

End of Project Documentation

Data Collection of Motorized Vehicles D.c.M.V



**SACRAMENTO
STATE**

April 27, 2020
Professor Russ Tatro
CpE 191/EEE 193B

Team 4 - Team Four Starz
Sergio Cortes, Michael Khoo, Ryan Uda

TABLE OF CONTENTS

Executive Summary.....	v
I. Introduction.....	1
II. Societal Problem.....	3
A. Overview of Societal Problem.....	3
B. Background and Design.....	3
C. Benefits.....	5
III. Design Idea.....	5
A. Resources.....	5
B. Feature Set.....	7
C. Identification.....	7
D. Occupancy.....	9
E. Car Counting.....	9
F. Power.....	10
G. Database and GUI.....	11
H. Integration.....	12
IV. Funding.....	12
V. Project Milestones.....	15
VI. Work Breakdown Structure.....	17
VII. Risk Assessment.....	22
A. Critical Path Risk Mitigation.....	22
B. Risk Mitigation for Non-critical Path.....	24
C. Final Thoughts on Risks and Mitigations.....	25
VIII. Design Philosophy.....	26
A. Data Collection.....	26
B. Cost-Effective and Accessible.....	27
IX. Deployable Prototype Status.....	27
A. Identification.....	27
B. Occupancy.....	28
C. Car Counting.....	28
D. Power.....	29
E. Database and GUI.....	29
X. Marketability Forecast.....	29
A. Perceived Market Audience.....	30
B. Comparable Services and Products.....	31
C. Market Value.....	32
D. Refining Needed.....	33
XI. Conclusion.....	34
References.....	37
Glossary.....	39

Appendix A. User Manual	A-1
A. In the field operation	A-1
B. Modifying the detection parameters	A-2
C. Uploading the collected data to the database	A-3
D. Charging	A-7
Appendix B. Hardware	B-1
Appendix C. Software	C-1
Software Flowcharts	C-1
Application Specific Code Review for the Car Parking Detection Program	C-3
Application Specific Code Review for the Car Counting Program	C-5
Application Specific Code Review for the GUI Program	C-6
Software Test Results	C-8
Appendix D. Mechanical Aspects	D-1
Appendix E. Vendor Contacts	E-1
Appendix F. Resumes	F-1

TABLE OF FIGURES

Figure 1: Schematic of Loop Detector [1]	4
Figure 2: Example of a convolutional neural net, LeNet-5 [6]	8
Figure 3: Multiple parking detection program [7]	9
Figure 4: Prototype car counting program [8]	10
Figure 5: Power Bank [10]	10
Figure 6: Power Bank Switches [10]	11
Figure 7: Database [11]	11
Figure 8: GUI [11]	12
Figure 9: Laboratory Prototype [13]	16
Figure 10: Enclosure [14]	17
Figure 11: Risk Assessment Heatmap [17]	24
Figure 12: Bus identified [7]	28
Figure 13: Parked motorcycle [7]	28
Figure 14: Horizontal thresholds [7]	28
Figure 15: Vertical thresholds [7]	29
Figure 16: Raw data [11]	29
Figure 17: GUI and data [11]	29
Figure 18: The California Department of Transportation Logo [20]	30
Figure 19: Attractive opportunities in the Traffic Management Market [26]	32
Figure A 1: Team4 Device I/O [14]	A-1
Figure A 2: Team4 Change Directory [7]	A-1
Figure A 3: Team4 Start Program [7]	A-2
Figure A 4: Team4 Car Park App Spots [7]	A-2
Figure A 5: Team4 Car Park App Integer Parameters [7]	A-3
Figure A 6: Team4 Car Park App Boolean Parameters [7]	A-3
Figure A 7: Team4 Car Park App String Parameters [7]	A-3
Figure A 8: Team4 Counting App Entrance Coordinate [7]	A-3
Figure A 9: Team4 Counting App Exit Coordinate [7]	A-3
Figure A 10: Team4 Import Parking Data [11]	A-4
Figure A 11: Team4 Import Car Count Data [11]	A-4
Figure A 12: Team4 Show All Parked Entries [11]	A-5
Figure A 13: Team4 Select Hour Range [11]	A-5
Figure A 14: Team4 Select Vehicle [11]	A-6
Figure A 15: Team4 Select Spot [11]	A-6
Figure A 16: Team4 Select Session [11]	A-7
Figure B 1: System Hardware [29]	B-1
Figure C 1: Team4 Python Application Workflow [7]	C-1
Figure C 2: TensorFlow Object Detection API [30]	C-1
Figure C 3: Training Pipeline [30]	C-2
Figure C 4: GUI Flowchart [11]	C-2
Figure C 5: TEMT6000 circuit [10]	C-8
Figure C 6: Structure 5 lux level [10]	C-8
Figure C 7: Structure 3 lux level [10]	C-9
Figure C 8: Primary Key Test: Sample Data in Database [11]	C-10
Figure C 9: Primary Key Test: Sample Data to Transfer to Database [11]	C-10
Figure C 10: Sample Data Used to Test GUI SQL Filters [11]	C-13
Figure C 11: Time Range Filter of Parking Data: 08:00:00 – 09:00:00 part 1 [11]	C-13
Figure C 12: Vehicle Type Filter of Parking Data: Truck [11]	C-14
Figure C 13: Vehicle Type Filter of Parking Data: Bus [11]	C-14
Figure C 14: Vehicle Type Filter of Parking Data: Car part1 [11]	C-15
Figure C 15: Vehicle Type Filter of Parking Data: Car part 2 [11]	C-15
Figure D 1: Full enclosure [14]	D-1
Figure D 2: Enclosure with Nano exposed [14]	D-2
Figure D 3: Full enclosure profile view [14]	D-2
Figure D 4: Team 4 Full Printed Enclosure Design [14]	D-3

TABLE OF TABLES

Table I: Nano vs RPi 3 B+ [2].....	6
Table II. USB Camera vs RPi Cam V2 [3].....	6
Table III. Expenses [12]	14
Table IV. Project Schedule, Milestones and Assignments [15].....	18
Table V. Work Breakdown Tallied Hours [15]	20
Table B I. Prolonged Use Test 1 [10]	B-2
Table B II. Prolonged Use Test 2 [10].....	B-2
Table C I. Runtimes in seconds of GUI to upload 400 entries (left) and 1,000 entries (right) [11]	C-11
Table C II. Runtimes in seconds of GUI to upload 400 entries (left) and 1,000 entries (right) [11]	C-12

Executive Summary

Elevator Pitch — Our project aims to create a cost-effective image recognition system to detect and collect parking data of vehicles in parking spots.

The team began on a journey to find a solution to the parking problem at Sacramento State. During the peak hours of traffic, it becomes increasingly difficult to find an open parking spot, which leads to drivers circling within a parking structure and causing traffic congestion. The original solution the team agreed upon was to create a smart parking system that would direct drivers to open parking spots and improve the flow of traffic. However, after discussions with a professional in the field and our senior design professor, the team came to the realization that this approach was a naïve solution to a problem that was much more nuanced and far too much of an undertaking for three college students without a background in traffic management. The final solution reached was to create an image recognition system, that was cost-effective, and would collect good and accurate parking data, this data would then be given to traffic professionals so that they could create solutions; this project would set the foundation for the solving of our societal problem. The system would also offer a more cost-effective option for data collection that is portable, easy to set up, and can be utilized in remote locations away from external power sources.

The main tasks the system had to accomplish were as follows; identify and categorize vehicles in a parking area at Sacramento State, distinguish occupancy of 4 car parking spots, keep an integer count of identified vehicles entering and exiting the area, run for 10 hours via battery, and have a GUI that will allow the user to transfer the recorded data to a database and display the data collected. These were parameters agreed upon by the team and were the functionality goals set for the deployable prototype. The funding for this project came out of the team's pocket, with the project's budget being about \$1,000.00, we accomplished our feature set with measurable metrics well under our budget.

A few of the important milestones of the project were the successful training of a recognition model that could identify toy-scaled vehicles, integration of the model to a microprocessor in order to run spot monitoring and car counting programs, upscaling to full-scale vehicles, the completion of the power bank and the enclosure, and the completion of a GUI that allowed filtering of the data the user wished to analyze. The workload was broken down between the three teammates with each one working on the aspects they specialized in. The team assessed a few risks that could occur during the project and thought of ways to mitigate them and continue working on the project even in the worst-case scenarios. The report will discuss the big picture of the project build and the status of the deployable prototype, as well as the marketability of the completed project.

Abstract — As Sacramento State student enrollment continues to grow, so will the duration and frequency of traffic jams on campus caused by students searching for an open parking spot. Since developing an elegant traffic management solution is beyond the team’s capabilities, we chose to focus on creating a system that collects parking data for professionals which may allow them to better understand the parking behavior at Sacramento State and develop a solution.

This report documents our data collection system. The system uses image recognition to monitor parking spots in order to record the type of vehicle that was parked, when it was parked, when it left, which spot was it parked in, and how many vehicles have parked in each spot per recording session. The system can also record when a vehicle has entered or exited a parking area, the type of car that entered or exited, and how many have entered or exited per session. The report will discuss the societal problem the team set out to solve, our design idea to tackle said problem. The project funding, milestones, and how the team dispersed the workload. The team assessed some possible risks and how they would be mitigated. The report will also discuss the broad picture of the project and the marketability of the final design. And, the report will discuss the status of the final deployable prototype.

Keyword Index – deployable prototype, funding, milestones, “parking problem”, project, risks, Sacramento State

I. INTRODUCTION

Every semester at Sacramento State there is an increase in students being admitted that is outpacing readily available parking spots. To accommodate for the increase in students, the university has built a new parking structure and is even in the process of creating more

residence halls in order to reduce commuting students. However, students will continue to drive to, and park, on campus as external housing is more affordable. There is also the matter of Sacramento State’s faculty commuting to campus. From professors to administrative staff, most of them need to drive and park on a consistent day-to-day basis. Commuting students may also be inclined to stay on campus despite not having a class scheduled for the day since Sacramento State provides many amenities such as The WELL, so that also contributes to parking spot scarcity and in turn introduces additional traffic on campus from commuters looking for a place to park. So, the combination of increasing student enrollment, need for faculty and staff parking, and draw of on campus commodities there is a large number of vehicles that need to be parked on campus.

Sacramento State only has two entrances, a South and a North entrance, that allows vehicles to enter the campus. So, while a few drivers are stuck looping around a parking area looking for an open parking spot the flow of all the traffic is halted creating giant traffic jams. Drivers can spend around 40 minutes looking for a parking spot. Combined with the traffic from the everyday hustle and bustle, this can cause students to be late to their classes.

Our project aims to solve the “parking problem” at Sacramento State. The original solution was to create a smart parking system similar to that present in parking structure 5 with the added component of direct driver’s directly to the available spot. The system would involve a user interface that would allow for the reservation of a parking spot and directions to said spot, however, we found this approach introduced too much human error and would be extremely costly. A smart parking system like the one present in parking structure 5 utilizes inductive sensors in each parking spot, and these sensors require invasive maintenance. So, the team decided to create a cost-effective image recognition system that could collect good accurate data of vehicles entering, parking, and exiting a

parking area. Our project will collect the data that professionals could analyze to create a solution for the “parking problem” at Sacramento State, it would set the foundation for the solution.

The main function of the project is to collect good accurate data that can be appropriately analyzed by a professional. With some guidance from a traffic professional and the course instructor the team was able to create a specific set of features the system is able to perform. The first feature is the system is able to identify vehicles entering and exiting the parking area as either a truck, a car, a bus, or a motorcycle. The second feature is the system will be able to distinguish the occupancy of four individual single-car parking spots and keep track of the start and end time duration of the vehicle parked in each spot. The third feature calls for the system to keep an integer count of the identified vehicles entering and exiting the parking area. The fourth feature allows the system to be used portably for at least 10 hours. And, the final feature introduces a GUI that allows for the filtering of the recorded database for analyzation purposes.

The project was funded by the team. Each team member paid for the parts that were required for their individual tasks. The most expensive individual part was the NVIDIA Jetson Nano which cost about \$100 dollars and the second most expensive piece was the 3D printing of the enclosure which cost \$50.

As the year progressed and the team continued working many milestones were reached. They mainly revolved around the completion of the project’s features. The biggest milestones were when the system was able to identify scaled down models and later full-scale vehicles, as well as the completion of the enclosure due to the constant delays in the 3D printing process. The approved completion of the project was a huge moment for the team because we had a rather slow start and are only a team of three.

The project could be divided into three main tasks; the identification program, the portability, and the GUI/database. Michael took the command of the system programming. Sergio took care of the power bank and enclosure for the system. Ryan was responsible for developing a GUI to interface with a database that stored collected data.

The team had to take the possible risks, that could delay the project or financially burden the budget, into consideration and think of ways to mitigate these risks. The biggest possible setbacks would be the complete loss of our program and trained models, so we created multiple save files and different save locations. The other major risk is the damaging of the development board that runs our program, so we purchased another unit.

The team believes the product is quite marketable, one of the points our expert consultant informed us of is the fact that the companies that already perform the tasks our project does are very proprietary. Those companies charge the client for the use of the system as well as for the analyzing of the data collected; the client never has access to the data collected and has to pay a large amount of money. Our system would collect the data and allow the client to analyze the data themselves, creating a more transparent process. The system is also cost-effective, so more clients would be attracted to the lower prices of use and the transparency of the data collected.

The system is portable and can be used in remote locations. The camera is connected via a USB cable, so the camera can be placed or mounted in whichever location is more convenient for the user. The enclosure protects the system from the environment and creates a hardy system. The deployable prototype was fully completed with the full functioning of the system and the completion of the enclosure and power bank.

II. SOCIETAL PROBLEM

A. Overview of Societal Problem

Finding an open parking spot can be a frustrating and time-consuming endeavor. Whether it be at the mall, downtown, sporting events, college campuses, etc., a lot of time is wasted just looking for open spots which are often very limited. Strategies like assigned parking or staff physically directing vehicles to spots streamline the parking process. However, assigned parking and in person assisted directing is often an impractical strategy for facilities that provide services to clients whom outnumber available parking and are coming and going throughout the day.

During peak hours of traffic at Sacramento State, dozens of drivers are attempting to enter campus and find a parking spot. Along with an ever-growing admission rate and only two entrances to campus, parking spots fill up quickly and the flow of traffic suffers. The North End, which is off J Street, and the South End, which is off the U.S. 50 freeway. Therefore, circulation becomes a major problem if drivers don't know where to find a parking spot and end up roaming around campus cluelessly. Drivers will enter a parking structure looking for a parking spot and upon not finding one will begin to circle numerous times in hopes of a spot opening; dozens of drivers will circle the parking structure creating traffic jams. If drivers are unlucky then they have to sit through more traffic just to exit that structure and repeat the process in a different parking structure. On the Sacramento campus there is only one parking structure that includes an electronic board that displays the amount of available parking spots in the structure. However, this parking structure is located the furthest from the classroom halls so most students park in a different structure. Parking Structure II and Parking Structure III are where this situation is the worst, since they

are the closest to the U.S. 50 freeway off-ramp every driver goes directly to those locations. Finding an open parking spot and it not being stolen by another driver just as you are approaching it could take up to 30 minutes. Combined with city traffic of the daily hustle and bustle, students could end up missing up to half of their class. This could be detrimental to a student's grade. There is the need for a sense of parking security.

