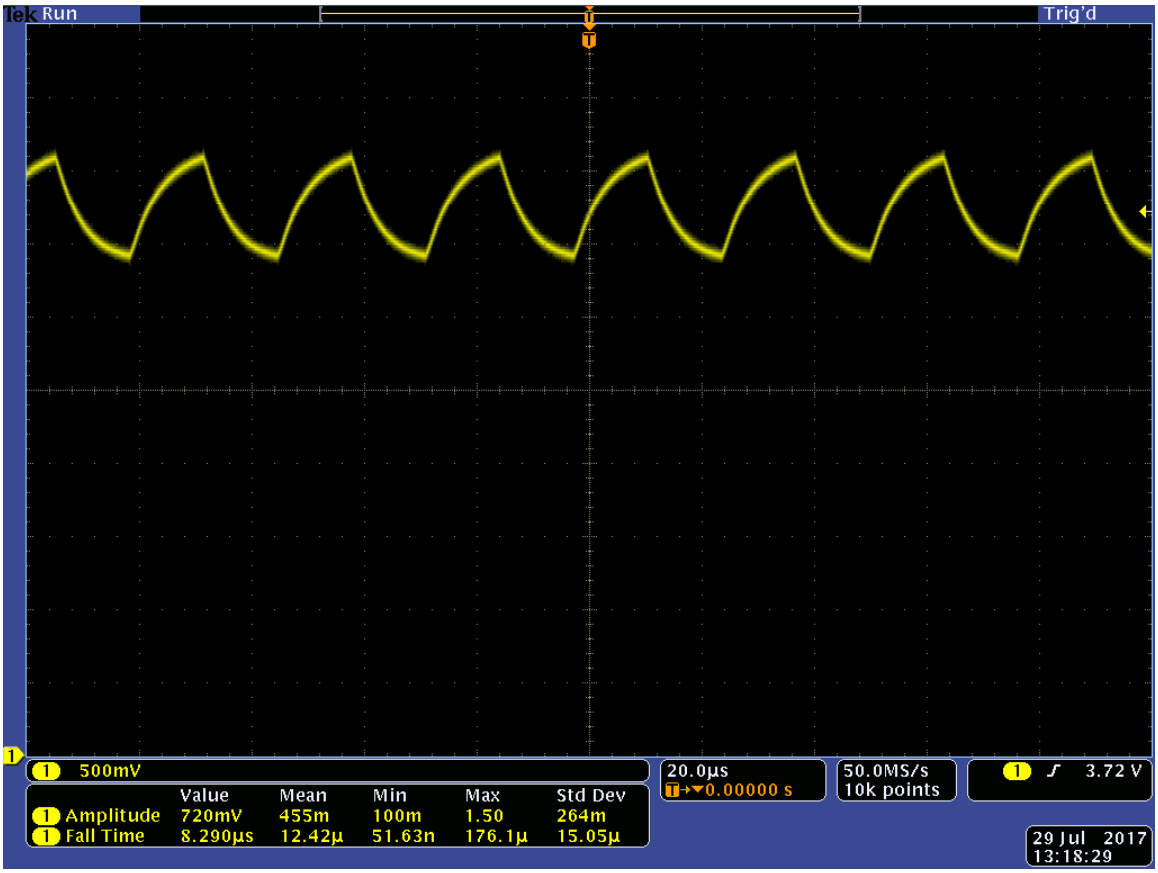


Figure 22 - PHOTODIODE OUTPPUT AT 32 KHZ INPUT

7.3 Printed Circuit Board

Cadsoft's EAGLE PCB program is the chosen PCB design software for its superior user interface, reasonable usage costs, and the ability to export standard gerber files for PCB manufacturing. Below shows the PCB design for the Kiosk's MCU and peripheral interfaces.

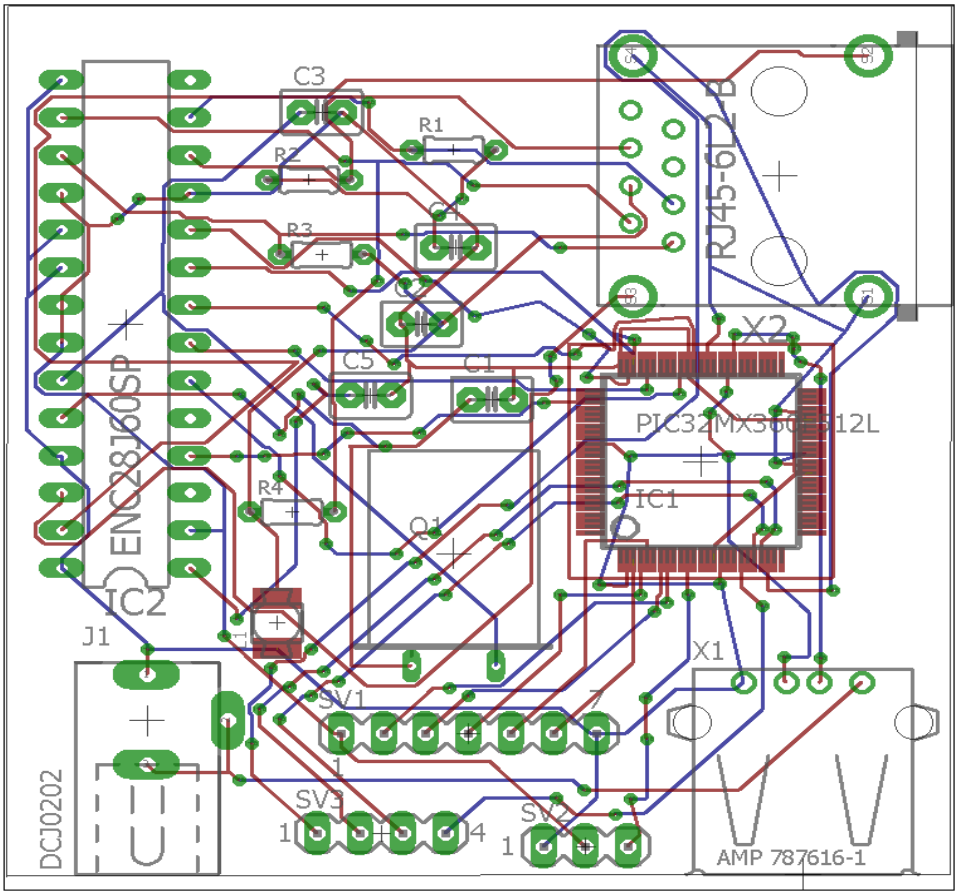


Figure 23 - Print Circuit Board Design (in EAGLE)

In figure 23 above you can see that on the PCB there are a number of devices and connectors which will attached. In the top right corner the connection labeled RJ45 is the standard ethernet jack which will be the medium which network data in transferred from the ethernet PHY controller on the far left of the board labeled ENC28.

This Ethernet controller is directly connected to the microcontroller labeled PIC32MX to the middle far right of the figure. In order for the two processors to work correctly a number of resistors, capacitors, an inductor, and a crystal oscillator are needed.

These can be seen in the center and mid-top of the figure. On the bottom of the PCB from left to right there is a DC barrel power jack, six pin GPIO header, four pin GPIO header, three pin GPIO header, and a female USB header.

7.4 Software

In this section, we discuss the prototype software that will be written during the fall semester of Senior Design 2 to demonstrate our idea. We talk about the web server, mobile application, as well as the database that will be created.

7.4.1 Database Prototype

The following pictures show our database tables created in MySQL, with some fake data inserted. This is figure 24 which shows our garages table with Garage “A” as the only entry. This will be used to create and store different garages with their specific information.

