Park Shark Parking Garage Monitoring System

Travis Bangs, Keegan Conway, Marcelino Galarza, Beatriz Jimenez

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* – The Park Shark system aims to provide students with accurate information about the parking availability in the school's parking garages. The Park Shark system uses a combination of computer vision software and ultrasonic sensors to monitor the parking spots throughout the garage. Each computer vision module will monitor three parking spots at a time. The sensor modules will monitor the parking spots that are not within the camera's field of view and all the parking spots on the roof of the parking garage. The information about the spot's availability from the sensor modules is then relayed wirelessly via Bluetooth to the computer vision module. The information gathered from both modules is then transmitted to the database over Wi-Fi. The parking availability throughout the garage can be viewed in real-time on the website and mobile application. Utilizing the Park Shark system will reduce the amount of time it will take for the students find a parking spot. By implementing a multi-sensor system, the required amount of hardware necessary to monitor the parking spots and overall cost of the system will be reduced.**

***Index Terms* – Computer vision, mobile application, Nordic nrf52832, parking garage, ultrasonic sensor, website, wireless communication.**

I. Introduction

At the University of Central Florida, finding a parking spot on campus is one of the everyday struggles a student will inevitably face. Every morning is a battle to see who succeeds in getting a parking spot before the parking garage reaches full capacity and then the real hunt begins. One solution to mitigate this problem would be to build more parking garages, but that would require more land and more money. The other solution would be to create a system that can report the availability of parking spots in each garage. By implementing such a system, time spent searching for a parking spot will be significantly reduced. This way students won't arrive late to class and miss important class material. The parking garages at UCF currently have a monitoring system implemented that is quite unhelpful. Their system utilizes sensors at the entrance of the parking garage to keep track of the number of cars entering and exiting the premises. Their monitoring system then uses this information and number of spots the garage possesses to report the estimated number of parking spots available. By utilizing this method, inaccurate information is reported to the student. The idea behind Park Shark is to create a system that will monitor each parking spot and report an accurate number of available parking spots left to students quickly and effectively via the internet. The Park Shark system utilizes a multi-sensor approach to monitor each parking spot in the parking garage. A single computer vision module is set to monitor three parking spots simultaneously. Any parking spots out of the camera's field of view is monitored by an ultrasonic sensor module. Each sensor module will relay the parking spot's status to the computer vision module via Bluetooth communication. The data gathered by both the sensor and computer vision systems will be transmitted to the database over Wi-Fi. This information will then be displayed on a website and mobile application. Ultimately, the parking spot availability for each parking garage will be available to the student's disposal at a click of a button.

II. Requirement Specifications

The main goal of the Park Shark system is to create an accurate and efficient monitoring system at a low cost per sensor unit. The system should be able to provide extremely accurate parking information, so users may use the app in real time to determine where to park. An additional goal of the Park Shark system is to create a user-friendly application for users to view up-to-date parking information quickly and efficiently. The below requirements and specifications layout how we will accomplish this task.

- The final system shall utilize two sensors, an Ultrasonic unit as well as a Computer Vision (CV) unit
- The cost for an Ultrasonic unit shall not exceed $50
- The cost for a CV unit shall not exceed $200
- The dimensions of the Ultrasonic unit shall not exceed 4 x 3.7 x 3.1 inches
- The dimensions of the CV unit shall not exceed 8 x 8 x 3.5 inches
- The Ultrasonic unit will be able to communicate with the CV unit via Bluetooth
- The CV unit will be able to send data over Wi-Fi to an external database
- The Ultrasonic unit will pull no more 50 mA and shall run off a small battery pack and require a battery change every month
- The CV unit will be powered by a 12-volt battery pack and require a battery change every 24 hours
- The Ultrasonic unit should be able to detect vehicles at least 5 ft. from the wall or the ceiling
- The Ultrasonic unit should also be able to enter a sleep mode if the parking garage is near empty to save power
- The CV unit should be able to provide availability for a minimum of 3 spots with an accuracy greater than 90%
- The data displayed on the website and mobile application will be updated every 30 seconds or less
- The website and mobile application will be able to display parking availability and statistics in a simple and efficient way to the users

III. Park Shark System

The Park Shark system consists of five major development areas: the PCB design, the Nordic nRF52 microcontroller's coding, the MongoDB database, the website, and the mobile application.