The societal problem our team tackles is the traffic congestion on college campuses caused by people searching aimlessly for limited parking spots. A clear solution for the problem is guidance to an open parking spot; this would help lessen traffic congestion by directing clueless drivers to an unoccupied parking spot. The initial problem statement idea sought to implement a Smart Parking System (SPS) that would allow the driver to reserve a parking spot and would then be directly led to that location. However, this would require a large quantity of sensors and there is no way in which to enforce the reservation of the spot. Thanks to the insight of our technical advisers, the live relay of the available parking spots would not adequately solve the parking issue as there are too many variables to account for in order to develop the system in a way that would efficiently direct people to a parking spot without causing more issues. Thus, the design goal shifted away from focusing on advertising open spots to people looking to park. Instead the team decided to develop a system that would collect data on parking statistics which can be used by professionals more attuned with traffic management, then the team is, to develop a sufficient method to streamline the parking process for all.

B. Background and Design

With the help of transportation engineer and traffic expert Professor Ghazan Khan, we were able to develop a more obtainable solution. Originally the team thought that with the use

of sensors and image recognition we will create a system to notify drivers of the available parking spots in a parking structure. Our team was advised by our technical adviser Professor Russ Tatro, and Professor Khan, that alleviating the parking issue on campus would require more than just simply advertising the available spots. For example, if everyone looking for a spot was notified of an available spot, a large influx of traffic would flock to said spot and, aside for one person, would find someone got there before them. This would be an insufficient solution to a very complex and nuanced problem. Instead, our team's goal is to create a cost-effective method to monitor parking spots and collect statistics on their occupancy. The collected data would then provide the means for professionals in the field of traffic management to develop an adequate solution to the parking issue. The team then sought out ways to collect parking data and find the most cost-effective and practical methods.

Through research, inductive loop sensors are among the most widely used methods for vehicle detection today. These sensors are embedded under the pavement of entrances, exits, and parking spots of parking structures. It typically yields an accurate vehicle count. Figure 1 shows us a schematic of how loop detectors work.

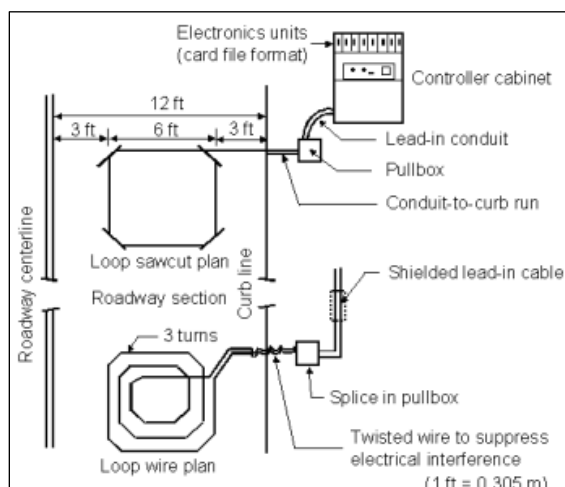


Figure 1: Schematic of Loop Detector [1]

When a motorized vehicle passes over the sensor, the metal underside of said vehicle will gloss over the loop wires causing an induction which translates to current flowing through the wires. This sends a signal to the control system as a pulse. And while our system will also keep count of vehicles entering and exiting the parking area, we intend to take more detailed data than just a number count. Whether it be a car, truck, motorcycle, or a metal cart, there is a chance that unintended objects will trigger the loop detector. This will contribute to the counting of vehicles making the system inaccurate. Several factors affect the reliability of the loop detector output data such as traffic density, vehicle movements, traffic composition and physical characteristics of the intersection such as pavement condition. Depending on the traffic density and traffic composition, the loop detectors may produce a constant pulse simulating a very long vehicle.

These inductive sensors are costly electronic components and are also extremely expensive to install and maintain due to their invasive installation method. They must be implemented into the ground so installing them into already built locations would require tearing up the ground, and the same goes for maintenance. The process of digging them out of the ground, conducting maintenance, and then reburying them and patching up the ground would require a good amount of time. During this time there would be lesser parking spots available, adding to the problem that is being tackled.

The team intends to create a cost-effective, accessible, and noninvasive method of surveying. Thus, the team decided to develop a system that will use a camera and a development board in order monitor and collect parking statistics data in a parking area. With the use of a camera, the project avoids the costly inductive sensors and the high maintenance that comes along with them.

The system will collect data on parking spot occupancy in the parking area as well as

collect data on vehicles that will be entering and exiting a parking area. The system needs to be portable and be able to be placed in remote locations where there is no power source available. The system will include a GUI to allow the user to access and analyze the data collected. The camera connection method will allow the use to locate the camera in the position most convenient to them as long they have the right angle for the video.

Professor Khan informed the team of the proprietary nature of the data collected using other companies' equipment. The user does not have access to the data collected since the company analyzes the data themselves. Our system will allow the user to analyze the data collected themselves and can see the true colors of the traffic behavior. This access will allow the professional a better understanding of the traffic in the area.

C. Benefits

Our system is veered towards helping professionals solve the "parking problem" at Sacramento State. By helping solve the "parking problem" we are also helping the students affected by terrible parking situations. Professionals can do their job more efficiently and a student will have a much easier time finding parking.

The professionals that use our system will gain a much better understanding of the traffic behavior in certain parking areas. The system's database will be accessible via a GUI and the user is able to analyze the data they wish to analyze. Unlike companies that treat their data in a proprietary manner, with our system, the user will have full access to the data and can analyze it in any way they wish.

Our consultant informed us that companies charge them for the use of their system and then charge them again for analyzing the data collected, and the user never has access to the raw data itself. Our system would offer a lower price option, making it more accessible

to the user. This would in turn encourage the user to tackle the "parking problem" more willingly and could help solve the issue faster. The accessibility would also encourage starting engineers to tackle the "parking problem" and offer a wider array of possible solutions. This would benefit both young entrepreneurs and students affected by the parking problem.

III. DESIGN IDEA

A. Resources

Initially, the team assumed that since our project is mostly software related there are not a lot of parts needed for a physical build of the system. However, as the team began working, it became apparent, through trial and error, that there would need to be changes to the parts needed to complete the design. The team first started with a Raspberry Pi board with a Raspberry Pi camera. The system needed a lightweight but powerful processing unit and ARM based processors fit our requirements. The reason our team used an ARM based unit was because it functions well as a small portable processor which can operate at low power. It functioned as the central processor of the system, computing all the processes in our design. Through training various models our system used machine vision to process the video feed. Running detection models with these components proved to be insufficient, as the computational power of the Raspberry Pi was not fast enough to keep up with the needs of the program. The team decided on using the NVIDIA Jetson Nano development board to run the detection model. The Jetson Nano has a much more powerful GPU.

Table I.
Nano vs RPi 3 B+ [2]

	Jetson Nano	RPi 3 B+
Processor	Cortex-A57	Cortex-A53
CPU Cores	4	4
CPU Speed	1.43 GHz	1.4 GHz
RAM	4GB LPDDR4	1GB LPDDR2
GPU	Maxwell	VideoCore4
GPU Cores	128	12
GPU Speed	921 MHz	400 MHz
Theoretical Peak Performance	247.216 GFLOPS	24.8 GFLOPS
Package PWR	20 Watts	12.5 Watts
Dimensions (mm)	100x80x29	85x56x17
Cost	\$99	\$50

As you can see in the table above, the Jetson Nano outperforms the Raspberry Pi 3 B+ given similar power design. Both microprocessors have 4 cores and clocked at around 1.4 GHz. The big advantage of the Jetson Nano is the 4 GB of ram and dedicated 128-core Maxwell graphics processing unit. The Raspberry Pi does not have a dedicated graphics processing unit. Because of this, the Jetson Nano outperforms the Raspberry Pi in floating point performance by 10-times! Again, all within a small package power of 20 watts. Another advantage of the Jetson Nano is to use the default power saving mode which brings the package power down to just 5 watts. This brings down the core clocks speed of the CPU from 1.479 GHz to .918 GHz, and the GPU from .9216 GHz to .640 GHz. [2] But even with this slight decrease in performance, the Jetson Nano can still outperform the Raspberry Pi by about 8-times.

The team also became aware of the risk of over-heating the board. Image recognition, and thus image processing requires a lot of computational power, which in turn develops a lot of heat. Therefore, the team needed to consider a way to physically cool the system when it is operating. A cooling fan was added

to the physical build of the design in order to address this. When we built the laboratory prototype, we purchase a chassis and it came with a no-name/no-brand 5V 40x40mm generic fan. The performance of this fan was suitable for our needs, but we decided to replace the fan with a Noctua NF-A4x20 PWM 5V 40x40mm PWM fan. This fan was exceptional and gave us 5.5366 CFMs of airflow at 5000 RPM with an acoustical noise level of 14.9 db(A).

Initially, our team was also contempt with using a Raspberry Pi Camera V2. After testing in our laboratory prototype, we found that this camera was insufficient. The length of the ribbon cable also made it difficult to maneuver the camera around if we wanted to place it in a desired angle. This was changed to a USB camera. The advantage of switching to a USB camera is not only the quality of the sensor, but the fact that the camera is able to attach and detach with a single USB cable. This allows us to extend the camera to a desired length and mount that camera onto a stand which we can position it in any way we want. With the larger sensor size and the wider viewing angle, we can get the maximum performance in barely lit environments and see wider areas compared to the RPi Camera.

Table II.
USB Camera vs RPi Cam V2 [3]

	USB Camera	RPi Cam V2
Sensor	CMOS OV2710	Sony Exmor IMX219
Sensor size	1/2.7"	1/4"
MegaPixel Size	2	8
Max Resolution	1080p	4K
Max Framerate	30@1080p 60@720p 100@480p	15@4K 30@1080p 60@720p
Focal Length	1.56mm	3.04mm
Viewing Angle	170 degrees	62.2 degrees
Connector	USB 2.0	CSI
Price	\$50.99	\$27.50

For hardware resources, the team acquired: the NVIDIA Jetson Nano, the Noctua cooling fan, the components for a power bank, the charger for the power bank, the USB camera, the flash drive, the laptop, the 3D printed enclosure, and other miscellaneous materials needed for developing to complete the project. The acquisition of these units, combined, cost around \$400.

Python is the language the team utilized for the car counting program and car occupancy of a parking spot program. The advantage of Python is the implementation between the program and the hardware. When communicating to hardware components, Python libraries allow us to easily request from device drivers. Python also has necessary libraries to accomplish our features. Python allows us to use TensorFlow in tandem with OpenCV. These software libraries ran seamlessly in our operating system environment for the NVIDIA Jetson Nano. More details of the software implementation can be found in subsection C. MySQL was used for our database and a GUI was developed to allow for the upload and display of data collected by the system. The GUI was developed with Python and SQL.

B. Feature Set

There are five essential features our system will adhere to in order to provide a more cost-effective method to solve the “parking problem” at Sacramento State. The first feature is the ability to detect and classify motorized vehicles. The knowledge of the type of vehicle entering a parking area can help predict if students might be carpooling and/or the demographics of students attending the University. The next feature is the ability to distinguish occupancy of a single standard car parking space. By collecting this information, it is possible analyze the behavior of certain parking spots; will the spot be occupied at a certain time during a certain day? The next

feature requires the system count vehicular traffic. Keeping track of the vehicles entering and exiting a parking area provides insight on traffic flow. Another feature requirement is the system will include a power bank for portability. It is important for the system to be able to be used in remote areas that don’t provide power sources. The final feature is the system’s database will be accessible via a GUI. The GUI will allow the user to access all the parking data that has been collected.

C. Identification

The ability to identify vehicles is a crucial part of our system. There is a plethora of vehicles in a parking area at any given time. Knowing what kind of vehicle is present can provide a great amount of insight to the community. Parking areas at Sacramento State offer regular size parking spots, compact parking spots, very few motorcycle parking spots, and parking area for busses transporting students.

If a parking area experiences a larger amount of trucks to cars, then there would be no need to provide compact parking spaces. Or perhaps there is a deficit of parking spots in comparison to the amount of motorcycles arriving on campus. The information could be used to create targeted parking areas and possibly improve the flow of traffic by eliminating the uncertainty of where to park.

We chose to utilize machine vision in our project as we want to simplify the number of external sensors in our project. Machine vision allows us to utilize image data and run it through a computer neural network to obtain necessary parameters for the application of our project. Neural networks consist of an artificial network inspired by the 2 human brain which allow the computer to learn and fine tune itself based on analyzing new data [4].

The projects machine vision portion uses a pretrained model optimized for object detection. The model used in our project is the

SSD Mobile Net Version 2 pretrained with the COCO large dataset. This model uses the SSD algorithm which stands for Single Shot Detector. SSD runs a convolutional network on the input images once and then calculates the feature map [5]. Then a small 3x3 sized convolutional kernel is ran on the feature map to predict the bounding boxes.

The team chose the Nvidia Jetson Nano development board as it runs a powerful quad-core ARM A57 microprocessor with a 128-core Maxwell GPU all on a 5-watt power design. Our 2-megapixel USB camera a CMOS OV2710 sensor with a sensor size of 1/2.7". These components run a special Linux Ubuntu operating system environment for ARM processors. The operating system is provided by Nvidia. As for the framework, the applications are written on Python 2.7. The machine vision portion of the application utilizes Tensorflow-GPU 1.14.0 and OpenCV-

Python 4.1.2.30. The components used for the development of the program is an Intel Core i7-9700k, 16 GB DDR4 ram, and a Pascal based GPU: Nvidia GTX 1080 ti.

Tensorflow is the framework for our machine vision program. Tensorflow provides a dataflow graph or computational graph for the manipulation of data. It is used to create mathematical symbols in the form of Tensors useful for machine learning. OpenCV is a library of programming functions used 3 for real time computer vision. It incorporates features to do all operations related to images. These libraries worked in tandem with Python. If it weren't for libraries like these, creating functions that operate with IO, in this case our camera, would have made the project more difficult and time consuming. These libraries accelerated the programming for the parking detection and car counting applications.