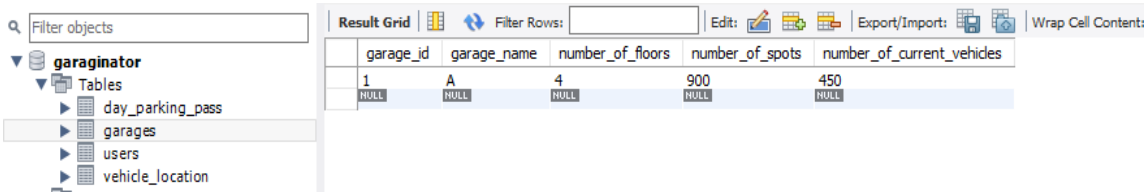


Figure 24 – Garages Table

The following figure 25 shows table “Users” with one entry of fake data. This table is being used to store basic user information.

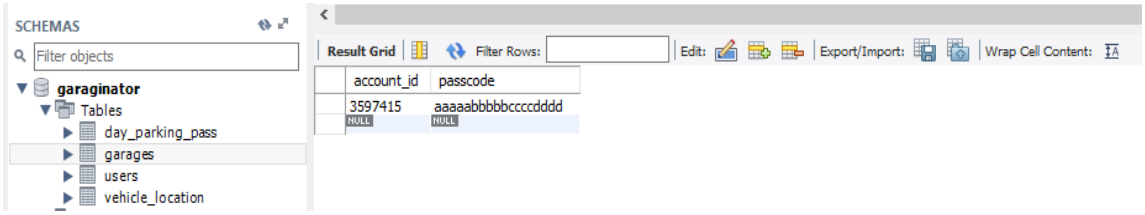


Figure 25 – Users Table

Below is figure 26 which shows the table “vehicle_locations” which will be the table that will hold user’s personal vehicle locations that they store and retrieve at their discretion. This table allows us as developers to not worry about overlapping users putting their vehicle in the same spot as another, because that will fall on the user, it will not affect the system in anyway.

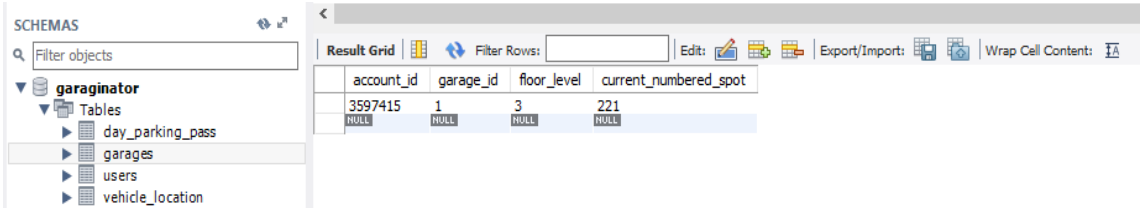


Figure 26 – Vehicle_location Table

Figure 27 which is the “day_parking_pass” table, holds user’s specific vehicle information such as the make and model, as well as their license plate. All this data is fake data created by our team.

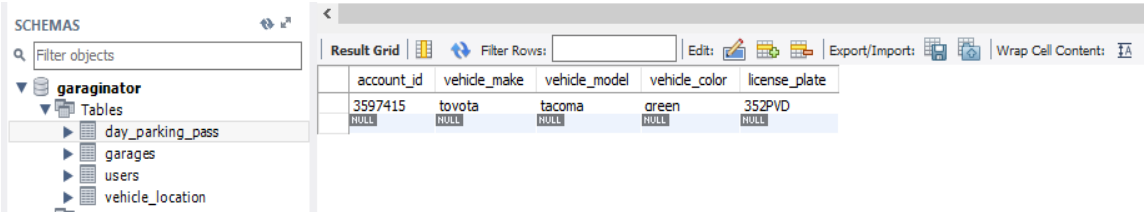


Figure 27 – Day_Parking_Pass Table

7.4.2 Web Server

As we have said in previous sections, we have chosen the NodeJS platform to run our web server, due to its prowess in handling client connections simultaneously. The web server will demonstrate its ability to handle requests coming from a single client to complete retrieval requests, storage requests, and can do this from both the kiosk system as well as our mobile system.

The actual software will be written so that if scalability was to be implemented it would handle all the new clients in the same fashion. This web server will be hosted on a cloud platform, most likely on digital ocean. To demonstrate our web server, we will need to test our mobile platform, as well as check our database for correct updates.

7.4.3 Mobile System

During senior design 2 we will be creating a small mobile application using the Ionic framework. To demonstrate it we will have the application on both an IOS and Android phone, and demonstrate how a user will access and use it.

The mobile application will allow a user to create an account with our system, as well as retrieve and store their vehicles location, using a manual input system, as well as a QR code system where they will be able to take a picture of a QR code located in the spot of their vehicle which then saves their vehicles location. Once a user has added their vehicle location using the mobile application they will be able to retrieve it using the kiosk and vice versa.

Figure 28 below displays the prototype of the log-in page of the mobile system. The figure displays the prototype, not only in iOS, but Android and Windows as well. The log-in page prototype was created using the Ionic framework, as well as basic HTML and CSS. As of yet, the prototype does not have functionality. It is merely a representation of a possible solution.

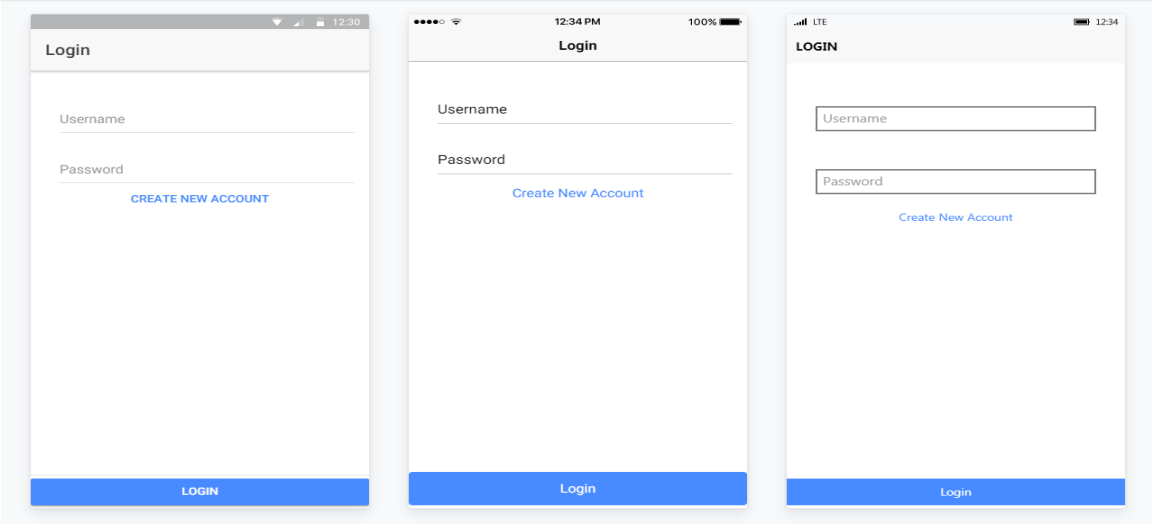


Figure 28 – Log-in Prototype

8 Testing Plan

To make a reliable product it is crucial to test it under different conditions. However, to find the cause of a problem you must be able to test the individual components at a more fine-grained level, to isolate the cause of the problem.

8.1 Testing the Display

The first test, which ensures that the display is operational, is to power the display with a five volt supply. The tester should observe that the backlight turns on, and a splash screen is displayed on the LCD. This test ensures that the display is not damaged or defective.

The next test is to ensure that the display shows all characters and symbols as expected. To perform this test the display must be connected to any serial port, either from a computer via a USB-Serial adapter or from a spare microcontroller. Once connected, the tester should send the ASCII values of all the printable characters (decimal values 24 - 126) to the display. A successful result is noted by all of the corresponding characters shown on the display, with a clear depiction of each character. This test is useful for discovering corruption of the memory which contains the font for the display, which is editable, and as a result could become corrupted.

Next the test administrator should test the display control codes. This is done by sending all control codes supported by the display and verifying that they alter the displays behavior as expected.