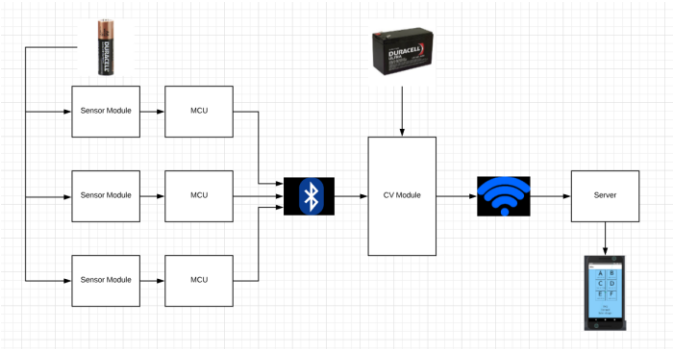

Fig 1. This figure shows the overall block diagram of the Park Shark system.

The block diagram displays the components that make up the system, how they communicate between each other, and how their power supplies. On the left side of the diagram, the sensor modules are shown to be powered by AA batteries and controlled by the MCU. The information gathered by the sensor modules are sent to the computer vision module via Bluetooth which is powered by a 12V battery. The data received by the CV module and its own data is transmitted to the database over Wi-Fi. The information is then displayed on the website and mobile application providing the user with up to date parking availability in the parking garage.

IV. Design

A. *Computer Vision Unit*

The computer vision unit was designed to allow for a more efficient way of counting cars across multiple spots. From our research of other systems, most parking garage monitoring systems used an individual sensor for each spot. Our goal was to improve upon this by using computer vision to monitor multiple spots. To accomplish this a form of object detection was necessary, as well as a central computing device that could handle running object detection. Several devices were considered such as the Raspberry Pi, BeagleBone, and Jetson TX1. While the Jetson TX1 offers incredible computation power and would allow for a much faster implementation of our system, the price difference ($299 for students) is simply too expensive for the desired price range of our product. If there was no issue in budget, and the desire was to make the fastest and most accurate system, the Jetson TX1 would be the ideal computational device for this project. The Raspberry Pi boasted better specifications compared to the BeagleBone and a much lower price compared to the Jetson, so ultimately the Raspberry Pi was selected. The computer vision unit is also responsible for receiving data from the Ultrasonic unit via Bluetooth, which can then be transferred to the database via Wi-Fi, which the Raspberry Pi is well suited for. The Raspberry Pi features a 1.4 GHz quad-core ARM processor which allows for more complicated forms of object detection to be used.

For object detection, several methods were researched and tested such as Cascade Classifiers and Histogram of Oriented Gradients, but eventually the team decided on an extremely new method that utilizes convolutional neural networks called You Only Look Once (YOLO). While the other methods of detection could reasonably be used for this project, they are relatively old versions of object detection compared to the much newer YOLO, which achieves high object detection accuracies with relatively fast detection times. The YOLO algorithm looks at an image once and divides the image into a grid of 13 by 13 cells with each cell being responsible for predicting 5 bounding boxes. This produces a total of 854 bounding boxes. It then produces a confidence score for each box and tells us if the bounding box is accurate. This confidence value is used to get rid of a large amount of the boxes created. It also predicts the class of an object, trying to classify the object within the box, such as a car or dog. The presence of overlapping boxes at the end of detection was detrimental to the overall detection, and therefore the process of non-max suppression is used to remove any boxes that overlap with each other.

Fig 2. This figure shows the result of the YOLO detection from the computer vision unit.

The team initially planned to use a version of the YOLO algorithm called Tiny-YOLO, which utilizes a smaller model and provides extremely fast detection times at the loss of some accuracy. While the Tiny-YOLO version offers much faster detection times, the loss in accuracy was far too great. For our project, with the goal of updating the database at least every 30 seconds, accuracy was favored over speed. The latest version of the YOLO algorithm, YOLOv3 completes detection in under 30 seconds on the Raspberry Pi, with very high

accuracy. Because of this the full version of YOLO is utilized over the Tiny-YOLO version. The team initially utilized TensorFlow and Keras implementations of the algorithm (such as Yad2k), but ultimately ended up running the algorithm using pure OpenCV with similar runtimes and accuracies. The ability to run the algorithm using pure OpenCV was made possible by the addition of a neural net module added to OpenCV past version 3.4 called DNN. This module allows the creation of neural networks which can then be used by the YOLO algorithm. Python is used as the primary language, as the languages ease of prototyping, simple installation of packages, and straightforward syntax made the language ideal for prototyping and the final design.