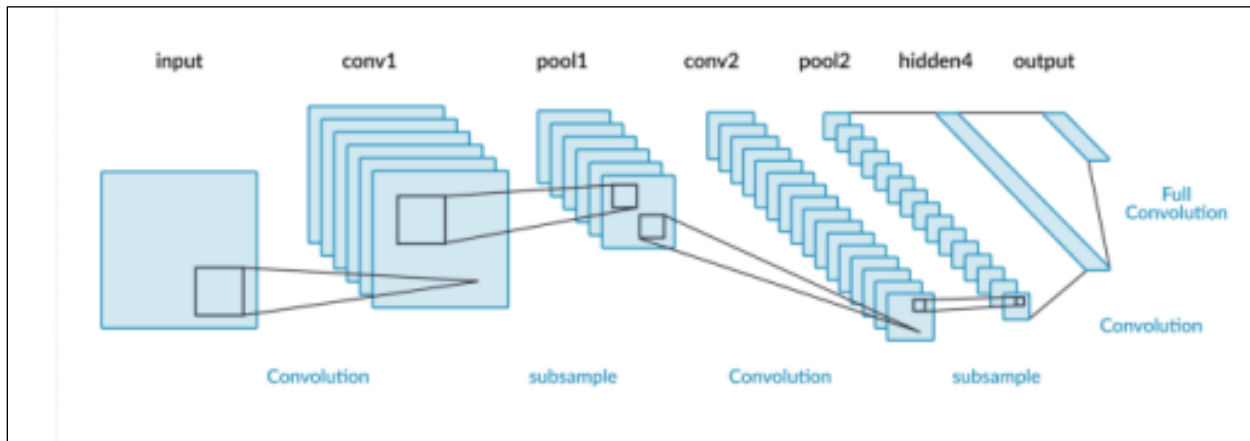


Figure 2: Example of a convolutional neural net, LeNet-5 [6]

D. Occupancy

The goal of the parking detection feature is to detect if an individual parking spot was occupied and keep track of the duration of the parked car. This would have to work for multiple parking spots with multiple cars in the frame of the camera. The punch list of the project stated that at least 4 parking spots are to be monitored. These individual spots would have separate data written into a single database file. This data file would be moved onto a flash drive and transferred to a computer with the database reading program to be analyzed by the user.

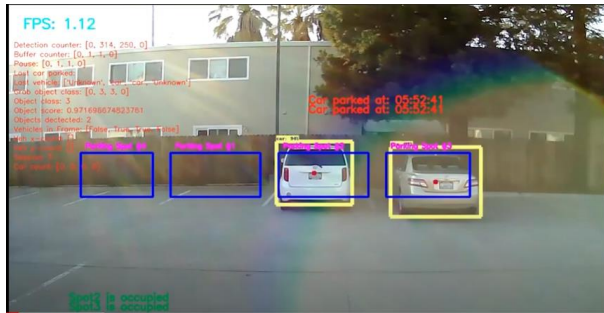


Figure 3: Multiple parking detection program [7]

The parking application was written in Python. The program would first have to use the machine vision element to detect a motorized vehicle. When the machine is confident enough and meets the confidence levels set by the programmer, the data of the detected vehicle would be stored in four separate arrays: bounding boxes, confidence levels (or scores), object classes, and number of detections in frame. The first dimension of each array accounts to one detected object. To obtain the parking of a single parked vehicle, the program checks the first dimension of the scores array and keeps a copy of the index if the machine vision is confident that the object being detected is a motorized vehicle. With the saved index number, the program then checks the bounding boxes array and grabs the coordinates of the detected object. A simple

algorithm gets the center of the bounding box of the detected object and draws a small circle. If that circle is within the imaginary parking spot, the program runs a counter and counts to 15 frames. If the detected vehicle is still within parking spot after 15 frames, the program counts the detected object as parked. At this point, the program grabs the system time as its start time. When the detected object leaves parking space, another counter buffers till 30 frames has passed. The program again reads the system time and outputs it as the leave time for the detected vehicle. Finally, the program writes this data onto a plain text file on a flash drive. The flash drive can be removed to transfer the data to our database.

E. Car Counting

A punch list feature of our project is to have the system count motorized vehicles entering and exiting parking areas. If a car enters a parking area, a variable is incremented representing an integer count of the total cars in that parking area. The opposite is done when a car leaves the parking. This is done with a separate Python application that utilizes the same machine vision algorithm that is used in the parking detection application. An imaginary line is created at the desired area instead of an imaginary parking spot. If the program sees that the detected vehicle is on one side of the line, a counter increments up to 5 frames. After 5 frames, if the same detected vehicle crosses the imaginary line, the program will count that as a vehicle entering or exiting a parking area. This is represented by an integer number on the screen. It will then increment a car count value and write to a text file.

The program runs on the NVIDIA Jetson Nano using the SSD MobileNet v2 neural net to run image classification. It uses the same dependencies to run the program and the initial base code of the parking program as the framework for the logic.

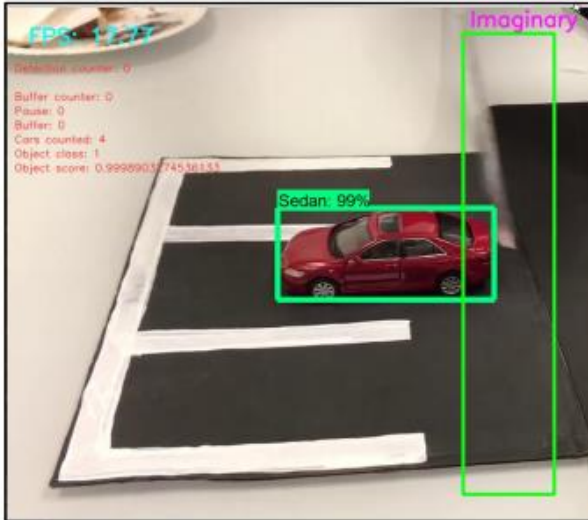


Figure 4: Prototype car counting program [8]

F. Power

The system requires a power source that is easily portable, so our solution was a rechargeable battery pack. We employed twelve Samsung 25R 18650 batteries; each cell had a nominal voltage of 3.6 v and a nominal capacity of 2500 mAh [9]. So, to reach a desirable voltage and capacity, 2 cells were soldered in series and 6 cells were soldered in 2 parallel; this configuration allows for a nominal voltage of 7.2 v and a nominal capacity of 15000 hours. These 18650 batteries are rechargeable lithium ion cells. The team chose to use these cells because they have superior capacity and discharge rates. They offer 300-500 charge and discharge cycles before the cells deteriorate too much. These are the cells that are many times used in laptops.



Figure 5: Power Bank [10]

Nickel strips were used to solder the 18650 cells together. Nickel has a low relative resistance which means low heat and low energy wasted. A BMS board is used to protect the cells when they are being charged and discharged. The BMS has a cut off voltage of $2.5 \text{ v} \pm 0.7 \text{ v}$ and works through passive balancing; burning off excess power as heat through the resistors mounted on the BMS board. The charge balancing begins when the safe voltage threshold of $4.2 \text{ v} \pm 0.025 \text{ v}$ is reached for either cell.

The power bank has a voltage regulator and a voltmeter whose voltage requirements line up with the cut off voltage of the BMS. And, a female barrel DC jack for charging. Our system operates at 5 watts of power; tests proved the power bank can successfully power the system for 14 hours, easily surpassing the 10 hours required by the feature set. The power bank is also highly portable and allows the system to be highly portable as well.



Figure 6: Power Bank Switches [10]

G. Database and GUI

The GUI feature is used to interface with a MySQL database. It was developed in Python using the tkinter library. To transfer data through the GUI, the user must take the flash drive from the microprocessor, after it transfers the data into a text file, and plug it into a computer that has the GUI program. Upon startup of the program it has button that will transfer the data to the MySQL database. In order to accomplish this, the mysqlconnector library is used. This library allows the program to connect with a MySQL database. It also allows the program to interact and manage the database using SQL code. The mysqlconnector library was essential in developing the GUI as its function is to transfer data to and from the MySQL database. MySQL is a relational database, which means it is a database structured to recognize relations among stored items of information.

ID	Enter/Exit	Time Stamp	Veh Type	Count	Session
1	out	9:40:22	car	1	1
2	out	9:43:10	car	2	1
3	in	9:47:31	car	1	1
4	in	9:48:34	car	2	1
5	in	12:33:18	bus	1	2
6	out	12:33:27	bus	1	2
7	in	12:37:26	truck	1	3
8	out	12:37:34	truck	1	3

Figure 7: Database [11]

The MySQL database has a table that stores the parking data. It has 8 columns of data to store for every entry. The “EntryID” column is an integer type value and is the primary key. In a relational database model, every entry should have a unique value in which to differentiate itself to prevent insertion anomalies, like inserting the same data multiple times. The “EntryID” column must have a unique integer value to differentiate all the entries from each other. The “Spot” column is an integer type value. It describes what spot a vehicle parked in. The “Entry Time” column is a time type value. The time type allows the database to store time-based values. It is formatted as, “HH:MM:SS”. This column stores data on what time a person entered a spot and parked their vehicle. The “Exit Time” column is also a time type value. It stores data on what time the person who parked their vehicle moves and leaves their spot. The “Veh Type” column is a variable length string type value. It stores information on the categorized vehicle that was parked. The “Count” column is an integer type value. It stores information on how many vehicles have been parked in a spot. The “Session#” column is an integer type value. It stores the session the system is running on. Every time the system begins collecting data and then finishes and writes to the flash drive, it increments the session. Finally, the “Park Duration” column is a time type value. The “Park Duration” column displays an entry’s difference between the “Entry Time” and “Exit Time”. Essentially, it shows how long a vehicle was parked.

The database also has a second table to store data aggregated by the car counting program. It has 6 columns of data to store for every entry as seen in Figure 7 [11]. Most of the columns are similar to the parking data table. Just like the parking data table, it has an integer primary key, “ID”. The “Enter/Exit” column denotes whether or not a vehicle has entered or exited a user defined area in the car count program. This column is a variable

length string type. The “Time Stamp” column denotes the time an entry has either entered or exited the user defined area. It is of type time. The “Veh Type” column denotes the type of vehicle recognized when it entered or exited the user defined area. It is also of type variable string length. The “Count” column denotes the amount of vehicles that have been counted for entered vehicles or exited vehicles. This value resets upon a new session start of the car count program. It is of type integer. The “Session” column denotes a data collection period of the car count program. It is of type integer.

The GUI also allows the user to look up specific information in the database. Using SQL code, the database can be queried in a variety of ways without the user knowing SQL. With a push of a button, the database entries can be filtered to show only entries of a particular vehicle type, entries that have parked in a specific spot, entries that have parked in a particular time range, and entries logged for a particular session.

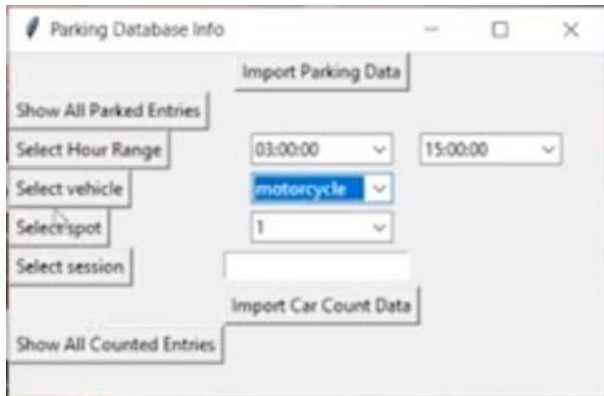


Figure 8: GUI [11]

H. Integration

All of the individual features come together to form the completed project. The project’s machine vision portion uses COCO, a pretrained model optimized for object detection. The camera feeds the microprocessor image data and a program interprets that data via an algorithm to

accomplish object detection and object classification. For software integration, the data obtained had to be carefully organized from the parking detector application before it was outputted onto the database file.

The microprocessor stores the data on a flash drive in a text file. The flash drive is then plugged into a computer that has the GUI program and MySQL database. The database uses 2 tables to store data collected from our microprocessor and stored to a flash drive. It has 8 columns of data for the parking program and 6 columns of data for the car counting program. The “EntryID” column is an integer type value and is the primary key that uniquely defines their respective row of entry. The GUI allows the user to manipulate and transfer data with the MySQL database without knowing SQL. It can filter the database to show only a particular vehicle type, spot, or time range.

Not every location will have a power source within reach. Our power bank is necessary for those remote sites. In order to collect enough relevant data, the system needs to be able to operate for hours on end. The power bank provides at least 10 hours of power. The power bank also contributes to the system’s portability, giving the user the ability to move the system from one location to another without the need to shut it off and relatively low effort.

Our project aims to collect and provide parking statistics to professionals in the traffic management field. These professionals can use this data to solve the issue of traffic congestion caused by limited parking. Our system will provide this data by using machine vision running on an NVIDIA Jetson Nano microprocessor powered by an external battery pack.

IV. FUNDING

All of the components needed for the project were purchased by the team members. Each team member purchased the components that

were necessary for their respective part in the overall project. The total dollar amount spent on the project was \$536.00. This was well below our budget of \$1000. Michael spent \$176.50 on the project, Sergio spent \$159.50 on the project, and Ryan spent \$200. Our spending for the project was relatively equal in dollar amount.

If we were to add the cost of the parts used in the deployable prototype, the cost would be \$334.50. These parts include the NVIDIA Jetson Nano, Noctua fan, SD card, batteries, electrical components, 3D printed parts, and other miscellaneous parts to complete the device. Table 1 below will list the team's expenses.

Most of our spending was during our first phase of the project year. We experimented with what we thought our project needed while keeping in mind of completing the laboratory prototype. You can see in the table that we purchased some toy cars. Although not vital for the completion of the project, this helped us making a controlled environment to test our features with. It wasn't until the second phase of the project year where we started to purchase necessary parts for the device itself. Around this time was when we decided to purchase the USB camera to switch over from the Raspberry Pi Camera. We also bought a second NVIDIA Jetson Nano for Ryan and Michael to do software development separately. We also bought this just in case we fried the first Jetson Nano as a safety measure.

Table III.
Expenses [12]

Item	Quantity	Price per unit	Purchaser
Raspberry Pi 3 B+	1	\$35.00	Michael
Raspberry Pi Camera V2	1	\$27.50	Michael
Noctua NF-A4x20 PWM V2	1	\$15.00	Michael
NVIDIA Jetson Nano	2	\$100.00	Ryan
NVIDIA Jetson Nano Case Holder w/ Fan	1	\$15.00	Michael
SD Card for Jetson Nano	2	\$10.00	Michael
Samsung 25R 18650 Rechargeable Batteries	12	\$4.00	Sergio
USB Camera	1	\$50.00	Michael
3D Printed Parts for the enclosure	1	\$45.00	Sergio
LM2596	1	\$12.00	Sergio
DC Voltmeter	1	\$13.00	Sergio
2S BMS	1	\$11.50	Sergio
Misc Parts (wires, tape, etc.)	n/a	\$30.00	Sergio
Toy Car	1	\$4.00	Michael
Toy Bus	1	\$3.00	Michael
Toy Truck	1	\$4.00	Michael
Toy Motorcycle	1	\$3.00	Michael
		\$536.00	Total Cost

V. PROJECT MILESTONES

Throughout the two semesters, the team spent many hours working hard on our system that collects parking data using machine vision. We hit many milestones showing the progression of the project and completing these milestones helped encourage the team to reach the next milestone in order to meet the feature set's measurable metrics. The following section describes the project milestones achieved by the team that lead to the completion of the deployable prototype. Each paragraph describes a milestone and the impact on the project's development.