8.2 Testing the Keypad System

Being a simple device, the keypad does not require a large amount of testing. To test that it is operational, the tester should connect an ohmmeter to the first and fourth pins on the package. After pressing the first button, the ohmmeter should change from reading an Open Loop to approximately zero ohms. This process is repeated for each button by moving the ohmmeter to the corresponding horizontal and vertical pin.

Alternatively, a testing rig can be made out of LEDs, some resistors, a power supply and a bread board. The first three pins (starting from the left) are connected in series with a resistor and a green LED. The last four are connected in series also with a resistor and a red LED. Figure 9 shows the breadboard circuit I used. Note: I used an Arduino for a regulated 5V power supply, because it is what I had on hand. Any power supply with suitable voltage and current parameters can be substituted.

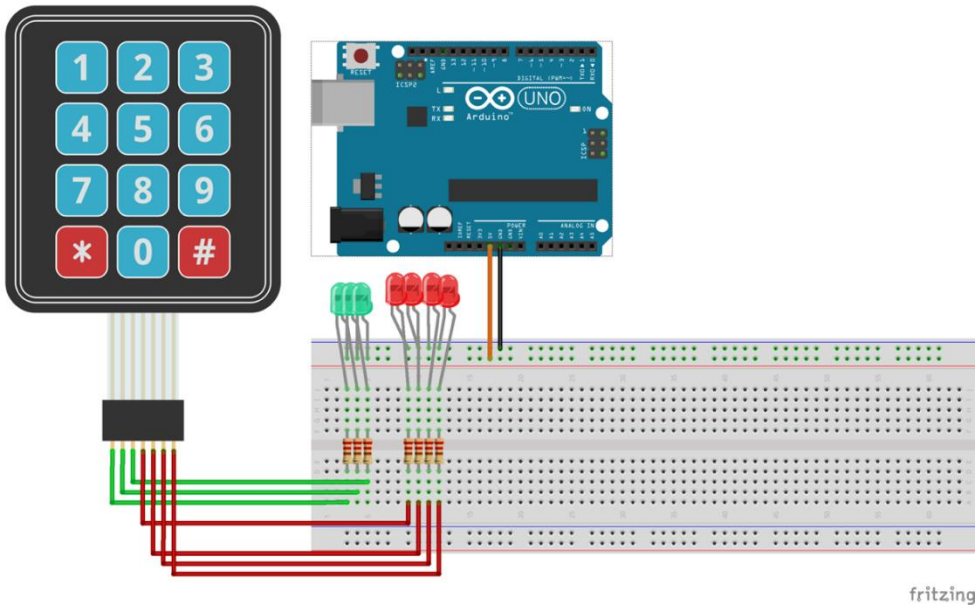


Figure 29 – Keypad Breadboard Simulation

8.3 Testing the Magnetic Card Reader

Since the magnetic card reader is an internally complex device, it is only useful to test it to the extent that it can be repaired. To test it, first connect it to a PC through USB, and a card reading software should be opened. If the reader is able to correctly read a UCF ID then the test is a pass.

8.4 Testing the Sensors

In order to test the sensor in the system as a whole, the sensor must be tested individually first. To this end, the sensor will first be built in an isolated circuit, which is to say that the sensor will be isolated from the other components of the system at large. A load resistor will be connected to the photodiodes so that the photodiode has a load to discharge through.

In order to generate voltage, a laser will be shone upon on the photodiode and the load resistor will be measured with a volt meter to verify that a voltage has been generated by the photodiode due to the laser. The photodiode will have to be either in a dark room or some other form of dark housing with only the light shining upon it. No other lights may be shining upon the photodiode because these lights may cause the photodiode to produce a voltage and thus it would be impossible to discern whether the ambient lights were producing a voltage or if the laser were producing a voltage.

Once it has been determined that the photodiode operates properly, as well as the laser that accompanies it, then the photodiode and the laser will be moved apart from each other and the voltage across the resistor load will be measured so that a graph of voltage generated by the photodiode can be plotted against the distance the photodiode is from the laser.

Before the sensor can be tested with the microcontroller, as an extra precaution, a circuit simulation software will be utilized in order to simulate a circuit for the sensor. A physical circuit which mirrors the simulated circuit shall also be built. The simulated circuit and the physical circuit shall be checked against each other to make sure all the various voltages, currents, and resistances are correct so as to not damage anything within the microcontroller. It is then that the sensor may be integrated with the microcontroller.

As for the laser, it shall be fed a low voltage at first and this voltage shall be increased until the maximum voltage that the laser is rated for has been reached. As this voltage is being supplied to the laser, the laser shall be shone upon the photodiode so as to see the effects of the intensity of the laser upon the photodiode. This also serves as a test of the photodiodes sensitivity. A graph of laser intensity versus photodiode voltage shall be generated based upon this information.

8.5 Testing the Breakout Board

For this project, a stand-alone PIC microcontroller has been chosen as the best candidate for this project as the heart running every peripheral needed for the garage sensor and kiosk. Two slightly different PIC microcontrollers have been ordered to be tested, and after testing is done the group will settle on which of the two to be used in the final project. These two microcontrollers are the PIC32MZ2048EFH144 and the PIC32MX675F512H MCUs. The first of will be

referred to as “PIC32MZ144” and the latter “PIC32MX64” where 144 and 64 refer to how many pins they each have. Testing of either of these MCUs will have to be done with the help of a breakout board because both MCUs have very low-profile pins that do not insert into a breadboard, and they are very close together (less than a mm). So, a breakout board will need to be developed in order to make testing easier and feasible

Breakout boards are used when connections are too close and shallow on devices such as low profile MCUs to allow easier access to the pins and connections. They are PCBs that are custom made to fit a device and spread out its connections by connecting the device’s pins to pins that are spread out over the breakout board.

The breakout board will have to be developed in a program that can create what are called gerber files. Gerber files are the standard files which PCB’s (printed circuit board) are made and saved with. There are a number programs which can be chosen to create such files. These programs include; Altium Designer, CircuitMaker, Eagle, DipTrace, Proteus, KiCAD, DesignSpark, and PCBE. All of these programs are very similar but have their slight variations from each other. the program called Eagle has been decided as the PCB design program the group will use for all PCB design, for its user-friendly interface and the fact that it is free to use.

The design of the breakout board will take some time to develop and provides great experience for developing the find PCB that will be needed for the final project. It will be needed as soon as possible to begin testing of the two different MCUs in order to decide which one will make the cut.

Once the breakout board is received after about 5 works days of development and a week of shipping, it too will have to be tested to make sure all the connections are correct. This can easily be done by using a multimeter to test the resistance of each connection. As they should only be wires connecting from one end to another the multimeter should display very low resistance (a short) if the connection is right. If the connection is not right, then a huge resistance will be read from the multimeter because it will see the connection as an open circuit, which are theoretically infinite resistance.

Once all the connections are proven to work the MCU can then be soldered onto the breakout board and then used to easily attach the MCU to a breadboard and other devices for testing and debugging.

8.6 Microcontroller

There are a large number of ways the microcontroller(s) can be tested to make sure every function and pin works correctly. To first run the MCU, it will need to be powered by a power supply giving anywhere from 2.3 to 3.6 volts to any of its VDD connectors and its GND connectors attached to ground. It will also require a few voltage regulating capacitors, but no pull-up or pull-down resistors because the MCUs have those built in. If external pull-up or pull-down resistors were used the

voltages would have been halved, causing the MCU to not run correctly. A simple task to test the MCU is to control the tone and timing of a buzzer. It is a simple task that will test a couple GPIO pins and the timers of the MCU to make sure they function as desired.