An issue in the original design of the computer vision sensor become apparent upon initial testing. Upon installing two computer vision sensors, if both sensors look over the same spot, the same car would be counted twice. Several attempts were made to remedy this issue. One attempt sought to create a panorama image from the images taken from each computer vision sensor and perform detection on the panorama. To accomplish this, the use of the SIFT algorithm and OpenCV libraries to stitch images together were used. If two computer vision sensors are installed, two images would be taken and transferred via scp to one of the Raspberry Pi's. The stitching algorithm would then be run and result in one image. YOLO would then run on the resulting image giving us a final count of detected vehicles. However, upon testing, the inability of SIFT to find unique key points within the images to perform the stitching on proved this technique to be inefficient, as the two images taken within the garage were extremely similar. The unfortunate solution to this problem was to install the camera, view the resulting image from the sensor location, and crop the image to look at the intended spots. While this solution is not entirely practical for non-trained individuals to accomplish, the result is a guaranteed number of occupied spots within the parking garage.

Upon the desire to report not just availability, but also the availability of individual spots, some additional methods were necessary. The goal was to attempt to use the white lines that separate each parking spot to create bounds for each spot. With the bounds created, we could then calculate the center point of the detected cars and determine if that spot if full or not. We first attempted to create the bounds using Hough Line detection, attempting to use the white lines of the parking spots to create the bounds. Upon testing, this method proved extremely inaccurate. Depending on where the car parked in the image, the lines where not always visible from the perspective of the camera, as well as depending on the time of day, the white lines were washed out by the parking garages lights. Another unfortunate solution to this problem was to install the camera, view the resulting image, and manually set the bounds for the parking spots. Again, this requires more work in the installation process, but allows guaranteed spot availability.

Fig 3. Parking garage D at night. This figure shows the issues of detecting the parking spot lines using computer vision.

The Computer Vision Unit is also responsible for getting the information from the Ultrasonic Unit and sending it the database. The communication between the CV unit and the Ultrasonic unit is done over Bluetooth, utilizing a Python library called PyGatt. This module is a wrapper for the GATT tool and allows us to gather the information by connecting to the individual Ultrasonic sensors and gathering the data from the characteristic being broadcasted by the devices. The communication between the CV unit and the database is handled by the Python Requests library, which allowed for the sending of GET and PATCH requests to the API to send and receive the necessary information to update parking availability as well as relay updated information that is used by the ultrasonic sensor.

B. Ultrasonic Sensor

One of the major components that makes up the sensor module is the sensor itself. After considering various sensor devices, the ultrasonic sensor was chosen due to its high sensing range, low power consumption, and low cost. The ultrasonic sensor can measure distance by emitting ultrasonic waves at a certain frequency and retrieving the reflected sound wave. The time it took for the wave to return to the sensor is used to calculate the distance between the device and the target of interest. The main advantage about the ultrasonic sensor is that has a high sensing range (about 4 meters). Hence, the sensor module could be placed on the ceiling, wall, or floor depending on the user's preference. It will still be able to detect a car in the parking spot no matter its orientation. The distance threshold of the device is set by modifying the module's code.

While researching ultrasonic sensor, it turns out that there are two version of the HC-SR04 sensor. The HC-SR04 requires a minimum operating voltage of 4.5V meanwhile the HC-SR04+ requires a minimum of 3V. Since the Nordic nRF52 chip requires an operating voltage of 3.3V, the HC-SR04+ was chosen. By doing so, the circuit design for the power supply is simplified by not requiring the use of two voltage regulators which also results in a lower cost. The following table list the specifications of the ultrasonic sensor used for the sensor module.

Table 1. HC-SR04+ ultrasonic sensor specifications.