The first milestone was the agreement on a societal problem the team would work to solve. The team was assigned to seek out and study a societal issue on their own. At this moment we decided what we would dedicate our efforts towards for the next year. We resonated with the "parking problem" at Sacramento State and decided to find a way to solve this problem. This was the beginning of our brainstorming and research into a concrete solution the team can implement and design.

The next milestone was the forming and establishment of the design idea. It was at this moment that we established our sandbox and set the goal we were aiming for. The team wanted to create a system that would use machine vision to collect parking data in order to help solve our societal problem. Initially, the team also wanted to implement a parking guidance system that provided information on which spots were open and where to park. However, after consulting with Professor Tatro and researching more into the parking problem, and traffic congestion management in general, it became clear that simply telling drivers of open spots would cause problems. On Professor Tatro's referral, the team met with Professor Ghazan Khan, a Civil Engineering faculty member here at Sacramento State. Professor Khan informed the team that the main focus of the project should be on data collection of parking

statistics using machine vision. After various revisions of our feature set the team finally created an adequate punch list of features and measurable metrics. With the design idea and punch list of features in focus, the team began researching the resources and parts needed to bring our solution to the parking problem into fruition. The team decided to use a classification model running on a microprocessor, a camera, and a power bank to collect parking data statistics and store the information onto a database. The team finally had a clear vision of the project. And then the time finally came to begin working on the lab prototype.

As the team worked toward assembling a functioning lab prototype, the necessary components began to get completed. One of the first milestones, in terms of the project's concrete development, was the training and integration of the toy-scale classification model. We trained a classification model to detect and classifies toy-scale models of vehicles. The model was able to successfully identify scaled down models of vehicles. The lab prototype consisted of model cars, and the system operated successfully in good lighting. We later discovered the camera we were originally using had a low resolution, so a new camera fixed the issue. The data collection programs can now be used to collect data.

After the achieving the previous milestone, we were able to implement a car counting program and a spot occupancy monitoring program to collect statistics on parking spots on the microprocessor. The car counting program would count vehicles that have entered and exited a bounding box and collect the time and vehicle type. The parking spot monitoring program would monitor a parking spot and collect the time a vehicle would enter the spot, when it left, and what type of vehicle it was. These programs would write to a text file stored on a flash drive which would be used to store data onto a database on another computer.

The next milestone was the assembly of the power bank and its functionality when integrated with the development board. The power bank was able to successfully power the system. Various modifications were made to make the power bank look cleaner and power the system more reliably.

The team completed a GUI that displayed the data collected from the system and was able to store the data into a MySQL database. The GUI was rudimentary, and it was not easy on the eyes when it came to display the data, but it displayed everything collected in the database that was collected from the system.

Our laboratory prototype was nearly complete, but not entirely. We presented the laboratory prototype to Professor Tatro and he informed us the system was not fully there. We had presented a mostly complete laboratory prototype through video but because we were not prepared to do a live demonstration, it showed we were not up to speed. Fortunately, Professor Tatro let us into the next semester with a provisional pass; we had to prove our laboratory prototype again the first week of the next semester. We were able to fix these issues before the showcase and Professor Tatro received an email from Schilling Robotics complementing our project.



Figure 9: Laboratory Prototype [13]

The next milestone for our project was fully demonstrating our laboratory prototype for Professor Tatro. The team demonstrated the system capturing data on a toy-scale parking lot with toy cars with the parking monitoring program. The data collected was then transferred from a flash drive attached to the developer board and plugged into a laptop that ran a GUI to transfer the data to the database at the click of a button. At this point, the project has achieved the completion of a laboratory prototype.

It's the beginning of a new semester and the team has revised the societal problem. Throughout the previous semester and thanks to consulting with a professional we gained a better understanding of our societal problem. We learned we couldn't directly solve the problem, but we could offer the foundation for solving the problem. This milestone only solidified our team's understanding of the societal problem and reassured us that our project's feature set was properly in place to address the societal problem.

By utilizing the common objects in context (COCO) dataset, we were able to use our system on full-scale vehicles. We finally moved out of the lab and into the real world. We began tests of live footage using our cars and found the new camera was an incredible addition. The system now worked with real vehicles, an important milestone to accomplish as the team can now collect data with the car counting program and spot monitoring program.

The next milestone was the upgrade to the car counting and parking spot monitoring program. In the laboratory prototype build, only 1 spot could be monitored in the parking spot monitoring program. For the deployable prototype build of the parking spot monitoring program, it can monitor up to 4 spots. The car counting program was updated to keep track of which session the program is running on. Achieving this milestone meant that our

system's collected data can now be transferred.

The GUI was upgraded, and it now allowed for filtering of the data collected and stored in the database. The parking data could be filtered by time range, vehicle type, parking spot, and session. The display of the data was also changed for better readability.

There were many delays to the enclosure due to constant failure of 3D printers and then the stay-at-home order announced by the state during the project's development. But the enclosure was finally completed, and all the components were combined to create our deployable prototype. With the enclosure done we were also able to finish our testing.



Figure 10: Enclosure [14]

We made a video presentation for our deployable prototype mid-term progress review and sent it in to Professor Tatro. Our presentation fulfilled most of the feature set requirements, but our team was informed that our demonstration was missing the classification of a bus, truck, and motorcycle; the demonstration only showed cars. None of the team members owned a motorcycle, bus, or truck so the team had to go out and search for some to use in the deployable prototype

review. This proved difficult due to the state issued stay-at-home order, but we were able to find each vehicle to demonstrate the classification of buses, motorcycles, and trucks. The team sent the video presentation with the new footage showing the classification of a bus, motorcycle, and truck to Professor Tatro and fulfilled the remaining requirement of our project aside from the end of project documentation.

With all the milestones outlined in this section achieved, the team has finally completed the physical aspect of the deployable prototype. Despite the shaky start the team met all the features on the feature set. The completion of the deployable prototype was the culmination of accomplishing the many milestones listed in this section.

VI. WORK BREAKDOWN STRUCTURE

Once the team decided on the societal problem to address and the design idea contract was established, tasks were assigned to each member that best suit their respective backgrounds and skills. The work breakdown structure is intended to portray the division of work amongst the team in order to complete the deployable prototype.

Table IV.
Project Schedule, Milestones and Assignments [15]

Features & Assignments	Task	Time Frame	Team Member(s) Assigned
Car Recognition			
A. Machine Learning (Detection Model)	A.1 Research A.2 Training A.3 Implementation A.4 Optimization	10/14/2019 – 11/10/2019	Michael/Ryan
A. Car Counting Method	B.1 Lab Prototype Program B.2 Implementation	11/4/2019 – 11/13/2019	Michael/Ryan
B. Car Occupancy Detector	C.1 Lab Prototype Program C.2 Implementation	11/11/2019 – 11/22/2019	Michael
C. Scaling	D.1 Scale detection model to real vehicles D.2 Scale car counting software D.3 Scale car occupancy detector software	1/27/2020 – 3/20/2020	Michael/Ryan
Display Software			
A. Database backend	A.1 Database Type A.2 MySQL database setup A.3 Lab Prototype GUI	11/4/2019 - 1/23/2020	Ryan
B. GUI frontend	B.1 Auto-copy data file from USB B.3 Display data from database B.4 Filter database entries	1/27/2020 – 3/20/2020	Ryan
System Hardware			
A. Camera	A.1 Hardware Implementation A.2 Software Drivers	10/22/2019 – 10/22/2019	Michael/Ryan
B. Network	B.1 Network Driver B.2 Remote desktop connection	2/17/2020 – 2/23/2020	Michael
C. Power Bank	C.1 Design C.2 Construction C.3 Integration	10/22/2019 – 11/6/2019	Sergio

Features & Assignments	Task	Time Frame	Team Member(s) Assigned
D. Power Bank Optimization	D.1 Testing D.2 Improvement D.3 Integration with hardware enclosure	1/27/2020 – 3/20/2020	Sergio
Physical Hardware			
A. 3D Enclosure Design	A.1 Outward enclosure design A.2 Inward mounting mechanism A.3 Frame mounting	1/27/2020 - 2/24/2020	Sergio
B. Enclosure Printing	B.1 Evaluation of 3D design B.2 Optimization B.3 Printing	1/27/2020 - 2/24/2020	Sergio/Michael
Course Assignments			
1. Problem Statement		9/2/2019 – 9/23/2019	All
2. Design Idea		9/23/2019 – 10/7/2019	All
3. Work Breakdown Structure		10/7/2019 – 10/21/2019	All
4. Project Timeline		10/21/2019 – 10/28/2019	All
5. Risk Assessment		10/28/2019 – 11/4/2019	All
C. Technical Review		11/18/2019 – 12/2/2019	All
D. Revised Problem Statement		1/20/2020 – 1/27/2020	All
E. Device Test Plan		1/27/2020 – 2/3/2020	All
F. Market Review		2/3/2020 – 2/24/2020	All
G. Feature Presentation		2/24/2020 – 3/2/2020	All

Features & Assignments	Task	Time Frame	Team Member(s) Assigned
H. Deployable Prototype		1/27/2020 - 2/24/2020	All
I. Midterm Progress Review		3/2 – 3/23/2020	All
J. Deployable Prototype Review		3/23 – 4/20/2020	All

Table V.
Work Breakdown Tallied Hours [15]

Feature/Task	Ryan	Michael	Sergio	Total Hours on Feature/Task
Problem Statement	11.5	13	14.5	39
Design Idea	15	17.5	19	51.5
Work Breakdown Structure	16	14	18.5	48
Project Timeline	14	15	18	47
Risk Assessment	16	9	5	30
Power Bank	0	0	43	43
Train & Integrate Toy-Scale Classifier	10	55	6	72
Lab Prototype Spot Detection	5	22	0	27
Lab Prototype Car Counter	5	11	0	16

Feature/Task	Ryan	Michael	Sergio	Total Hours on Feature/Task
Lab Prototype GUI & Database	44	0	0	44
Technical Review	2	3	2	7
Revised Problem Statement	8	7	5	20
Device Test Plan	9.5	4	9	22.5
Integrate Full- Scale Classifier	0	12	0	12
Market Review	13	9	9.5	31.5
Deployable Prototype Spot Detection	14	29	5	44
Deployable Prototype Car Count	7	12	2	21
Deployable Prototype GUI & Database	47	0	0	47
Enclosure	0	22	42	64
Feature Presentation	13	11	6	40
Deployable Prototype Presentation & Documentation	40	40	60	140
Total Hours	290	305.5	264.5	860

VII. RISK ASSESSMENT

The process of developing a portable system that extracts traffic data from video footage, specifically in regards to parking statistics, is wrought with many risks, especially for a team of computer engineering and electrical engineering undergraduate students with little to no background experience in the field of traffic data collection and image recognition. With minimal collective experience amongst the development team, identifying and mitigating the potential events and risks that may hinder the completion of the project can be difficult. However, amongst the planned system's features, the team has identified the project's most significant critical path to be the development of the image recognition. Thus, the development needs of the project's image recognition feature will be priority and the development process of the feature will be closely reviewed for foreseeable risks, which will be further explained later in this report. However, the biggest risks to the image recognition feature will be those which are unforeseeable due to our limited knowledge of the technology. Unforeseeable risks will become the primary focus of the team if they manifest during the development of the image recognition feature. The creation and execution of a practical risk mitigation plan will have to occur as soon as possible in order to allow for the completion of the project. There are also risks outside the development of the image recognition feature, like potential mounting failures of the system or limited testing due to an inability to acquire permission to mount the system in the right environment. This section will present and discuss the potential risks associated with the development of a portable system that extracts traffic data from video footage, and possible mitigation strategies if said risks do occur.

A. Critical Path Risk Mitigation

The critical paths of the project will revolve around the development of the car recognition

feature; identifying and mitigating risks to this feature will be crucial to the project. The car recognition feature will use machine learning to allow the system to categorize vehicles, monitor the occupancy of parking spots in a parking garage, and count vehicles that pass these spots. It goes without saying, if this feature does not work the project will be considered a failure. Thus, identifying and mitigating risks which will hinder the development of this project must be top priority. In context of the research and work done by the team on this feature so far, we have identified and even partially pre-mitigated a few foreseeable risks had they come to fruition.

One risk the team already confronted is the potential inadequate processing power and memory of one of our microprocessors. This is a risk to the project's development and even the critical path of the project because if our processor cannot adequately process video footage it will aggregate inaccurate data which defeats the purpose of a parking garage traffic data collection system. Originally the team had been using a Raspberry Pi 3 as the microprocessor, but as the team tried to implement a few basic image recognition programs, it became apparent that the GPU and limited memory lead to poor framerates and overall poor performance. Thus, the team shopped around and found another microprocessor, the NVIDIA Jetson Nano, which has a GPU and enough memory to run image recognition programs. The team was able to transfer the libraries and programs onto the Jetson Nano. As the team continued to develop and work with the Jetson Nano, the risk of an inadequate processor did not manifest.

Another risk the team may have encountered is the overfitting of our machine learning model. One of the risks of training a machine learning model is developing a model that accurately recognizes the data used to train it but is very inaccurate to new data [16]. Another way of putting it is that machine learning models want to track a signal or pattern amongst the hoard of data inputted but ignore any noise that is irrelevant to recognizing a pattern of the signal. If the model doesn't ignore irrelevant noise and includes it as a requirement to the pattern, it is overfit to its training data. In general, overfitting a model means it sees a pattern where there is none; it doesn't generalize well from training data to unseen data and has developed

specific criteria irrelevant to recognizing the pattern. Research done by the team has revealed that overfitting is a common pitfall for beginners in machine learning and is also relevant amongst experienced developers. Overfitting is a potential risk in the system as the model being trained may be overfit to recognize cars from only certain angles. Potential mitigation solutions would be to place the system at a vantage point in which the overfitting to specific angles is not a problem. Another overfitting risk is that the pictures of cars the team will use to train the model will cause the model to only recognize the cars shown in the training data but not any new car pictures. To mitigate this, models will need to be saved in different states of their training. By doing so, the model can be rolled back to a state in which it was not overfit. Another strategy to avoid overfitting is to use datasets available online that are widely used to train image classifier models. This strategy also offers a streamlined development path by cutting out the process of procuring and labeling a vast array of images.

Speaking of saving different states of the project, the loss of the original code, erroneously or not, is a risk with a potentially huge impact. Code being lost with no way of recovery would be devastating to the development of an image recognition feature. In fact, it would probably kill the project outright. The probability of this happening can be kept to a minimum if there are copies and backups of the code. To mitigate the risk of a team member deleting or corrupting code, it would be best for each team member to have their own personal copy to safeguard against the potential malicious intent of team members.