Once the microcontroller has proven to generally function correctly, the peripherals will need to be tested also to double check that they work. The first of which to check would be the LCD screen. The LCD screen requires 11 GPIO pins to function correctly. These 11 pins will then be connected to the MCU with the help of the breakout board. A power supply from the UCF senior design lab will be used to make sure the correct voltages are supplied. Using the PICKIT3 debugger/programmer to program the microcontroller, a simple “hello world” program will be programmed into the MCU to display the string “hello world” onto the LCD display. After the simple program is done correctly, the next step is to test that every pixel on the LCD monitor works correctly and isn’t burnt out or dead. This is a simple task that can be done by writing a program to turn on every pixel of the LCD and then inspect for anything off or wrong.

Now the USB female connector will need to be connected correctly to the microcontroller. Both MCUs have onboard USB controllers so only an external physical USB connected is needed to be connected to the microcontroller. USB only requires four pins to work properly. One for ground, another for Vcc, and two for data in and data out. D- is pin 36 and D+ is pin 37 on the PIC32MX64 microcontroller, Vcc will be supplied with 5 volts DC and the ground pin will be connected to ground. A Multimeter will be used to check that each of the connections on the USB header are correct and the voltages are stable. This USB header will be used to connect a USB powered card reader that user will use to quickly input their ID to keep track of their car.

Both of the Microcontrollers have built in ethernet controllers with unique MAC’s. So only a physical ethernet port will be needed to be attached to the microcontroller at the correct pins to allow for internet access. There are eight connections that need to be attached from the ethernet port to the MCU’s pins. The wiring is attached to the Ethernet port by essentially jamming the wires into the contact which are held by bladed holders, almost like a vampire tap; where the wire makes contact with the metal leads by the blades cutting through the wire’s insulation to get to the copper wiring.

Each wire and contact within the Ethernet port will have to be checked to make sure a good contact has been made. This can easily be checked with a multimeter by checking for resistance between the two leads of the multimeter. If the resistance is very low then a good contact has been made, as the wire should be read as just a sort. If there is not a good contact made the multimeter should read a huge resistance, mean that there is an open circuit and the wires haven’t made good contact with the leads. This can be fixed by either trying the push the wire into the contact even more, or to try again with another area of the wire and then test the contact again.

8.7 PCB Design Software

There are a few ways to test the chosen printed circuit board software for this project. The first step is to install it onto any of our personal computers and make sure the program is able to start-up without any hiccups. Anything from crashing to freezing or even a bluescreen of death would be a sign that the program is not working correctly on our operating systems. Once it has proven to run on a personal computer running windows version 10, the software will have to up uploaded onto a machine running MAC operating system and prove it is stable on that operating system. If all goes well then the first test is passed that our machine can at least open the program.

Next test is to see how the program loads past projects. As no one in the group has used any sort of PCB program before, we will have to download someone else's finished work in order to test the programs stability when loading a project. The test will first be done with a small project to see how long it has to run and load the project until it is finished. Once that passes, the next step is to see how it does loading a larger project file, maybe even a PCB for a homemade motherboard. The large numbers of variable and objects in this kind of project is sure to push the software and the machine to the limit. Once this test finishes and passes, the program has then proven to be stable loading past projects.

Some of the PCB design software's support the ability for multiple people to edit a circuit diagram at the same time through multiple devices on the internet. To test this feature the program will be installed on multiple computers, then a project will be shared among the users. The users will try to edit the program at the same time and see how stable and reactive the program becomes. If the program does well support just two users, then another will be added for 3 users editing the program at the same time. If the program remains reactive with no hiccups or slow down, then the program will pass this test. Four users will be very unlikely and there will be only one person editing the project at a time, since the group will only be willing to pay for one software license.

8.8 PCB Testing

After the PCB board passes the initial visual check after shipping and handling, the board will then need to be scrutinized further to find if it has working connections. The board will be referenced with the gerber file first used to create the PCB, while using a multimeter to check if all the connections are not dead on arrival. For each checked connection, the multimeter should display a very low resistance as the connection should essentially be a short; as it is just a wire. If the connection displays a high resistance on the multimeter then that displays that either the multimeter is looking at the wrong to pin connections, or the wire is an open circuit with an incomplete connection. If the multimeter is simply that at the wrong two pin locations, this will be checked by referencing the main gerber file by the tester. If it is deemed true that the connections are right, then the only conclusion is that he

printed circuit board is a dud and the next PCB will begin the process of connection checks via the multimeter.

Next the board will need to be checked if all the desired equipment and electronics will be able to fit onto the designed PCB. If any connection is off, then the group will have to decide whether to keep the board and use wiring to make the given connections work, or to scrap the board and make the needed spacing corrections to the PCB gerber file and order a new batch from the same or another manufacturer.

If it is deemed necessary by the group to go further in depth with the testing, then the next test will be and Adjacency test. Which tests if any connections which should not be touching are totally isolated from one another. This test involves using the multimeter again to test for resistances between adjacent nodes and wiring. The resistance on the multimeter should always read a high resistance for nodes that should not be connected to one another. If the resistance reads anything too low to consider the nodes isolated from one another via the insulation, then the PCB will fail and be scrapped.

8.9 Software Testing

We will be testing our software throughout development. We will be using an AGILE approach when it comes to the creation of our software. AGILE is based off a set of principles to follow when developing software. It emphasizes adaptive planning, early delivery, continuous testing and improvement, and it encourages flexible response when it comes to changes in requirements needed for our project. Using this approach, we will be able to develop our software and test while creating our project, allowing us to deliver our software on time and possibly earlier.

We chose AGILE over other approaches such as the “Waterfall” approach because of the better use AGILE brings to our style of development. We will be constantly testing out our software, instead of writing huge amounts and integrating testing during another specific time. Using AGILE allows us to debug off the start, showing us what errors we are running into.

To test our software, we will be performing weekly testing once all our hardware components are set. We will take a bottom to up approach when testing out software, starting with the lower level software we need to create for the microcontrollers, and then testing the higher-level JavaScript we will be writing for the webserver.

8.10 Database Testing

To test the database, we will be performing functional testing. Checking tables after the web server requests have been performed. We check the tables to ensure the correct data has been inputted or updated. We will then do a performance test,

where we will send the server hundreds of requests to update the database to see how well both the web server and the database handle the many requests.

8.11 System Test and Demonstration

The system test will begin by first installing our demonstration system into a UCF parking garage. Once installed we will power on all the hardware components. We will ensure that the web server and the database have already been running at this point. From there we will drive a vehicle through an entrance of the garage, then we will check the database to see if that garages number of current spots counter was updated accordingly.

After this first test we will perform a similar test by making a vehicle leave the garage, and then check the database accordingly. Once this is complete we can ensure our sensors system is working completely. Then we will move onto the kiosk system, where we will have one of our team members manually login and store their vehicles location. Once complete we will double check the database to see if it was updated with his request, then our team member will then try to retrieve their vehicles location using the same kiosk system.

Next, we will test our mobile system, by having a team member open the application on their phone and try to store their vehicles location using the manual input system. We will then test the same system but by switching from the manual input system to having the user take a picture of a QR code located at the spot of their vehicle. Once these tests have been performed we will ask the user to retrieve their vehicles location by using their phone.

Finally, we will ask the user to switch how they store and retrieve their vehicles location. By having them store it using the kiosk system, but then retrieve the location using their phone. This will also be done in the opposite way where they store the location using their phone and retrieve it using the kiosk system.

9 Administrative Content

9.1 Project management

Management is an essential part of a good overall team. Through the first few weeks we have been able to come together as a team using good communication skills to accomplish our tasks. From our very first meeting we have set a schedule of meeting times every Tuesday from 11AM – 1PM to meet and collaborate on what needs to be accomplished. We did this to always meet with each other in person to begin building team chemistry, as well as discuss requirements and what's expected of one another, and being in person for these discussions usually leads to better results.

We use google drive and google docs to both store and edit our documents. In google drive we have a task sheet that we keep updated with lists of objectives we need to accomplish with descriptions as well as deadlines. We also keep track of superfluous content we discuss during team meetings in a meetings document.