Specification	**HC-SR04+**
Operating Voltage	3.0 to 5.5 V (DC)
Operating Current	10 to 20 mA
Quiescent Current	3 mA
Frequency	40 kHz
Range	0.2 m to 4.5 m
Resolution	3 mm
Temperature Range	-15 to 70℃
Dimensions	43 mm x 20 mm x 15 mm
Weight	8.5 g
Sensing Angle	30˚cone
Angle of Effect	15˚cone
Trigger Input Signal	10µs TTL pulse
Echo Output Signal	Output TTL PWL signal
Price	$1.44

C. Microcontroller

The ultrasonic sensing units are powered by a Nordic nRF52832 microcontroller unit. This microcontroller chip contains an ARM Cortex M4 CPU and a Nordic Bluetooth transceiver in the same chip. This means that the CPU and the Bluetooth stack can communicate easily rather than having to design dedicated hardware connections between the Bluetooth transceiver and a separate dedicated CPU.

The ARM Cortex M4 is more than powerful enough to handle the calculations involved in the seconds-long sensing intervals of the battery-powered ultrasonic sensing unit. Another benefit of a combined solution is a much lower power draw. Overall the power draw of the Nordic nRF52 chip is much lower than the more common and less powerful MSP430 or ATMega128 CPUs. Minimizing power draw within the battery-powered sensing unit relies on putting the CPU to sleep as much as possible between sensing intervals, only waking up to check the ultrasonic sensor or to send/receive Bluetooth data.

The microcontrollers on the circuit boards are programmed via the Nordic nRF52 Development Kit. The development kit has debug passthrough pins that allow external Nordic-powered boards to be flashed and debugged by the Development Kit's internal SEGGER J-Link debugger. This eliminates the need for the group to purchase the much more expensive standalone J-Link debugger or the need to use an unauthorized cheaper clone in order to download the firmware programming onto the ultrasonic sensing unit's processors.

D. Power Supply

In the Park Shark system, the sensor modules require an operating voltage of 3.3V. The goal is to create a sensor module that can operate for at least a year before demanding a battery change. Before selecting the power supply, the consumption characteristics of each active element in the sensor module was taken into consideration and an activation plan was created for the sensor module. The activation plan will ensure that the battery life will last for as long as possible by activating the MCU and sensor only at specific intervals. For example, the MCU will simply go into sleep mode when the parking garage is not at near capacity.

Once the overall power consumption of the device is calculated, a battery that supplies the necessary voltage needed to power the device for a year can be selected. To select a battery for the sensor module, the required voltage supply for the sensor module must be considered. The two main components in the sensor module that require voltage input are the Nordic nRF52832 chip and the ultrasonic sensor. The nRF52832 chip requires a supply voltage of 3.3V. The supply voltage range for the ultrasonic sensor is between 3 and 5.5V. Thus, a supply voltage of 3.3V was selected for the ultrasonic sensor because it's the same voltage required to power the nRF52 chip. By doing so, the circuit is simplified by utilizing one voltage regulator instead of one for each component. This will also lower the total manufacturing cost of the PCB since it would eliminate the need to purchase two separate regulators. Based on this information, the supply voltage necessary to power the sensor module is about 3 to 4.5 volts.

The two most important parameters to take into consideration when selecting a battery is its nominal voltage and its battery capacity. The battery capacity necessary for the device can be calculated knowing the total current drawn by the active elements in the sensor module and the desired battery lifetime. The desired battery lifetime is at least a year and the calculated total current drawn by the module is 50µA. By plugging these values into the equation shown below, the calculated battery capacity necessary to power the device for a year is 438mAh.

$$\frac{\text{Battery Capacity (mAh)}}{\text{Current Draw (mA)}} = \text{Battery Life (hours)}$$

The batteries chosen to power the device are AA batteries due to its small size, easy attainability, and low cost. These batteries provide more than enough battery capacity considering a single AA battery provides 2870mAh. By utilizing AA batteries, the device is estimated to last for 6 years. Since the nominal voltage of an alkaline cell is 1.5V and

the necessary supply voltage for the device is 4.5V, three AA batteries will be used.

After testing the final sensor module, it turns out that the actual current drawn by the device is about 0.2 - 0.3mA. By utilizing AA batteries, the device is estimated to last for at least 1 year.

E. Bluetooth Communication

Each of the ultrasonic sensing units use the Nordic nRF52832 Bluetooth MCU, which implements Bluetooth 4.2 and additionally supports some Bluetooth 5 features, such as lower power consumption. This Bluetooth chip allows a far more robust connectivity solution than older standards that other parking systems may have used, as the newest Bluetooth standards allow far more connections between devices and far less power draw, making a battery-powered solution much more feasible than mere years ago.