As mentioned in the introduction, unforeseeable risks perhaps pose the biggest threat to the development of the project's critical path. Since the team has beginner level experience with machine learning and image recognition, there may be many risks the team may not foresee but may become apparent as the feature is being developed. Thus, in order to mitigate these unforeseen risks, continuing research on machine learning and image recognition will be helpful, but the best way to mitigate unforeseen risks and risks in general is to get ahead of development schedule in a practical manner to allow for more time to solve or seek help for issues that may occur down the road. In regards to seeking

counsel, Professor Parham Phoulady was recommended to us after reaching out to the computer science department here at Sacramento State University. Of course, Professor Phoulady's assistance should not be relied on, but it is important to have the option to reach out to someone if the project runs into an issue that cannot be resolved by the limited knowledge of the team. However, once again, reaching out and seeking counsel is only an effective mitigation tool if it is done early as there may not be a simple and quick fix to an issue.

		Impact				
		1 - Very Low	2 - Low	3 - Medium	4 - High	5 - Very High
Probability	5 (81% - 100%)	Overshoot Project Design				Lack of communication
	4 (61% - 80%)			Programming errors	Time management	Mounting hardware failure
	3 (41% - 60%)		Microprocessor overheating	Microcontroller failure		Car recognition fail to detect consistently
	2 (21% - 40%)		Database error	Low Battery Capacity	Camera Failure	Model Overfitting
	1 (0% - 20%)	Display Malfunction		Fan failure		

Figure 11: Risk Assessment Heatmap [17]

B. Risk Mitigation for Non-critical Path

There are many risks that are not directly related to the critical path but are nonetheless a potential hinderance if not mitigated for. From the system's mounting hardware to the testing environment of the deployable prototype, there are many risks that pose a threat to the project's development.

A minor but potential risk to the project's development is if the GUI or database manifest a bug or an error in the middle of development. This risk can be properly mitigated through research on SQL and Python. As well, there are plenty of faculty members experienced with SQL and Python if all else fails.

Before we talk about the risks of the hardware itself, we need to mention the risk of obtaining incompatible hardware. Let's assess what can actually be lost and how can we mitigate it. Since the hardware at the laboratory prototype level is

non-critical, what we will lose is money and time. Hardware like the heatsinks cost about \$10 or less and purchasing an incompatible heatsink will only set our project estimated cost back by \$10. System hardware like Raspberry Pi camera or USB camera are more expensive but is will in the range of purchasing out of pocket if needed. These kinds of parts can mostly be found on Amazon which does provide fast shipping, so not much time is loss if we work on other aspects of the project while we wait. The risk of incompatible hardware can mostly be mitigated by working around the problem by either researching if we can purchase a compatible part or disregard the purchased part altogether and work around not having it.

Regarding the mounting hardware and encasing, there are several risks involved. Initially, the team did not know the dimensions nor weight of the deployable prototype. The goal of the project is to have a system that is portable and can be placed

inside a parking garage in order to collect data. Ideally, the design would use minimally invasive mounting strategies like suction or Velcro with adhesive to attach itself to a surface. The problem with this is that if the system turns out to be decently massive, the risk of the system being unable to stick to a surface would become highly probable. Originally, the mitigation plan to address this risk, is to research stronger mounting tools that are not invasive but may need to include invasive strategies in order to mount the system. This risk eventually became a non-issue as the team decided to have the enclosure be grounded for the most part while the camera would be the only part of the system suspended.

In light of mounting hardware, one of the biggest risks to the project is the possibility of the system falling off its mount and causing damage to the system or even pedestrians. The most important and expensive component currently used in the project is the Jetson Nano development board, which was about \$100.00. While not ideal, the cost of this risk would be minimal in this regard, and per the previously mentioned mitigation strategy on preventing loss of code, the software should be backed up somewhere that can be reuploaded to another Jetson Nano. Of course, there may be the off chance that the microprocessor may break before the team needs to demonstrate and another one cannot be retrieved in time, so having a backup would be ideal. However, the most dangerous risk would be if damages are done to someone as a result of the system falling. If this were to happen it could be potentially very costly as anyone who is victim to this could be compensated for special and general damages. Special damages could range from property damage as a result of the system falling on a vehicle, to damages which would result from medical services needed as a result of the system falling on a person causing injury. General damages would compensate the victim for unquantifiable damages such as pain and suffering [18]. Worst case scenario would be a wrongful death case. The team does not have the budget to compensate for a civil claim against us. The budget for the project is only about \$1,000.00, and about \$400.00 was already used. Ideally, to mitigate this risk, the system would be put in place that if it were to fall no damages would occur. However, the system will need a vantage point with a minimally

obstructed and wide view. Some risk will need to be accepted as camera angles would need to be in high up places. Thus, it is imperative that the mounting hardware be tested to stay up without any issues. The best the team can do is only setup the system in a lofty place if its fall would not cause damage to pedestrians.

Another risk of the project would be overheating, and heating damage caused to the system. Machine learning, and image recognition requires a lot of power to run. As a result, the system generates a lot of heat that needs to be dissipated. The Jetson Nano has heatsinked fins to help mitigate heat development, but if the process is running for a long time, that may not be enough. In fact, if it's just too hot outside, the system would be even more susceptible to overheating. If the system overheats a few things will happen. The image recognition process will lose precision and will provide less reliable results. As well, the Jetson Nano may end up breaking due to overheating. In order to mitigate these risks, including cooling components may be necessary. The system may even have to be limited to use during the temperate to cold seasons.

The procedures of testing the system in a suitable environment may also run the team up against harmful risks. One potential risk that would prove problematic would be the denial of testing in the Sacramento State parking garage. If the team is denied permission, it would make testing and developing the system very difficult as a similar environment would have to be found that would allow for testing. If this were to happen it would slow the development process down needlessly. In order to mitigate this risk, scouting the garages to figure out where the system will be placed and requesting permission early will be important in order to figure out the requesting process.

C. Final Thoughts on Risks and Mitigations

When writing this risk assessment section, it was important to focus on what is probable, but also keep an open mind to what is possible and what event would be the most damaging were it to happen. Developing a portable system that uses machine learning to extract data from video footage is riddled with risks both foreseeable and unforeseeable for students with little to no

experience. Figuring out the risks associated with the critical path is the most important risk assessment as a halt in the critical path halts the entire project. The project's most critical paths revolve around the development of the machine learning model to perform image recognition. If the image recognition software or hardware were to fail, then the whole project would fail. One potential risk the team has already encountered is the lack of processing power and/or memory, so we purchased a new stronger processor, the NVIDIA Jetson Nano, to mitigate this issue. Another risk the team may come across is the system overfitting; meaning that during the training process it may recognize patterns and adapt them even if they are just noise that isn't necessary to our project. We will need to make sure the program only learns what is required. Another huge risk is the loss of our code altogether, if that happens then absolutely all progress made would be lost. In order to mitigate this risk, we will create multiple copies of the source code and save it in distinct locations. The most critical path of our project is that the whole team has minimal experience with machine learning meaning this whole project we are learning as we work. We have to stay one step ahead of the program in multiple ways. Some of the non-critical risks include compatible heatsinks for our system so that our processor can continue working at its full potential and not have negative effects on the results. We will have to make sure we can create mounting hardware and encasings that will not have detrimental effects on our system. So, we will need to make sure we can easily move our system from place to place and easily accessible for maintenance. The mounting hardware will have to be sturdy enough to hold the system in elevated places and not fall onto vehicles causing property damage or causing damage to the place of installation. The enclosure will have to be able to protect the system from weather and other external factors; these may include rain, heat, or wind, as well as, passing pedestrians who may be inclined to make contact with the system. We may also encounter an issue with the availability of testing locations; our system seeks to collect data on Sacramento State parking structures but if for some reason permission were to be denied then we would have to rescope and find new testing grounds. In this report we discussed what we consider to be critical risks and what we consider to be non-critical

risks, as well as how we will deal with these risks. There are other risks such as monitor failure or low battery capacity but those can be dealt with extremely easily.

VIII. DESIGN PHILOSOPHY

The societal problem the team wanted to address was the "parking problem" at Sacramento State. The team wanted to offer a solution to thousands of students that suffer from the headache that is parking at Sacramento State during peak hours. It is an issue we, as students, have also experienced. During peak hours it can take up to 40 minutes to find a parking spot because driver's all flock to the same location in hopes of finding an open spot. So, we decided that our project would gather data to help professionals create a solution for the "parking problem."

A. Data Collection

One of the major issues created by the "parking problem" is traffic jams. Sacramento State University only has two entrances that lead onto the campus, and if all the driver's entering campus flock to the same parking area, well that greatly reduces the flow of traffic. We wanted to create a system that would lead drivers to open parking spots but that introduced too much human error and we have nearly no knowledge of traffic management. Therefore, we decided our system will collect good accurate parking data that professionals could use to create a solution. They have the knowledge of traffic management and the means of creating a solution, our system will give them the analytics they require to understand the problem. If professionals can find a way to help students make better choices when it comes to picking a parking area, then the flow of traffic will greatly increase.

The Sacramento State admission rate is ever-growing, meaning peak hours will also last longer. Our system will keep track of how many cars are entering and exiting a parking, as well as, how long certain parking spots are occupied throughout the day. This data can help predict the behavior of drivers and provide insight on which parking areas observe the largest influx of drivers looking for a

parking spot. With this knowledge a driver could make a more educated guess at where they could find an open parking spot. By eliminating the guessing situation, drivers will head to a location with purpose and this could improve the flow of traffic.

By keeping track of the classification of vehicles a professional can gain insight to the demographics of a parking area. Is there a need for bigger parking due to large trucks? Is there not enough parking available for motorcycles? Are students carpooling in larger vehicles? Is traffic affected by the busses and shuttles transporting students? By taking these factors into account then parking could become vehicle specific or offer better accommodations for certain vehicle types. Many times, when I've found an open parking spot, I would have to keep searching because my car didn't fit in that location; whether it be because the spot is for compact cars or a wide truck was spilling into the next spot.

This data is important for understanding the behaviors of the other drivers parking on campus. For understanding the demographics of the parking areas, what kind of vehicle is most likely to park there. With this information, professionals can create a solution for the "parking problem" at Sacramento State.

B. Cost-Effective and Accessible

Current smart parking systems require a large number of expensive components. The sensors utilized by these systems are embedded into the ground which makes maintenance a costly and time-consuming process. We wanted our system to be cost-effective and easily maintainable. The most expensive single component on our system is the development board processing the video feed. The system uses a fairly inexpensive camera that is connected via USB and allows for the user to mount the camera through any method they wish. If they so desired, they could even purchase a higher resolution camera. Maintenance is also as simple as unscrewing the enclosure and disassembling the parts as necessary. The system also has a rechargeable power bank which allows the user to operate in remote locations where there is no power source. The system is easily portable and gives the user a

wide range of mobility, they aren't confined to only areas that have power sources.

We wanted to make our system as accessible to professionals as possible. Through our research and consultation with Professor Khan we discovered the products already on the market are extremely expensive. And, the user is not given direct access to the data they collected. They have to rely on the company to process the footage for them, typically offsite. Our system will offer transparency so that the user can see all the data they gathered. This way they will gain better insight to the parking situation. There is information that only a professional can truly understand. The system is easy to use only requiring the user to enter a few lines of code to get the system up and running. The GUI makes the data collected extremely accessible since you can either view all the data collected or set filters to only view what is desired.

IX. DEPLOYABLE PROTOTYPE STATUS

As of April 20, 2020, the team has completed the deployable prototype and fully met the intended feature set. Thanks to the efforts of the team's hard work, a completed deployable prototype was finally achieved. We will now discuss each feature and our results.

A. Identification

The system is able to identify motor vehicles and categorize motor vehicles at a parking area in Sacramento State following the definition default naming scheme of the COCO dataset for vehicle classification. In order to meet this feature, the system measures the accuracy of identifying motor vehicles entering and exiting the structure. Through testing, we will have an accuracy of 99% making 1 out of 100 cars misidentified. We will categorize vehicles at the entrances and parking spots as either a bus, a truck, a car, or a motorcycle with a confidence level above 85%.

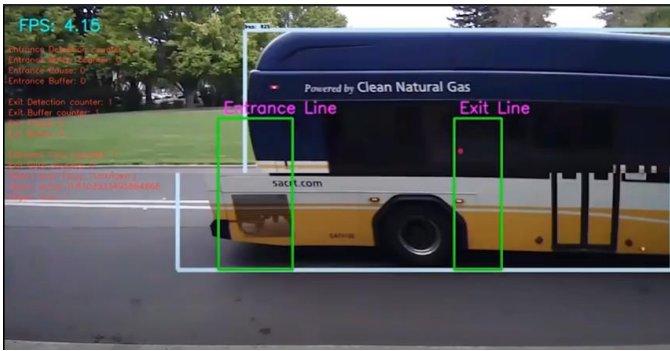


Figure 12: Bus identified [7]

Our midterm progress video demonstrated the system accurately categorizing each type of vehicle with the only mistake being one truck in the background being identified as a car. But, Michael fixed this issue and we were able to show a follow up demo with a correctly categorized truck. One misidentified vehicle is acceptable within our feature set. The system has a threshold of 85% confidence that won't allow it to identify a vehicle if under that level. The system was able to accurately identify vehicles it considered about the confidence level.

B. Occupancy

The next feature is the ability to distinguish occupancy of a single standard car space at a parking space in Sacramento State following the Zoning Code Parking Regulation [19] for the city of Sacramento. The tasks we set in order to meet this feature are as follows: we will measure the accuracy of distinguishing if a car occupies a single standard parking space of 8.5 by 18 ft within one aisle of 4 car parking spots. We will ensure each occupied parking spot with a confidence level of 94%. We will keep track of the start time and end time duration (in seconds) of a parked vehicle for each of the 4 observed parking spots.

Our demo video demonstrates these tasks being accomplished. We took two vehicles and would park them in different spots then relocate to other parking spots and the system registered the occupancy perfectly for each occasion. It kept track of the time the spot became occupied and the time the spot became unoccupied. The parking spot occupancy was also proven using a motorcycle and a truck. We weren't able to perform this test with a bus since no one in the team owned a bus.

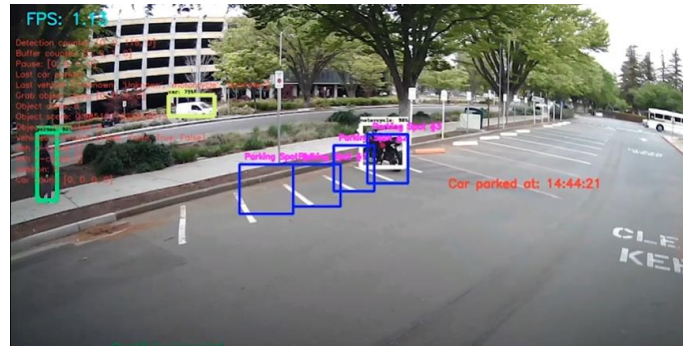


Figure 13: Parked motorcycle [7]

C. Car Counting

Another feature is the counting of each identified motor vehicle that comes in-and-out of the entrances and exits of the parking area. This feature requires the system keep an integer count of vehicles that have entered and exited the parking structure. The category of the vehicle will be recorded with a confidence level of 85% (as mentioned previously).