To communicate outside of meetings, we have made a group in an application called slack. It is both a mobile application as well as a web application. Here we can communicate outside of our meetings if needed. Slack also provides a neat feature of creating multiple channels to keep discussions in certain channels relevant to whatever topic is supposed to be talked about. Slack also provides features of allowing you to copy paste code snippets and keep their indentation, so it's good for communicating about code. It also allows you to upload documents to the channel and allows others to preview the document as well as download it.

9.2 Division of Responsibilities

Responsibilities for the system are divided into separate domains, varying degrees of overlap. Each domain has exactly one owner and each student owns exactly one domain. The owner of a domain is the student who is the most directly responsible for the performance of that domain. Any failures of a particular domain should be primarily considered a failure of the domain's owner but at the same time the whole group shares some fault for not detecting the error as well.

Each domain may have 1-2 deputies, who are expected to understand the system, and should be able to explain it as well as the owner but who are only responsible for working in that domain in assistive roles. Deputies are chosen based on overlaps in skills and overlap of domains, since each domain does not necessarily have a clear boundary, and because the performance of most domains depends heavily on the performance of certain domains in particular. In addition, deputies are expected to hold the domain owner accountable for making progress, assisting them directly if necessary. People involved in each domain are expected to meet with each other individually outside of the regularly scheduled weekly group

meetings, to ensure a greater level of accountability. The specific role of deputies is expected to vary within each domain, to allow flexibility in development processes.

This task allocation pattern has significant benefits. There is no part of the system that can fail as a result of only one person, yet if it does fail, it is clear who is responsible. Furthermore, each part of the system ends up being developed by multiple people who have relevant experience. Furthermore, there is a natural overlap of skills in different domains which makes the owner-deputy roles come about as a natural consequence of the requirements. The domains were allocated according to Table 21 below:

Table 21 – Division of Responsibilities

Domain	Owner	Deputies
Application	Elliot Rodriguez	Jonathan Gillis
Kiosk - Software	Jonathan Gillis	Sebastian Rodriguez & Jon Staudt
Kiosk – Electrical	Jon Staudt	Jonathan Gillis & Sebastian Rodriguez
Sensors & Communications	Sebastian Rodriguez	Jon Staudt

9.3 Estimated Project Cost and Financing

This project is funded mostly by the project members themselves. Additionally, a gofundme was created to bring in donations. Table 22 lists our estimated costs.

Table 22 Project Cost

ITEMS TO BE BOUGHT	AMOUNT	COST	Max Total
Microcontrollers	1 - 2	$40.00	$80.00
PCB	~ 3	$20.00	$60.00
Sensors	8	$4.00	$32.00
Electronic Components (resistors/capacitors/inductors)	Bundle	$15.00	$15.00
LCD Screen Display	1	$30.00	$30.00
Cloud Hosting Service	1 Virtual Machine	~$20/mo	$60.00
Electrical Piping	25 feet	$10.00	$10.00
			$287.00

9.4 Project Milestones

Project Milestones were created based on rough estimates, and were adjusted to try to align with the project members schedule. Most milestones for the first half are focused on research and design, while functionality and implementation are left for Senior Design 2. Table 23 following table lists out our planned milestones and deadlines.

Table 23 - Project Milestone Schedule

	Objective	Deadline	Time Frame
	Senior Design 1		
Documentation	**Project Idea**	**5/28/2017**	**5/19 - 5/28**
	Initial Project	**6/2/2017**	**5/30 - 6/1**
	Table of Contents	**7/1/2017**	**6/12 - 6/29**
	Draft Document	**7/7/2017**	**6/12 - 7/2**
	Final Document	**8/1/2017**	**7/8 - 7/30**
Formalization	**Standards, Specifications, Constraints**	**6/1/2017**	**5/28 - 5/30**
	Choose Sensors	**6/7/2017**	**6/2 - 6/7**
	Choose Microcontroller	**6/7/2017**	**6/2 - 6/7**
	Choose Form of Communication	**6/7/2017**	**6/2 - 6/7**
Prototyping	**Sensor Interface**	**6/20/2017**	**6/9 - 6/19**

	Power Supply	6/18/2017	6/9 - 6/17
	Communications	6/21/2017	6/9 - 6/20
Design	PCB	7/4/2017	6/17 - 7/3
	Casing	7/8/2017	7/1 - 7/7
	Connections	7/4/2017	7/1 - 7/7
	App	7/4/2017	6/17 - 7/4
	Senior Design 2		
Presentation	CDR Presentation	TBA	TBA
	Conference Paper	TBA	TBA
	Midterm Demo	TBA	TBA
	Final Presentation and Demo	TBA	TBA
	Exit Interview	TBA	TBA
Fabrication and Evaluation	Order and Assemble	TBA	TBA
	Specifications Review	TBA	TBA
	Testing	TBA	TBA

10 Conclusion

This project began as an idea to track the availability of individual parking spaces, with an expensive sensor package, and to locate every car, down to the parking space, all of this happening automatically, without any interaction from the user. We quickly realized that this would be both expensive, and unoriginal, so we set out to tackle a design problem not adequately addressed by previous attempts.

Numerous engineering problems were solved in this project during the period research. We believe we have implemented all the specified requirements in a very cost-effective way, as desired, and will continue to drive our ideas of each component, as it is successfully integrated into the system

We believe we have demonstrated that there is major Printed Circuit board design in our project, going as far as to design a mock-up of the final PCB. This design will be revised and revisited during our Senior Design 2 semester, and the hiatus in between, until we are more confident in the design.

We believe we are fully prepared going into Senior Design 2, to begin construction of a final product, and that we will not need significant extra time to complete the project, as described by the requirements specifications. We believe that our design is sound and that we have adequately considered all related standards and constraints which have influence our design process.

10.1 Features Left out

In the pursuit of a cost-effective product, we have left out many features we originally wanted to implement, but which showed themselves to be too costly. For example, we wanted the system to track the locations of cars without any input or interaction from users. After researching possible solutions such as RFID, we discovered that any attempt would be too costly, with the necessary RFID readers being in the hundreds of dollars range.

Furthermore, we wanted the system to track individual parking spaces without interaction from users, but after researching previous Senior Design Projects, we decided this was not possible to implement without exceeding our budget constraints.

11 Project Actualization

Over the course of Senior Design II, much of our design was revisited or changed. In this section, we will describe the final design used

11.1 Kiosk Design – Software

In the end, we ended up using the Arduino IDE and platform to write the code for the kiosk. The platform was used for programming the microcontroller and the Wi-Fi module, which is where some computation was offloaded to. This helped reduce requirements of the host processor.

11.2 Sensors

The sensors remained largely unchanged. The sensors were composed of a photoresistor which had a laser shining into it. When a car would break the laser beam, the output of the photoresistor circuit would go low and trigger the counter to increment or decrement, depending on whether a car had entered or exited. There were three variations of circuits for the sensors, depending on where the sensors were placed. Two circuits were used per lane in order to reduce false positives.