The Bluetooth SIG also has a Bluetooth Mesh standard, allowing devices to connect through each other to reach a specified device that would normally be out of range. However, the official Bluetooth Mesh standard relies on devices that are always broadcasting and receiving data constantly and thus rely on being connected to a constant power source. This is not compatible with the paradigm of using a mesh of battery-powered units.

Therefore, a mesh implementation would have to be manually created. It is simple to allow a device to passthrough data from one other sensor, however allowing a sensor to forward data from multiple other sensors is much more complicated and would also result in much more time spent scanning for new devices, which would have a very high power draw. In the implementation, the devices send their sensor info via Bluetooth to the constantly-powered Camera unit of the Park Shark system, which passes the data along to the backend on the Internet.

F. Backend / Database

At the beginning of the project, the Park Shark team initially intended to create a backend using Node.js and Redux, allowing for the front-end and the back-end to sit on the same machine. Upon discussion of the implementation of both the website and the mobile application, the decision to split the two became more appealing. Hosting the backend on a separate machine to serve as an API would allow the website and the mobile application to make requests for the parking availability, ensuring that both applications display the same information. This also simplifies the job of the computer vision module as now the unit only has to send requests to the API instead of trying to separately update both applications. This would also allow students at UCF to have a public API for parking availability that they could use in projects or hackathons. The backend was created using Node.js and utilizes a MongoDB database to store all information. The use of the Mongoose node package was utilized for all communication between Node and the MongoDB database. The database stores two models, Sensor and Garage, which stores all parking availability information. The details of both models can be seen in the data model given below. The API was deployed to a Heroku server for all hosting purposes.

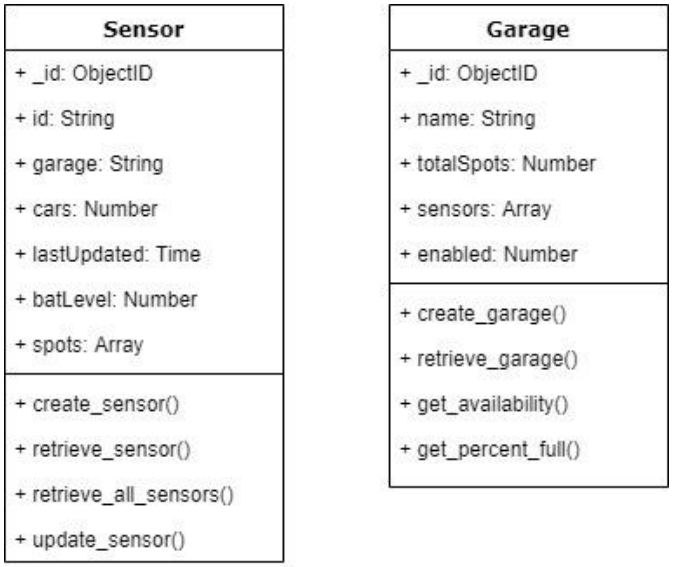

Fig 4. Data model figure for database

Each ultrasonic sensor and computer vision sensor are manually given an ID and populated with which spots the sensor is monitoring. This information is then used to create a sensor model as well as update the garage model. The Node.js backend itself follows the REST API principle, with routes for GET, POST, PATCH, and DELETE HTTP requests.

G. Website

The website will be one of two ways the user of the Park Shark system can access the UCF garage availability. We understand that most user will be using the mobile application as it's always at the palm of their hands, but the Park Shark team wanted an alternative to just a mobile application. Having a website allows anyone from any device to be able to access the garage systems information.

When the website is launched it will not require any information from the user just as the mobile app. The website will simply display the current garages available for the system. Once you click on one of the garages it will render the statistics for that specific garage. It will have the title and the remaining spots with a garage occupancy status percent bar. Underneath that is the specific floor and spot availability, which shows the spot ID and if it is taken or not. It also shows the floor capacity.

The website was built using React and deployed on a Heroku server. Unlike the mobile app the website does not

have an administrator login for the system. We decided that since the mobile app is the main source to the system, we would keep that functionality there and treat the website as an auxiliary source to garage data.