Our demo video again demonstrates this feature being accomplished. Michael set thresholds in the video feed that acted as the entrance and exit to a parking area. The system detects and identifies a vehicle and a little red dot follows the center of the vehicle, when the red dot crosses either threshold then the system will count it as entering or exiting respectively. The entrance and exit thresholds can be either set up as vertical lines or horizontal lines.

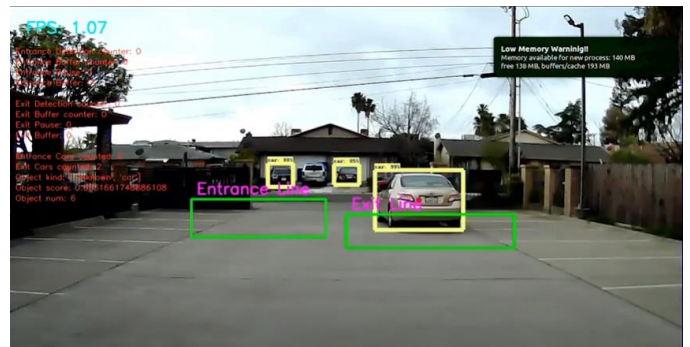


Figure 14: Horizontal thresholds [7]

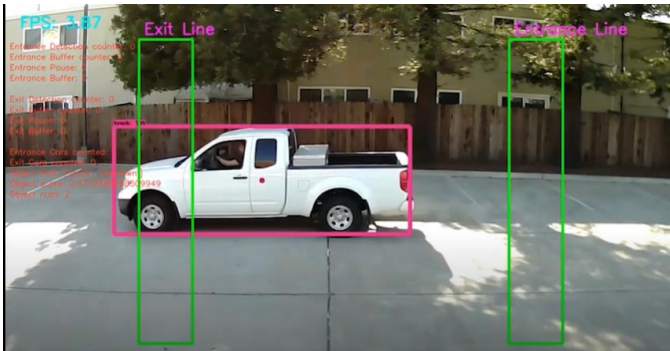


Figure 15: Vertical thresholds [7]

D. Power

The next feature revolves around portability with the use of battery power. It requires that the power bank last at least 10 hours-worth of battery life. This allows for system to be able to be used on remote locations and collect enough data for analyzation.

The tests the team ran proved that the system can provide reliable power to the system for at least 10 hours. Operating the system at 5 watts, the power bank has a battery life of about 14 hours.

E. Database and GUI

The last feature our system met involves a database that will record the data and the GUI that displays the data collected. The database will have a record of the integer count of cars being counted, the car classifications, and the statistics of a specific car occupying a spot. And, by using the GUI, we can measure how long a car has occupied a space within a certain period of time by the seconds. We can also filter data entries by the different data collection sessions.

1	2	3	4	5	6	7	8
1	out	09:40:22	car	1	1		
2	out	09:43:10	car	2	1		
3	in	09:47:31	car	1	1		
4	in	09:48:34	car	2	1		
5	in	12:33:18	bus	1	2		
6	out	12:33:27	bus	1	2		
7	in	12:37:26	truck	1	3		
8	out	12:37:34	truck	1	3		

Figure 16: Raw data [11]

The GUI allows access to the data collected in a format that makes sense. The GUI also allows for the filtering of the data so that the user can access specific data. The GUI allows for filtering by hour range, vehicle type, parking spot, and data collecting session.

EntryID	Spot	Entry Time	Exit Time	Vet
3	3	7:01:00	7:16:06	se
5	2	7:16:50	7:18:08	se
8	1	7:01:50	7:22:02	se
10	4	7:23:00	7:26:30	se
12	2	8:27:00	8:38:40	se
15	1	7:55:00	12:00:30	se
17	1	7:10:43	7:35:00	se
20	4	7:04:04	10:03:05	se
21	1	7:45:08	8:03:02	se
24	2	6:05:30	6:34:00	se

Figure 17: GUI and data [11]

X. MARKETABILITY FORECAST

Determining the deployable prototype's marketability is a matter of examining the perceived market audience, comparable services and products, and perceived market value. Our perceived market audience would be made up of organizations that can use these statistics to provide transportation construction projects and studies to better their local transportation infrastructure. This would include the individual involved in academia who wish to develop studies on parking behavior. Using machine vision to monitor parking spots, and vehicular traffic in general, is not a new practice. In fact, there are many companies that provide traffic monitoring services far more advanced and polished than what our own system is capable of. Companies like Miovision, TrafficVision, and Pelco have developed their own cameras, machine vision software, database infrastructure, and data forwarding platform neatly packaged for the market. It should also be noted that machine vision is not the only way to monitor parking spots. Companies like Indect use cameras for machine vision as well as ultrasonic sensors to monitor parking spaces. Compared to our design, these professionally developed products are far more sophisticated and polished. However, one of the features that makes our system stand out is the fact that the video processing is done on site and is in the hands of the user, unlike other companies who

process the video on their end and forward the aggregated data to the user. Determining the value of our system is also a bit ambiguous since similar services' prices are a case by case determination. There's the cost of the hardware, like the cameras being used, but the price of the video processing is variable. Since how much traffic being monitored scales based off how many areas are being monitored, the prices can vary widely. Even the type of user determines the price tag of utilizing these professionally made systems.

A. *Perceived Market Audience*

Cameras that monitor traffic on roads and freeways are often maintained by state departments of transportation. Along with monitoring the roads for accidents or major closures, footage from traffic cameras is influential in decisions regarding future road development and construction. The same can be said about monitoring parking spaces; data collected on the statistics of parking spaces can pave the way for improving roadways or parking spaces.



Figure 18: The California Department of Transportation Logo [20]

The California Department of Transportation (Caltrans) is a perceivable market audience that can benefit by the use of our system. Caltrans services the California's public transportation welfare. According to their program for Traffic Operations, Caltrans aims to provide integrated and efficient transportation systems. These systems are in the forms of Integrated Corridor Management (ICM), Intelligent Transportation Systems (ITS), and Connected Corridors. This includes how freeways,

arterials, transit, and parking systems work together by use of well-integrated communication networks into transportation infrastructure. Caltrans is currently testing the Connected Corridors project in the I-210 of the Los Angeles Area [20]. Within the corridor project, Caltrans must rely on good monitoring systems to adequately cover needed areas of freeways and major arterials. The pilot project listed the following as critical data to be characterized: traffic conditions of freeways and arterials, parking availability, traffic monitoring systems, information devices, traffic control devices, and ICM system status. Caltrans states that the ICM program should incorporate parking information whenever possible. They intend to provide real-time parking information to encourage the use of parking facilities to use of public transit and reduce the number of cars on the road. Very few facilities have systems that count the occupancy of vehicles which the project intends to integrate more in the future. A system like ours can potentially aid in collecting data needed to integrate counting systems that can benefit facilities that need them the most.

Another perceived market audience that can benefit in our system are students and professors in educational environments. Civil Engineering students and professors need relevant useful data to teach transportation engineering concepts. In order for professors to obtain such data to be taught to the students, one must purchase it from contractors or obtain said data themselves. This can be an expensive venture for purchasing data and time-consuming task for non-experts in the field of machine learning. For Civil Engineering students and professors wanting to collect data themselves would find it difficult to create such a device to collect data if they do not have any programming or machine learning background. This task would be beyond their field of study and is best done by experts that would normally be outsourced. Depending on the volume of the project, contractors that can collect data can charge up to \$1800 for complex classification. For statewide projects, contractors can charge \$120 per traffic count [21]. This does not include the needed equipment to perform the data collection. Our system would benefit with this audience as the cost would go only to the equipment and software. The user would have

full control of the data which makes it marketable to students and professors in large volumes.

B. Comparable Services and Products

As stated before, monitoring parking spaces, and vehicular traffic in general, as a service has been around for a while. In fact, there are entire companies that specialize in providing these services to municipalities all over the world. Miovision, TrafficVision, Pelco, and Indect are just a few companies to provide these services. These well-established companies would absolutely quash our team's design, but our design has a few features that stand out amongst these companies' products.

Miovision is a traffic solution company that uses artificial intelligence to help cities actively monitor traffic to become smart cities. Their products track pedestrians, cars, and bicyclists. They can determine the travel time, road volume, intersection count, and a whole slew of statistics based off the user's request. They offer two methods of monitoring vehicles. They can apply permanent fixtures with their Miovision TrafficLink or their portable device, Miovision Scout, that can be used in remote areas. Their Miovision TrafficLink service includes a SmartView 360 Camera and SmartSense Hardware to interface with the camera and forward the captured video to Miovision's data center to process the footage. These devices need to be plugged into a main power source, so their implementation relies on the infrastructure it is placed in. The Miovision Scout is a portable device that can be used in remote locations that do not have the infrastructure to support the Miovision TrafficLink hardware. The Scout hardware weighs approximately 80lbs, has a battery life of 72 hours, can store 355 hours of video, has LTE and Wi-Fi capabilities to forward footage and monitor or change studies, and a 5.5" backlit LCD display. It can also operate in temperatures as low as -40° F and as high as 140° F. Some of Miovision's accomplishments includes helping Detroit, MI to modernize their dated infrastructure to monitor and analyze traffic performance, provide traffic data to 3 Chicago, IL to help their city become more bikefriendly, and optimized signal timing in the city of Maricopa, AZ to help reduce traffic congestion [22].

TrafficVision implements their machine vision software onto existing camera infrastructure to monitor and collect data on vehicular traffic. If existing cameras use a standard digital encoding, the video feed can be analyzed over their network. TrafficVision software can operate on numerous hardware platforms such as Cloud, commercial off-the-shelf servers or robust equipment in the field, or virtual machines. TrafficVision uses a browserbased application which allows anyone with designated network access to configure analytics from anywhere. Some of TrafficVision's accomplishments includes helping the Colorado Department of Transportation reduce time for incident detection, helped Kansas City's electronic active traffic management systems to increase incident detection accuracy and decrease incident response time, and helped the Ministry for Transportation Ontario assess traffic flow and plan for traffic management during the 2015 PanAmerican games [23].

Pelco is a company that leads the industry in video surveillance and security system products and technologies. More specifically, Pelco specializes in the development of high-end cameras. Thus, it is no surprise Pelco has also expanded into providing services such as monitoring vehicular traffic using their high-end cameras. With Pelco's intersection, roadway, bridge/tunnel management solutions, the company can provide traffic incident alerts, analytics about what is causing traffic congestion, and data for engineering studies such as car and pedestrian count. Some of Pelco's notable accomplishments includes the update of South Korea's Cheonan-Nonsan Expressway traffic camera monitoring system, enhancement to Fresno Police Department's surveillance cameras and video management system, and upgrade of Istanbul Metropolitan Municipality Traffic Control traffic cameras and video management system [24].

While the previous companies' services deal with vehicular traffic data collection in general, Indect is a company that focuses solely on studying parking. As Indect specializes in monitoring parking, it allows them to utilize not only cameras to monitor parking spots but also ultrasonic sensors to monitor individual spots. Their camera-based sensor, UPSOLUT, is placed on the ceiling of a parking garage in the middle of the parking rows and can monitor 6 parking spaces at a time in a parking

garage. UPSOLUT can monitor parking duration, turnover, number of entries to spaces in a period, traffic flow, and even search license plate number. The camera is also capable of being immersed in water without being damaged. Indect also develops an ultrasonic mini sensor, UMS, to monitor individual parking spots in a parking structure. The sensor is completely independent with no need of central control. Indect is not just limited to indoor parking structures. Indect's ODE Outdoor Camera Detection System is designed to detect parking cars in outdoor parking lots. The system uses a pan-tilt-zoom or fisheye camera, is capable of detecting 20+ spaces with one camera depending on positioning, and has a controller with artificial intelligence computing and controls up to 8 cameras. The camera or cameras forwards the video via Wi-Fi to Indect where detection algorithms evaluate the space status. The ODE Outdoor Detection System provides 98% or better accuracy, given there is a clear view of all the spaces and heavy rain or other extreme weather conditions are not present. Indect's resume of accomplishments is vast and impressive. Indect has provided parking monitoring systems to major airports like John Wayne Airport, upscale hotels owned by MGM, universities like Texas A&M, garages in notable cities like Beverly Hills, and even famous malls like the Dubai Mall [25].

With all these accomplished companies in the market, our team's design clearly falls short in

many ways. However, our system stands out in a few ways. For instance, save for Miovision's Scout, most monitoring systems lack portability, limiting their systems to mostly permanent fixtures. This portability allows for quick parking monitoring endeavors, with little preparation, in a wide array of locations. Companies are quick to offer the hardware, but the software that processes data is proprietary. The main source of revenue for these companies are in the processing of the video feed which isn't typically done on-site in the hardware. The data is forwarded to their network and is processed on their end. Our system puts the hardware and software in the consumers hands. Because of this, those in academia who wish to control all aspects of the system will be able to and can adjust the system as they please.

C. Market Value

The traffic management market is an ever-growing market. The market will increase and become more competitive along with advancements in technology. There are major vendors from all over the globe, but North America is expected to have the second-largest market by 2024. As of 2019 the traffic management market is worth USD 30.6 billion and is expected to be worth USD 57.9 billion by 2024 [26].

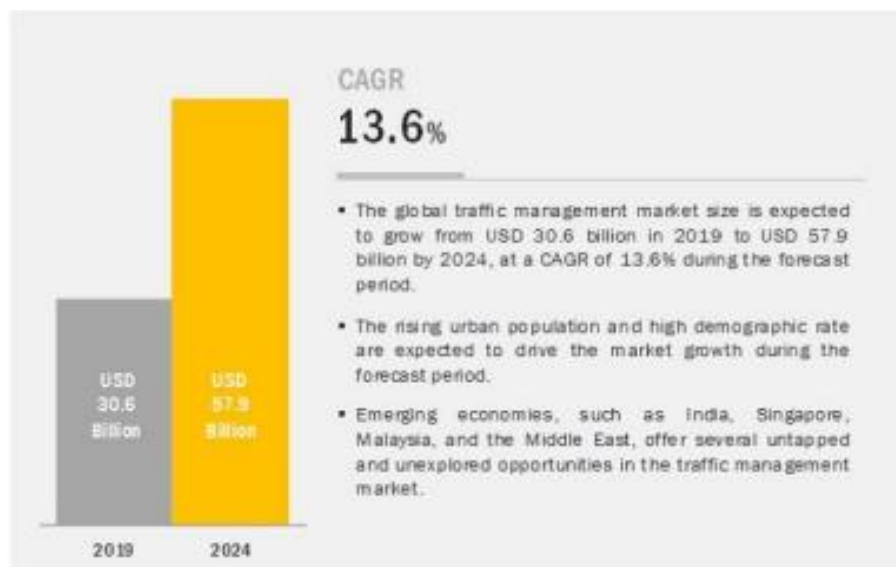


Figure 19: Attractive opportunities in the Traffic Management Market [26]

Unfortunately, finding information on the price of the services and products available is a bit difficult. The cost of the services provided are determined by a case by case basis. This is because companies typically charge for the analysis of the video as an additional price separate from the hardware they may provide. This price is determined by how long the video being analyzed is. This rate isn't advertised readily either, as one has to receive a quote from the company. Thus, purchasing these products and services is like buying car insurance, but it's more ambiguous because these services are typically only used by municipalities not the general public. After reaching out to Miovision, TrafficVision, Pelco, and Indect, only Miovision gave us information on their Miovision Scout, which is very similar to our team's design. We were informed by an email from a Miovision employee that the Scout unit costs \$5,000.00 just for the hardware itself. No information was given on the rate of the video processing price as it varies based off the unit's use [22].