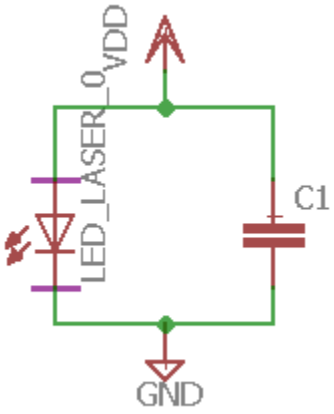


Figure 30 – First Variation of Sensor Circuit

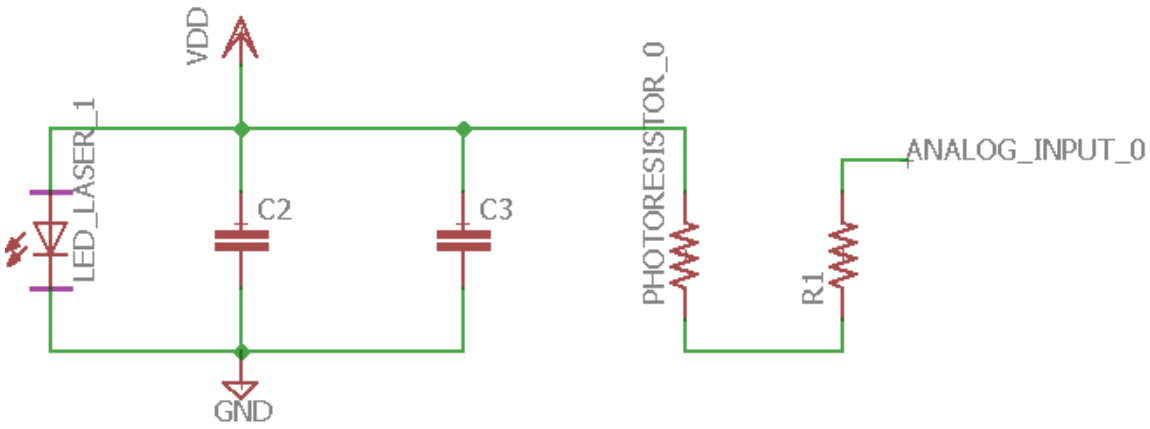


Figure – Second Variation of Sensor Circuit

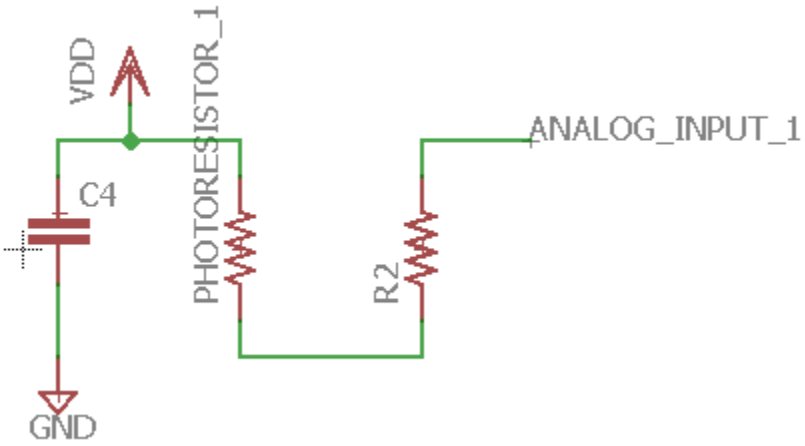


Figure 31 – Third Variation of Sensor Circuit

11.3 Housing

The housing was a simple plastic 5.25" x 3.25" x 1.6” box. There were seven of these boxes in total; one for each sensor circuit and another for the Kiosk system. Holes were drilled in the sides for wiring between the housings, as well as holes for the lasers and photoresistors. The Kiosk housing also had holes bored into the top for the display and the keypad.

Figure 32 – Housing Example

11.4 Microcontroller

The microcontroller used for the final project in the ATMEGA328P-PU purchased from microchip technologies. It has a 20 MHz core clock speed with twenty-eight pins, one UART and 6 analog to digital converters. The chip's pins are through hole pins making it easy to solder onto the printed circuit board rather than a surface mounted chip.

Figure 33 – ATmega328P

11.5 Printed Circuit Board (PCB)

The printed circuit board used was very simple in design using only two layers of copper, including the copper pours used for the main Vcc of +5 volts and ground, being Vdd. Ten PCB's were ordered for a total of $2.00, making each only $0.20 each. Our supplier chosen is Easy Eda, chosen for their fast turnover rates and low prices. The design of the board was done completely within EAGLECAD. everything but the sensors was mounted on the PCB.

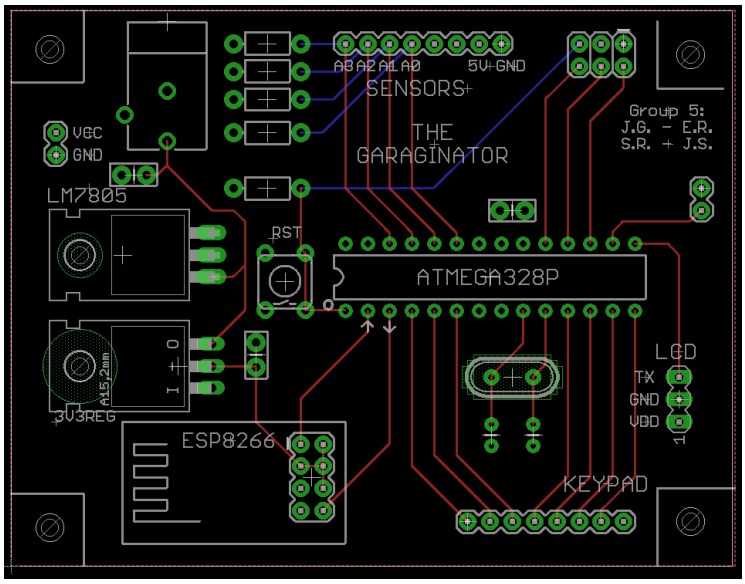
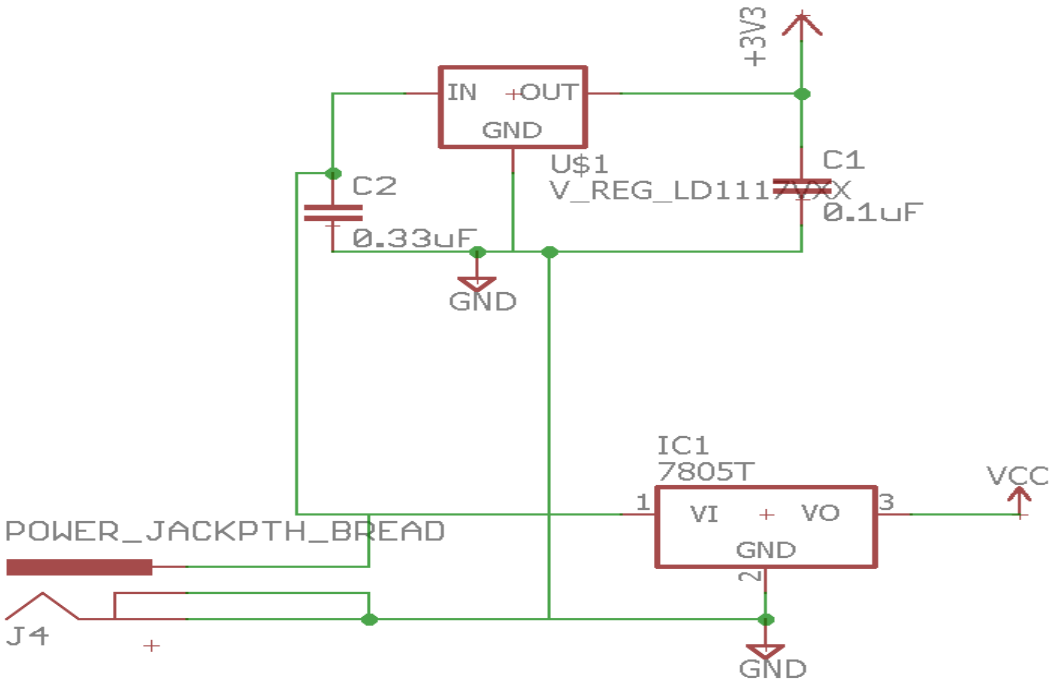