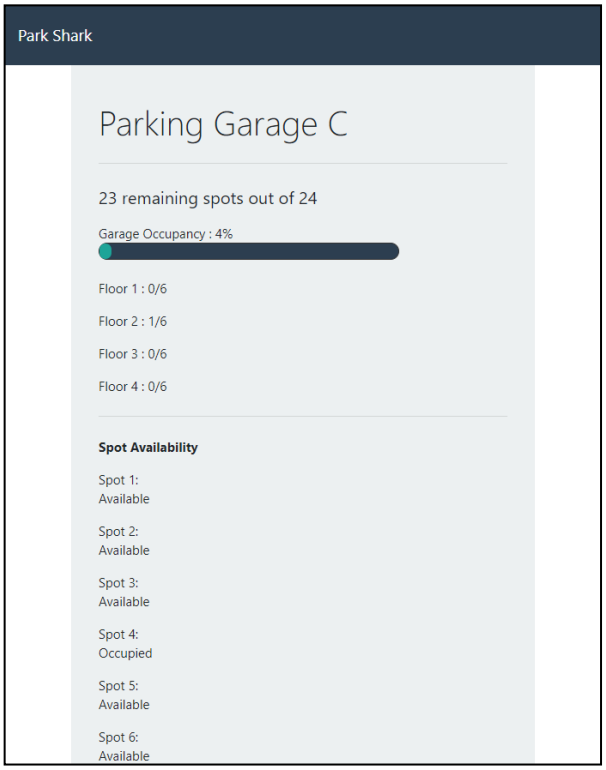

Fig 5. Park Shark Website frontend displaying garage C information.

H. Mobile Application

The mobile application will be one of two ways the users of the Park Shark system can access the UCF garage availability. The mobile application is the main method for students accessing specific garage availability, as most students have a phone. It will be pulling data from the database in order to display the parking garage availability.

When the mobile application is launched it will not require any information from the user. The user will not have to login and enter any data to acquire access to the application. The application should simply display the garages and the current availability of each garage. The user will have an option to click on any garage for more specific details if wanted. This is only an option, and not required to get percentage filled. This allows the mobile application to stay simple with the user in mind.

Although the mobile application does not have a user login it does have an admin account login. This allows the manager of the Park Shark system to edit the number of garages available on campus. The status of garages can also be adjusted. This is useful in the case of a garage being under construction or expanded. A perfect example is garage C at UCF which was temporarily unavailable to latter become available with many more parking spots. There are also special events that may require a garage to be closed off or made partially available.

The main page for any user will consist of a screen with the current UCF garages. This is displayed with the letter associated with the garage and right underneath it will be the number of spots available in that garage. The main page will also be the accessing point for our FAQ, contacts, and admin login screens. The FAQ page has the most basic frequently asked questions as well as a button to the UCF's official FAQ. The contacts page has information on all park shark team members as well as their emails. If a specific garage is clicked, then the user will be taken to that garage's specific statistics. Which will show what specific spot is taken and what floor it is on.

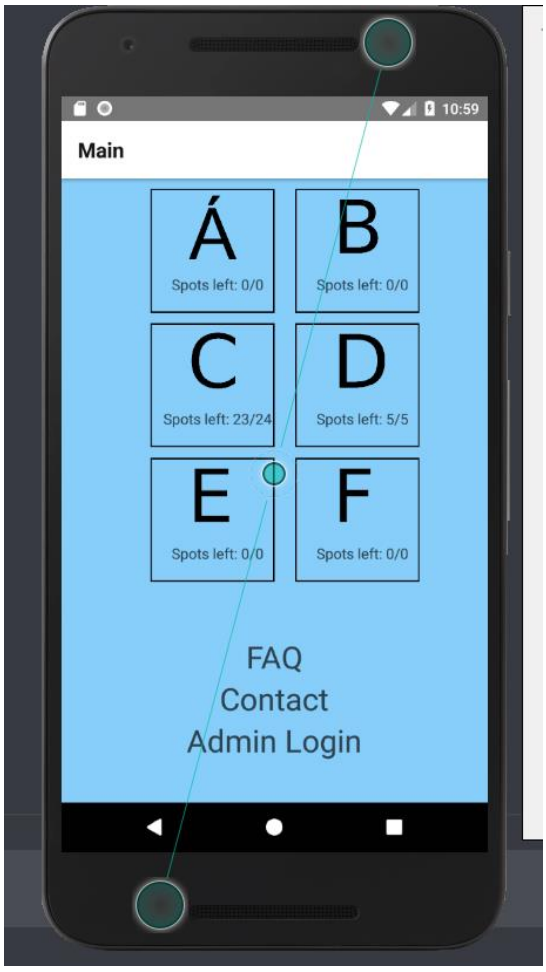

Fig 6. Main page for the mobile application.