D. Refining Needed

By investigating comparable services and products, looking into the market audience they supply, and observing the market value they pursue, the team learned just how lacking our system was. For our project to be able to compete in the market, there are still a ton upgrades necessary. To put it bluntly, our project is still very much a rudimentary design created by three college students. It goes without saying, too, that the team also lacks what companies in the market have plenty of, funding. Aside from that, even if overhead costs were magically waved, the deployable prototype needs many improvements to reach a state in which it can be manufactured and put on the market.

To begin, our system may need an updated microprocessor. Although this is one of the best development boards, the NVIDIA Jetson Nano is relatively old in terms of architecture. The board was released on March 18, 2019, but the Maxwell architecture is 5 years older! [27] This can't be controlled, but if we had a development board with NVIDIA's Turing architecture, we can utilize special instruction sets to help accelerate the

machine vision process. The Turing architecture has special hardware and instruction sets that is specialized in machine vision applications. [28] As of writing this document, there is no development board of that capability ready to buy for consumers. This may be the Jetson Nano 2. But for now, we are stuck with the current Jetson Nano and its limitations. The 4 GB LPDDR4 ram is plenty for our application, but as you increase the number of objects to detect, the 4 GB ram becomes a limiting factor for FPS performance. In our testing, the Jetson Nano continues to utilize the swap space since the detection model uses most of ram. Theoretically, if we had the same Jetson Nano with at least 8 GB of ram and rerun our controlled tests, we won't run into any memory usage issues which will decrease FPS. FPS doesn't become a concern for our application, but our application can have large workloads, therefore increasing the boards memory size with a newer architecture may yield better and more accurate data when doing data collection with machine vision.

When designing the enclosure, we were limited to the platform size of the 3D printer. Due to these limitations the components are fit very tightly inside the enclosure. One of the walls printed a little warped so it left a gap in the system that we covered using silicone. The system needs a sturdier enclosure that offers a better fit and better protection. The system could also use a battery that would allow it to be operated for a few days at a time.

The system would need a better way to integrate all the features and components together. Although the enclosure does a real good job of keeping the processor and all electrical components bundled nicely, the camera is simply connected via USB to the front. If the deployable prototype were to be released to manufacturing, a camera would need to be developed in house. The data transfer and storage via a GUI interfacing with a database would also need work done. As of now, the database used is not of the team's design. A locally stored MySQL database is used which was a free community version download. In addition, while the GUI does filter collected data based off user selected parameters like time or type of vehicle, it provides no analytics of the actual data. The recognition model can also only classify vehicles into categories

specified by the COCO dataset. In other words, it can only classify vehicles into a car, bus, truck, or motorcycle. And perhaps the biggest flaw of the system, it can only reliably keep track of four parking spots at a time. This greatly limits the parking occupancy data that can be collected per session.

XI. CONCLUSION

The frustrating and time-consuming task of finding a place to park is an issue that all drivers have experienced on numerous occasions. This is especially true for those who commute to Sacramento State regularly. After researching the issue and forming solutions on our own, our naïve understanding of the issue and its nuances became more apparent. Thanks to the counsel of Professor Tatro and Professor Khan, we learned that the complexity of the issue requires a solution that will most likely emerge from professionals who are dedicated to traffic management in general. However, this potential solution is something that a team of 3 college students who study computer engineering and electrical engineering can help the professionals achieve. Solving an issue, intentionally, requires information on the issue. Therefore, the collection of information will then help professionals move closer to a more tenable solution to the parking problem. Thus, the project we set out to create was a system that would collect parking data that can be used by professionals to solve the parking issue on Sacramento State and even other establishments with parking issues.

With the team's new perspective and a concrete goal to work towards, the team set out to come up with the best and most pragmatic design the team could achieve. The team started with researching the traffic data collection methods that were commonly used. One of the most widely used methods was the utilization of inductor loops. This method was fairly reliable in terms of collecting data on when vehicles approached it. However, the main drawback of this method was that it was extremely costly, in respect to the team's budget, and its invasive installation was beyond the means of the team's capabilities. Not to mention the permission to effectively build and use this on campus was out of the question for a group of

undergraduate engineers. In addition, our goal was to collect as much information as possible; the inductor loops may be good at counting cars or indicating when a car has approached it, but it cannot give any information on the type of car. Plus, in order to monitor parking spots, the inductor loops would need to be installed under every parking spot we wanted to monitor and collect data on. Doing so would be both costly and impractical for the team. Therefore, our design idea was shaped by the requirements that it be cost-effective, easy to install, portable, and collect more information on the vehicles, such as its type.

In order to fulfill these requirements, the team decided to create a system that would use machine vision on a microprocessor to collect data on parking areas. Initially, the team started developing the system with a Raspberry Pi board and Raspberry Pi Camera V2. However, these parts were insufficient to run detection models as the computational power of the Raspberry Pi board was not fast enough to keep up with the needs of an image recognition model. The team decided to use the NVIDIA Jetson Nano development board as it had a much better GPU and could meet with the demands of a program that used a recognition model. In regard to the camera, the team switched to a USB camera that had a higher quality sensor and a larger viewing angle. These changes in parts would better allow the design to identify motor vehicles and categorize them as either a car, bus, truck, or motorcycle. Thus, it allowed the system to run a program to monitor and collect data on vehicles occupying parking spaces. It also allowed the system to run a program to count and collect data on vehicles that entered a parking area. With more computational power comes more heat, therefore our team, through some trial and error, decided to install a Noctua fan to cool the system and allow it to collect data unencumbered by the heat it would generate. Since the design idea was focused on data collection, we needed to find a way to store and organize the data that can be readily available to professionals who are working towards finding a solution to the parking problem. Therefore, a GUI was developed in Python to

interface with a MySQL database. The GUI allowed the user to transfer and upload the data collected by the system and can be filtered to find specific types of data. In order to provide the system with portability, a power bank was created to allow the system to run without a nearby power outlet, and an enclosure was made with 3D printed parts. The enclosure was capable of housing the power bank, Jetson Nano development board, and attached cooling fan.

The budgetary constraints have been mentioned many times in this paper, because anything beyond \$1,000.00 was basically out of the question. Most of the budget was used in the first semester as it was the period in which most of system's parts would be purchased. Some parts such as the Raspberry Pi 3 and Raspberry Pi Camera V2 were not used for the deployable prototype build. However, their share of the budget was not for nothing as each and every purchase helped provide a path to the final deployable prototype. The acquisition of parts and other development materials came out to \$536.00, which was \$464.00 under budget.

The milestones mentioned in this paper, could be used as an adequate indicator of the team's progression from start to finish. As mentioned, the development of a concise societal problem was imperative to the inception of our team's deployable prototype. Once the team agreed to collect data on parking statistics to be used by professionals, the next step was deciding on a design idea. After consulting with our technical advisors, we decided to use a classification model running on a microprocessor, a camera, and a power bank to collect data on parking statistics and store the information onto a database. With a design idea in place, the team set out on completing the laboratory prototype. Michael and Ryan were primarily assigned to train and integrate a detection model that could classify toy-sized vehicles. With the additional help from Sergio to train the detection model, the team was able to accomplish the training and integration of the detection model onto the Jetson Nano developer board. After this milestone was accomplished, Michael and Ryan were assigned to work on the parking spot monitoring

program and car counting program. After failing to meet the standards of a laboratory prototype, the team worked on until the next semester. These programs were later completed, at the laboratory prototype scale at the beginning of the next semester. The power bank was completed by Sergio and was able to successfully supply power to the system. Changes were later made to the power bank in the second semester in order to power the system more reliably. The next milestone completed was the development of the GUI by Ryan that could upload data from a flash drive to a MySQL database. The laboratory prototype version was also presented, which failed to pass the initial technical review. Work continued on the GUI and database until the beginning of the next semester in order to meet the standards of technical review. After the team demonstrated a laboratory prototype that met the standards of the technical review, the team revisited the societal problem. The next milestone was accomplished by Michael, which was integrating a recognition model trained with a COCO dataset to use our system on a full-scale vehicle. With this done Michael and Ryan could now run tests on the spot occupier and car counting programs. Modifications were made by Michael to include 4 spots that could be monitored at once. Ryan helped modify the logic of the program to include what type of data would be written to the text file. Thanks to the entire team's efforts, the programs were tested to show that the programs were able to collect data on parked vehicles and count vehicles entering and exiting a parking area. The next milestone was the completion of the deployable prototype version of the GUI which was upgraded to include the filtering of data entries stored on the database. This met the feature requirement of the GUI and database, since it allowed for the data to be filtered by session amongst other non-required filtering capabilities like time range and vehicle type. The next milestone was the completion of the enclosure made from 3D printing which was worked on by Sergio and Michael. It housed the Jetson Nano, fan, power bank, and displayed the voltage of the battery. Finally, the last milestone was the presentation of

the deployable prototype. In this presentation we were able to demonstrate a functioning deployable prototype that could collect data on vehicles in parking spots and count vehicles entering or exiting a parking area.

The development path for the project needed to be treaded carefully in order to avoid risks that could potentially sink the entire project. This is especially true when it came to the implementation and integration of the recognition model. Risks were organized into those that would affect the critical development path of our design, and risks that the team identified that would not. Some of the risks, as they relate to the critical path of the deployable prototype's design, were issues like overfitting our recognition model. The team was able to avoid the risk of overfitting the model by using a recognition model trained with the common objects in contrast, or COCO, dataset. Other risks involved the failure of the recognition model to properly recognize vehicles according to the COCO dataset. Mitigation plans included changing the camera, adjusting the framerate in which the program takes an inference of a video, looking into other recognition models, and as a last resort contacting a specialist. Risks that did not relate to the critical path included the possibility that the enclosed system may fall and cause damage to the system or bystanders. This risk could be mitigated by simply keeping the enclosure near the ground as much as possible. If placement of the device needs to be substantially above ground, the system should be backed up to another Jetson Nano Developer board. However, the system should entirely avoid being suspended in a place that could cause harm to others, no matter what. If anything, the responsibility of damage caused to others would lie on the team, and a potential lawsuit would be devastating to the team's budget and beyond.

In order to get a better idea of the deployable prototype's marketability, the team needed to do some research on the potential market audience, the market value, and the design's biggest competitors. Typically, the design would be geared towards professionals who need to collect data on parking behaviors in order to create more efficient roadways and better parking areas. These types of professionals are typically those who work for transportation municipality departments, such as

Caltrans, in charge of developing and maintaining vehicular roadways. In terms of market potential, there is a ton of room for growth. In North America especially, the market value in traffic management is expected to grow to USD 57.9 billion [26]. With such a highly valued market, it has fostered the development of plenty of products and services by very large and established companies. In fact, companies like Miovision have developed a product very similar to our deployable prototype, the Miovision Scout.

In order for the deployable prototype to compete and become ready for manufacturing, many upgrades need to be implemented. For starters, the deployable prototype will need to update its microprocessor. Ideally, a board geared specially for machine vision programs would a huge improvement to the performance of the deployable prototype. NVIDIA, the same company that developed the Jetson Nano development board, has developed a new architecture called the Turing architecture which has specialized hardware designed to accelerate the machine vision process. If they implemented this new architecture to a new development board, it would be a much better alternative to the current microprocessor the deployable prototype is using. The database and GUI would also need work done since the database used is a locally stored community version MySQL database and the GUI does not provide any analytics beyond displaying filtered data. The recognition model would also need to be retrained as it can only classify vehicles into categories specified by the COCO dataset. Currently, it can only classify vehicles as a car, truck, bus, or motorcycle. If further distinctions are to be made, it would need to be retrained with new training data.