Figure 34 – PCB Layout

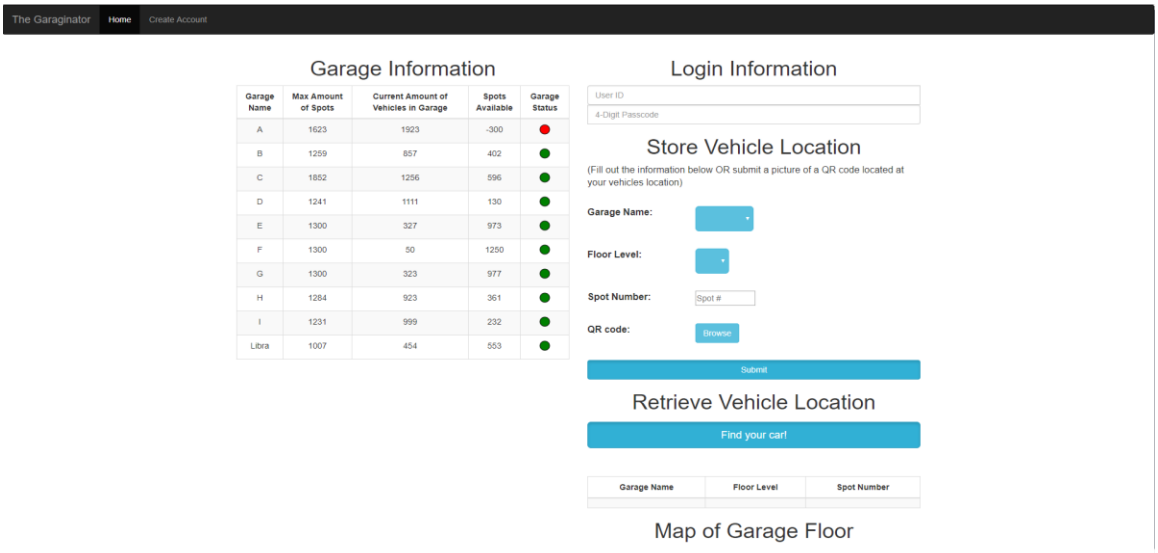
11.6 Power Supply

The entire device ran off of a wall jack to supply a 9 volt DC voltage through a DC barrel jack, which is then stepped down to 3.3 and 5 volts by two separate voltage regulator chips located on the PCB. The 3.3 volts is used to power the wifi module, while the 5 volts powered everything else including the MCU.