The admin login will take you to a login screen. There is no registering option because the concept is that if this project were being implemented by someone the admin user would be created from the developers in to ensure security. Once the admin logs in with valid credentials the admin page will

appear. From here the admin can disable a garage by simply putting the garage letter and clicking the disable garage button. To enable a garage the admin would have to put a garage letter and the number of sensor/spots wanted for the garage and clicking the enable garage button. As an added feature the admin will have a button called sensor battery status which will output a list of sensors that need their batter replaced.

The application was built using React Native. React Native allows the developers to create real native applications without having to develop two separate apps. The alternative to native apps would be developing an android app using Java and using swift/objective-c for an iPhone app. This means that any user not just android or iPhone user can use our mobile application. Therefore, if our system were implemented for UCF all attendants would be able to quickly check all garage availability at the palm of their hands.

I. Housing and Assembly

The housing for the sensor module and computer vision module will be described in this section. The housing for the sensor module was made to contain three components: the battery pack, the PCB, and the ultrasonic sensor. On one side of the housing, two holes were cut out to allow the ultrasonic sensor to pop through. The other side, another hole was made to allow easy access to the switch button on the battery pack. The battery pack was attached to the inside of the lid of the box for accessibility and convenience for the user to change the battery when necessary.

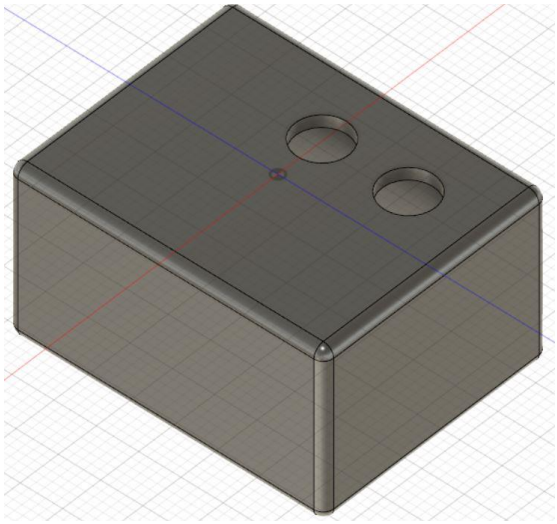

Fig 7. CAD design of the housing for the sensor module.

The main idea was to create a sensor module that was small and light, so the user would have the option to place it in a variety of locations. The sensor modules can be placed on either the wall, ceiling, or floor. It all depends on the user's preference. Ideally, placing the sensor modules on the ceiling will prevent anybody from accidently damaging the device. Although, on the top floor of the parking garage, the user would have to choose between placing the sensor either on the wall or the floor due to there being no ceiling.

The housing for the computer vision module was made to contain the 12V battery pack, the voltage regulator, the Raspberry Pi, and the camera. The housing must be big enough to fit all the components inside and have a hole for the camera to fit through.

The computer vision module will be mounted on the ceiling clamped to one of the many beams in the garage. The module will be positioned to oversee three parking spots. Thus, the module will be slightly tilted downwards to get a clear view of the parking spaces.

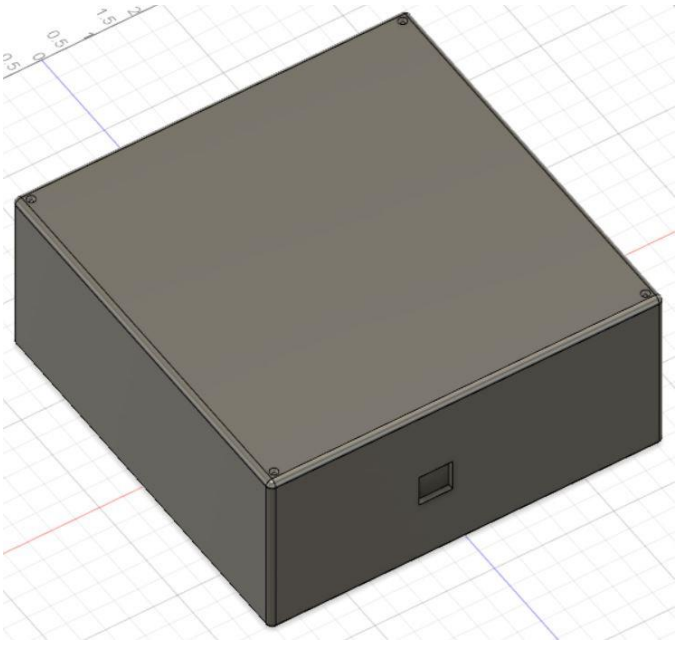

Fig 8. CAD design of the housing for the CV module.

V. TESTING

During Senior Design I and Senior Design II, the Park Shark has been implementing testing on each individual component of the system. Early testing was done to ensure all individual components work as intended. For the ultrasonic sensors a dev board was used to program the sensor component. It was also used to test the sensing range. For the mobile application testing was done throughout the whole process of development. The terminal was used as a console log to output intended results.

Final testing and integration were performed during the late mid Senior Design II semester. This testing focused on the whole system working together. The computer vision and ultrasonic sensors were set up in garage C and were tested for accuracy and responsiveness. This was the integration testing of the system working and verified that our system worked. The mobile application was also tested on a physical mobile

app to ensure the app was rendering properly on a real device. Due to constraints of not having a mac computer we were not able test it on an iPhone.

VI. Conclusion and Reflections

Park Shark is a parking monitoring system that relies on recent advancements in computer vision and wireless communication technology. The computer vision unit allows flexibility with client budgets for implementation, and the lower power consumption allowed by Bluetooth 4.2 and Bluetooth 5 results in far longer battery life than older systems without compromising responsiveness, performance, or cost.

Most of the challenge involved in the hardware design of the project involved tuning the Bluetooth parameters in order to ensure a minimal power draw without affecting responsiveness or connectivity. The main challenge of the software design of the project was that most of the group members were unfamiliar with React Native, meaning that creating the frontend and backend of the system took more time than anticipated as members working on it had to get used to React Native first before significant progress was made.

VII. Future Improvements

One improvement for the park shark system would be the ability to be implemented by multiple universities or businesses. This could change some of the requirements for the users because the system would need to know what information is trying to be accesses so a register account option could be made. An alternative would be to add a search bar to find the system you are looking for on the app on website end. In this case the system would be a group of systems divide up the keys, keys most likely being the name of the business.

A far more user-friendly way of displaying available parking spaces would be to show them on a map for the user to see. This functionality should already be easily supported by the backend as each parking space is already tracked individually as opposed to UCF's current counting implementation. The status of the spots merely needs to be overlaid on top of a map of the parking section.

The Bluetooth sensor network can be improved by allowing each node to communicate with more than one other sub-node. This adds difficulty in that supporting multiple connections would require the device to be constantly scanning for other devices to add to its network tree. Scanning draws a high amount of power, meaning that using regular scan intervals is antithetical to long battery life. One solution is adding a button to each sensor to manually enable pairing mode during installation, meaning that scanning is only enabled during the sensor's first connection, which should result in minimal power draw compared to regular scanning intervals. Another similar solution is that the server can tell a specific sensor to start scanning using a special command. This feature was thought of before the project was finished, however time constraints and the requirement of a new PCB revision that adds a button prevented its implementation.

VIII. Park Shark Team

The Park Shark team is made up of three computer engineering students and one electrical engineering student.

Travis Bangs, a computer engineering student, worked on the firmware design and programming of the ultrasonic sensing units and their Bluetooth connections, and assisted with the hardware schematic design for the wireless ultrasonic sensing units.

Keegan Conway, a computer engineering student, programmed the computer vision algorithms for the camera sensing unit, created the parking information database, the Park Shark website, and created an API to communicate the sensors' occupancy information to the server backend.

Marcelino Galarza, a computer engineering student, designed and programmed the server backend, and mobile application of the Park Shark system, creating the user interface and database interaction required for the it to function.

Beatriz Jimenez, an electrical engineering student, designed the printed circuit board layout and assembled the three circuit boards used in the demonstration of the project, in addition to handling the budget and logistics of ordering parts and manufacturing the circuit boards and housing.

Fig 9. Park Shark team