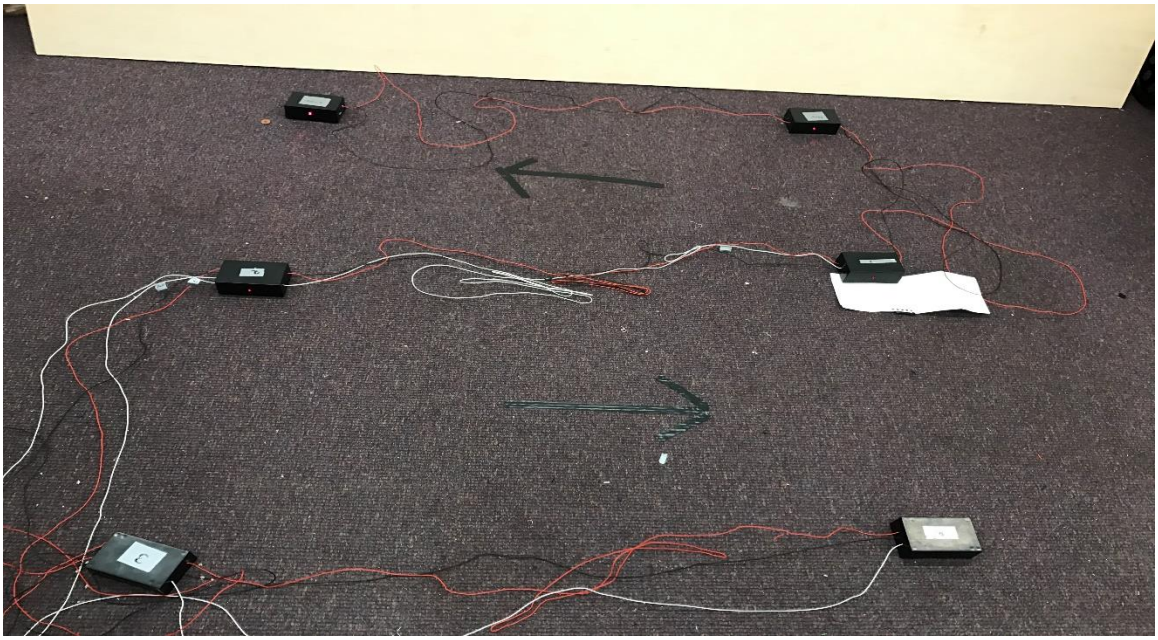


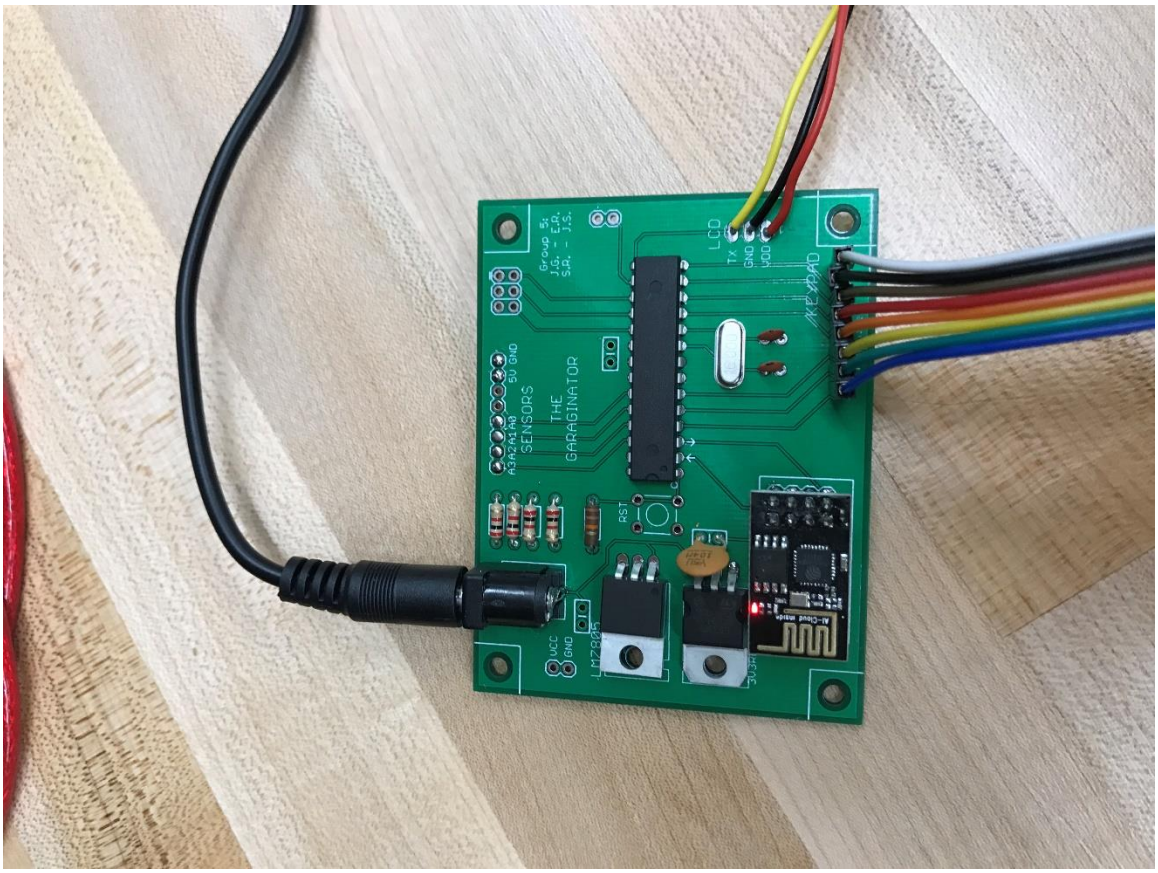
11.7 Software

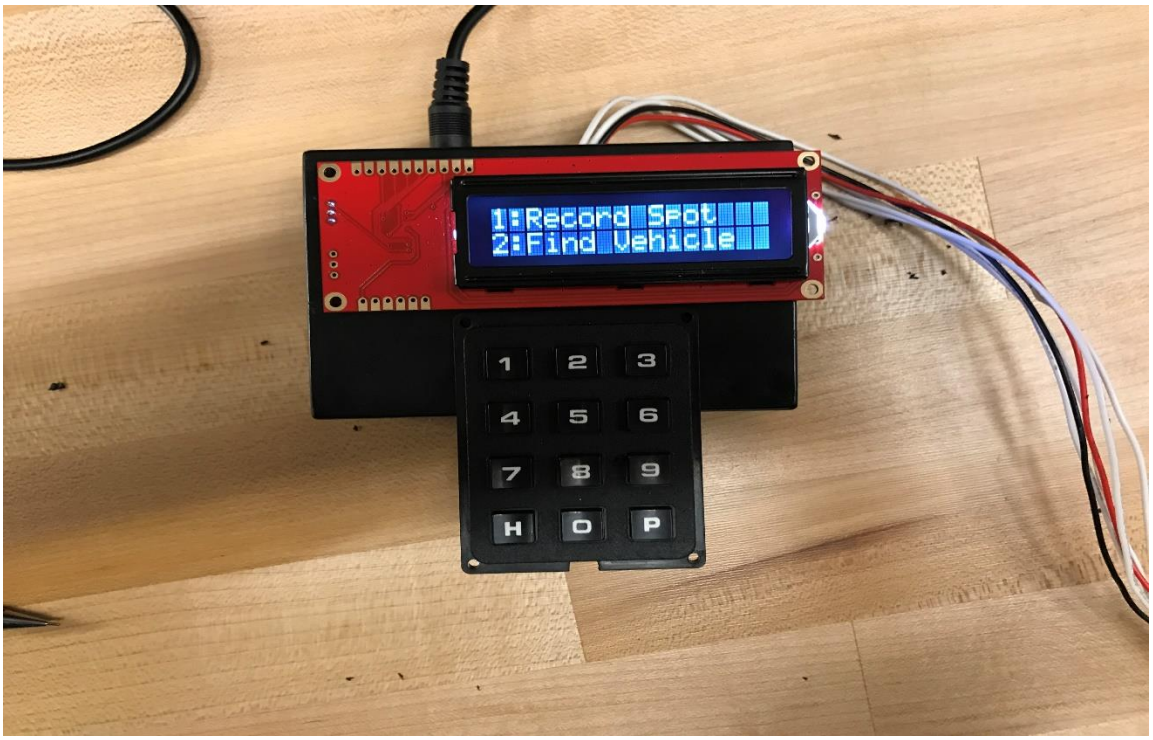
Our software system did not differ much from our initial idea, except that we did not create a mobile application. We instead decided to only create a web site which allowed for anyone with a smart phone that can run an internet browser the ability to visit and use the website. Below is a picture of the final website.



11.8 Final Product Pictures




THE
GARAGINATOR
SENSORS
LCD
KEYPAD
RST
LM2596


1:Record Spot
2:Find Vehicle
1
2
3
4
5
6
7
8
9
H
0
P

12 Appendices

12.1 References

[1] L. Ada, "PIR Motion Sensor," adafruit, [Online]. Available: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor?view=all. [Accessed 20 July 2017].

[2] Rockwell Laser Industries, "Laser Standards and Classifications," Rockwell Laser Industries, 2017. [Online]. Available: https://www.rli.com/resources/articles/classification.aspx. [Accessed 29 July 2017].

[3] CUI Incorporated, "Power Supply Safety Standards, Agencies, and Marks," CUI Incorporated, Tualatin, Oregon, 2014.

[4] CUI Incorporated, "Efficiency Standards for External Power Supplies," CUI Incorporated, Tualatin, Oregon, 2016.

[5] C. Severance, "JavaScript: Designing a Language in 10 Days," vol. 45, no. Computer, p. 8, Febuary. 2012.

[6] M. Widenius and D. Axmark, MySQL, Bonn: Mitp, 2003.

[7] A. Harris, *The Birth of Node: Where Did it Come From? Creator Ryan Dahl Shares the History,* 2013.

[8] C. E. Spurgeon, *Ethernet: the definitive guide*. Beijing: OReilly, 2009.

[9] “Faster speeds. More data. SmartTechnologies.,” *Fast Satellite Internet Service from HughesNet | 1-855-881-3072*. [Online]. Available: https://www.satelliteinternet.com/. [Accessed: 20-May-2017].

[10] “Wi-Fi Alliance,” *Wi-Fi Alliance*. [Online]. Available: http://www.wi-fi.org/. [Accessed: 12-Jul-2017].

[11] “Evaluating the communication performance of an ad hoc wireless network,” *Evaluating the communication performance of an ad hoc wireless network*. [Online]. Available: http://ieeexplore.ieee.org/document/1017498/. [Accessed: 21-May-2017].

[12] “How it works | Bluetooth Technology Website,” *Bluetooth*. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works. [Accessed: 02-Jun-2017].

[13] J. H. Davies, *MSP430 microcontroller basics*. Amsterdam: Elsevier, 2011.

[14] “Universal Serial Bus,” *Universal Serial Bus*. [Online]. Available: http://www.usb.org/home. [Accessed: 01-Jul-2017].

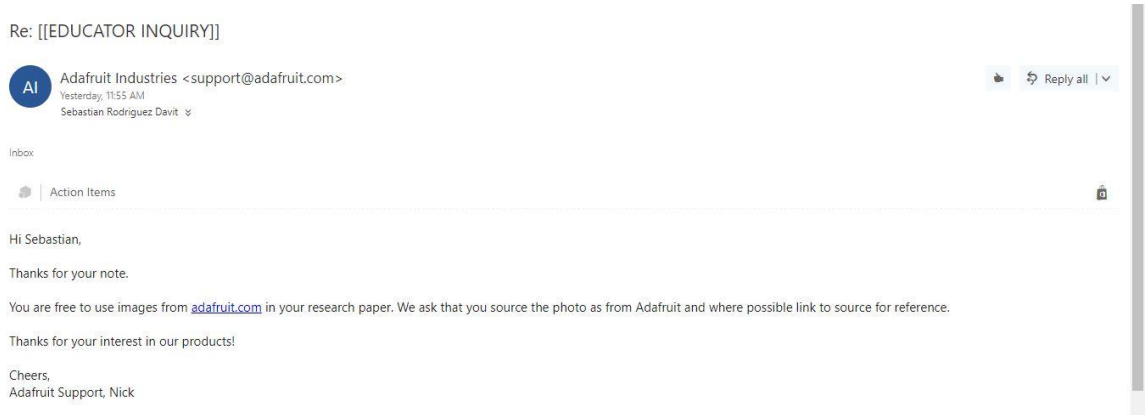
[15] “SAM V71 Xplained Ultra Evaluation Kit,” *SAM V71 Xplained Ultra Evaluation Kit*. [Online]. Available: http://www.atmel.com/tools/atsamv71-xult.aspx. [Accessed: 05-May-2017].

[16] “Microchip Technology Inc,” *Microchip Technology Inc*. [Online]. Available: http://www.microchip.com/. [Accessed: 14-Jun-2017].

[17] “Teach, Learn, and Make with Raspberry Pi,” *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 16-Jun-2017].

[18] “IEEE,” *IEEE*. [Online]. Available: https://www.ieee.org/index.html. [Accessed: 31-Jul-2017].

12.2 Emails



Re: [[EDUCATOR INQUIRY]]

AI Adafruit Industries <support@adafruit.com>
Yesterday, 11:55 AM
Sebastian Rodriguez Davit

Reply all

Inbox

Action Items

Hi Sebastian,

Thanks for your note.

You are free to use images from adafruit.com in your research paper. We ask that you source the photo as from Adafruit and where possible link to source for reference.

Thanks for your interest in our products!

Cheers,
Adafruit Support, Nick