REFERENCES

- [1] *Fhwa.dot.gov*, 2019. [Online]. Available: http://www.fhwa.dot.gov/publications/research/operations/its/06108/images/fig1_4.gif. [Accessed: 23- Sep- 2019].
- [2] “Tegra Linux Driver”, *Docs.nvidia.com*, 2020. [Online]. Available: https://docs.nvidia.com/jetson/14t/index.html#page/Tegra%2520Linux%2520Driver%2520Package%2520Development%2520Guide%2Fpower_management_nano.html%23wwpID0E0MO0HA. [Accessed: 27- Apr- 2020].
- [3] “Rpi Camera Module - eLinux.org”, *Elinux.org*, 2020. [Online]. Available: https://elinux.org/Rpi_Camera_Module#Technical_Parameters_.28v.2_board.29. [Accessed: 27- Apr- 2020].
- [4] O. Knocklein, “Classification Using Neural Networks”, *towardsdatascience.com*, 2019. [Online]. Available: <https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f> [Accessed: 29- Feb- 2020].
- [5] A. Sachan, “Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN, YOLO, SSD”, *cv-tricks.com*. [Online]. Available: <https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/> [Accessed: 29- Feb- 2020].
- [6] *missinglink.ai*, “Convolutional Neural Network Architecture: Forging Pathways to the Future”. [Online]. Available: <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/> [Accessed: 29- Feb- 2020].
- [7] M. Khoo, “Full-Scale car program”, 2020
- [8] M. Khoo, “Prototype car counting program”, 2019
- [9] S. Battery, “Samsung 25R 18650 2500mAh 20A Battery INR18650 -25R - 18650 Battery Store”, *18650BatteryStore.com*, 2020. [Online]. Available: <https://www.18650batterystore.com/ProductDetails.asp?ProductCode=SAMSUNG-25R-18650>. [Accessed: 01 - Mar - 2020].
- [10] S. Cortes, “Power Bank Testing and Lux Testing”, 2020
- [11] R. Uda, “Database and GUI”, 2020
- [12] R. Uda, M. Khoo, and S. Cortes, “Expenses”, 2020
- [13] M. Khoo, R. Uda, and S. Cortes, “Laboratory Prototype”, 2019
- [14] S. Cortes, “Enclosure”, 2020
- [15] S. Cortes, M. Khoo, and R. Uda, “Timeline”, 2020
- [16] “Overfitting in Machine Learning: What It is and How to Prevent It”, *elitedatascience.com*, 2019. [Online]. Available: <https://elitedatascience.com/overfitting-in-machinelearning> [Accessed: 3- Nov- 2019].
- [17] M. Khoo, R. Uda, and S. Cortes, “Risk Assessment Heatmap”, 2019

- [18] “Special Damages in Breach of Contract”, legalmatch.com, 2019. [Online]. Available: <https://www.legalmatch.com/lawlibrary/article/special-damages-in-breach-ofcontract.html> [Accessed: 3-Nov-2019]
- [19] “Zoning Code Parking Regulations - City of Sacramento”, Cityofsacramento.org, 2019. [Online]. Available: <https://www.cityofsacramento.org/CommunityDevelopment/Planning/CurrentPlanning/Zoning/Zoning-Code-Parking-Regulations>. [Accessed: 07- Oct- 2019].
- [20] “Project Planning | Caltrans”, Dot.ca.gov, 2020. [Online]. Available: <https://dot.ca.gov/programs/trafficoperations/connected-corridors/project>. [Accessed: 24- Feb- 2020].
- [21] Safety.fhwa.dot.gov, 2020. [Online]. Available: <https://safety.fhwa.dot.gov/rsdp/downloads/fhwasa17034.pdf>, Pg. 8 [Accessed: 24- Feb- 2020].
- [22] “Miovision - Smart cities start here”, Miovision, 2020. [Online]. Available: <https://miovision.com/>. [Accessed: 17- Feb- 2020].
- [23] “TrafficVision”, TrafficVision, 2020. [Online]. Available: <http://www.trafficvision.com/>. [Accessed: 19- Feb- 2020].
- [24] “Pelco Security Cameras and Surveillance Systems”, Pelco.com, 2020. [Online]. Available: <https://www.pelco.com/>. [Accessed: 19- Feb- 2020].
- [25] “Parking Guidance Systems | Camera Based Parking Guidance | Parking Signage”, Parking Guidances Systems from INDECT, 2020. [Online]. Available: <https://indect.com/>. [Accessed: 19- Feb2020]
- [26] T. Market, “Traffic Management Market Size, Share and Global Market Forecast to 2024 | MarketsandMarkets”, Marketsandmarkets.com, 2020. [Online]. Available: <https://www.marketsandmarkets.com/MarketReports/traffic-management-market-1036.html>. [Accessed: 24- Feb- 2020].
- [27] N. Newsroom, "NVIDIA Announces Jetson Nano: \$99 Tiny, Yet Mighty NVIDIA CUDA-X AI Computer That Runs All AI Models", *NVIDIA Newsroom Newsroom*, 2020. [Online]. Available: <https://nvidianews.nvidia.com/news/nvidia-announces-jetson-nano-99-tiny-yet-mighty-nvidia-cuda-x-ai-computer-that-runs-all-ai-models>. [Accessed: 27- Apr- 2020].
- [28] *Nvidia.com*, 2020. [Online]. Available: <https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf>. [Accessed: 27- Apr- 2020].
- [29] S. Cortes, R. Uda, and M. Khoo, “System Hardware”, 2020
- [30] “Intel Developer Zone Articles”, *Strophically35.rssing.com*, 2020. [Online]. Available: http://strophically35.rssing.com/chan-18301187/all_p48.html. [Accessed: 26- Apr- 2020].
- [31] *Sparkfun.com*, 2020. [Online]. Available: <https://www.sparkfun.com/datasheets/Sensors/Imaging/TEMT6000.pdf>. [Accessed: 27- Apr- 2020].

GLOSSARY

Algorithm: A set of instructions or specifications for performing calculation, data processing, automated reasoning, and other tasks.

ARM: Advance RISC Machines

BMS: Battery Management System.

Camera Module: A camera module is an image sensor integrated with a lens, control electronics, and an interface like CSI, Ethernet or plain raw low-voltage differential signaling.

COCO: Common Objects in Context. A library of image datasets used to train classifier models to recognize certain objects.

CPU: Central Processing Unit.

FPS: Frames Per Second.

General Damages: Damages that compensate for non-quantifiable loss like pain and suffering.

GPU: Graphics Processing Unit.

GUI: Graphical User Interface.

ICM: Integrated Corridor Management.

ITS: Intelligent Transportation Systems.

Mounting Hardware: Physical hardware like hinges or stands, that is used to physically mount the device to a surface.

NVIDIA Jetson Nano: A microprocessor which has a GPU that is often used in machine vision systems.

ODE: Outdoor Detection System.

OpenCV: Open Source Computer Vision Library.

RCNN: Region Convolutional Neural Network.

RPi: Raspberry Pi.

Special Damages: Damages that compensate for quantifiable monetary loss like medical bills or damaged property.

SQL: Structured Query Language. The standard language used for relational database management systems.

SSD MobileNet v2: Single Shot Detector Mobile Net v2. A classifier best geared for mobile devices.

TensorFlow: a free and open-source software library for dataflow and differentiable programming across a range of tasks.

UMS: Ultrasonic Mini Sensor.

APPENDIX A. USER MANUAL

A. In the field operation

The device is packaged for operations in standalone use case environments, but at its current prototype state, the device cannot simply operate and perform its task without an initial setup. This means that the device requires input (keyboard/mouse) and a display monitor.

- 1) Place the camera tower in desired location and extend to a desired height and angle
 - a. For best results angle the camera aiming directly at the vehicles
- 2) Attach the camera USB to an available USB slot in the microprocessor

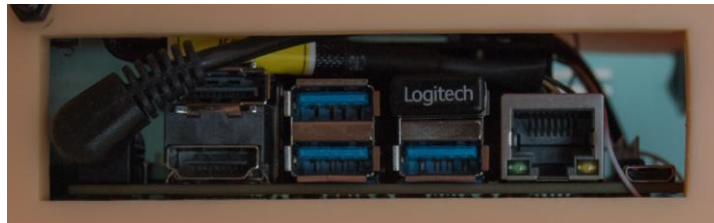


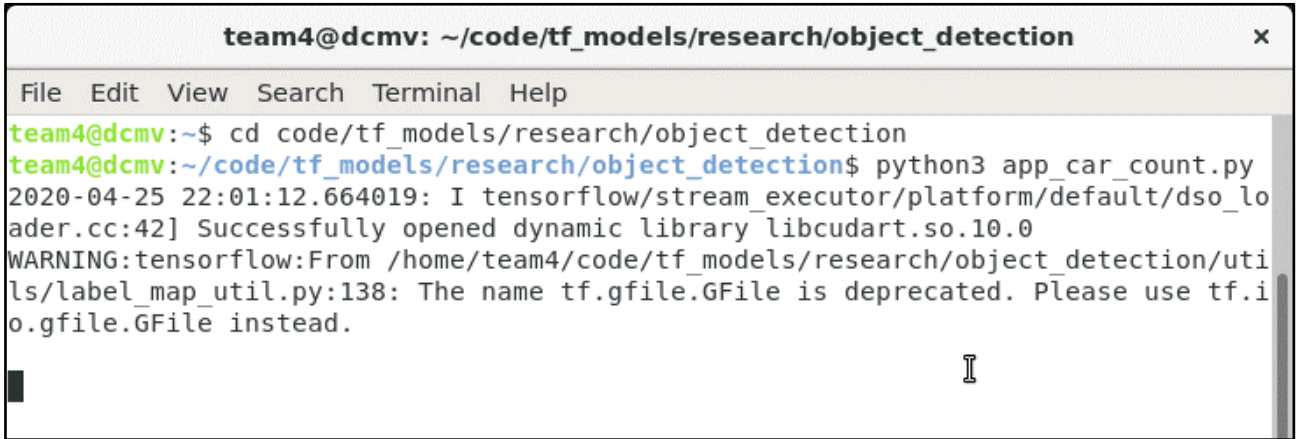
Figure A 1: Team4 Device I/O [14]

- 3) Attach the keyboard/mouse and display monitor to the microprocessor
- 4) Switch on the device power
 - a. The device includes two switches on the enclosure; the left switch activates the voltage regulator and the right switch activates the voltmeter.
- 5) Log into the machine and start command prompt
- 6) Change your directory to *code/tf_models/research/object_detection*

```
team4@dcmv: ~/code/tf_models/research/object_detection x
File Edit View Search Terminal Help
team4@dcmv:~$ cd code/tf_models/research/object_detection
team4@dcmv:~/code/tf_models/research/object_detection$ █
```

Figure A 2: Team4 Change Directory [7]

- 7) Start either the entrance/exit counting program by typing: `python3 app_car_count.py` or the car parking detection program: `python app_car_park.py`



```
team4@dcmv: ~/code/tf_models/research/object_detection
File Edit View Search Terminal Help
team4@dcmv:~$ cd code/tf_models/research/object_detection
team4@dcmv:~/code/tf_models/research/object_detection$ python3 app_car_count.py
2020-04-25 22:01:12.664019: I tensorflow/stream_executor/platform/default/dso_loader.cc:42] Successfully opened dynamic library libcudart.so.10.0
WARNING:tensorflow:From /home/team4/code/tf_models/research/object_detection/utl
ls/label_map_util.py:138: The name tf.gfile.GFile is deprecated. Please use tf.i
o.gfile.GFile instead.
```

Figure A 3: Team4 Start Program [7]

- 8) Once you believe you've collected enough data, plug in a USB to transfer the data
 - a. You will need to name the USB device: **ECS**
- 9) Press the M key on your keyboard to transfer the data
 - a. If the transfer was unsuccessful, you can move the data text files onto your USB device:
 - i. **datatext.txt** for parking data
 - ii. **carcount.txt** for entrance/exit data
- 10) Go to **Error! Reference source not found.**section C of this appendix to learn how to read your data

B. Modifying the detection parameters

Parameters for the programs can be easily adjusted by editing the Python programs. To edit the program, simply use a text editor of your choice: vim, nano, or gedit.

In the following example, we will show how to modify the parameters for the parking spot detection boxes in `app_car_park.py`. The parking spots are determined by the size referencing the origin points of the input video feed's resolution. By default, the feed is set to 720p which is 1280 by 720 pixels (horizontal/progressive). The bounding boxes uses the video width and video height and multiplies with a percentage value to find the top-left and bottom-right coordinates of the bounding boxes. Each parking spot detection box coordinate is contained in an array. To modify these coordinates, change the values for `parking_spot_tl` and `parking_spot_br`.

```
parking_spot_tl = [(int(IM_WIDTH*0.27),int(IM_HEIGHT*0.44))
parking_spot_br = [(int(IM_WIDTH*0.42),int(IM_HEIGHT*0.57))
```

Figure A 4: Team4 Car Park App Spots [7]

By default, the deployable prototype the array contains coordinates for 4 (four) parking spot detection boxes. If you want to include more bounding boxes, you will need to increase the elements for the data arrays. Add another zeroed element in: **buffer_counter**, **car_count**, **grab_vehicle**, **grab_object_class**, **park_counter**, and **pause**.

```
# Initialize variables for parking spot detection
buffer_counter = [0, 0, 0, 0]
car_count = [0, 0, 0, 0]
grab_vehicle = [0, 0, 0, 0]
grab_object_class = [0, 0, 0, 0]
park_counter = [0, 0, 0, 0]
pause = [0, 0, 0, 0]
```

Figure A 5: Team4 Car Park App Integer Parameters [7]

Add a default False value to the **park_detector** array.

```
park_detector = [False, False, False, False]
```

Figure A 6: Team4 Car Park App Boolean Parameters [7]

Last, add a string value to the **vehicle_kind** array. Match the default values.

```
vehicle_kind = ['N/A', 'N/A', 'N/A', 'N/A']
```

Figure A 7: Team4 Car Park App String Parameters [7]

For the entrance and exit car counting application (**app_car_count.py**), the method is similar to modifying the coordinates for the car parking detection application. To move the entrance detection box coordinates, modify the **entrance_tl** (top-left coordinate) array and **entrance_br** (bottom-right coordinate) array.

```
entrance_tl = (int(IM_WIDTH*0.32),int(IM_HEIGHT*0.3))
entrance_br = (int(IM_WIDTH*0.43),int(IM_HEIGHT*.70))
```

Figure A 8: Team4 Counting App Entrance Coordinate [7]

For the exit detection bounding box, modify the **exit_tl** (top-left coordinate) array and the **exit_br** (bottom-right coordinate) array.

```
exit_tl = (int(IM_WIDTH*0.5),int(IM_HEIGHT*0.55))
exit_br = (int(IM_WIDTH*0.75),int(IM_HEIGHT*.64))
```

Figure A 9: Team4 Counting App Exit Coordinate [7]

C. Uploading the collected data to the database

Once the data is transferred to the flash drive, the data can now be transferred to the database stored on the user's computer.

- 1) Insert the flash drive into the computer through the USB port
- 2) Run the GUI by selecting the guitest.py program
- 3) To upload the parking data, click the "Import Parking Data" button



Figure A 10: Team4 Import Parking Data [11]

4) To upload the car counting data, click the “Import Car Count Data” button

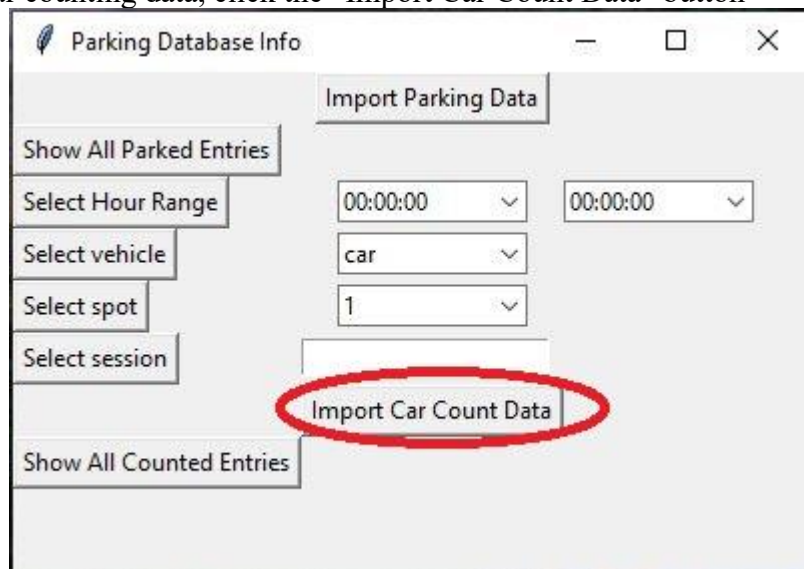


Figure A 11: Team4 Import Car Count Data [11]

Now that your data is uploaded, you can now start viewing and filtering your data

1) To show all the entries in the parked car database, click the “Show All Parked Entries” button.

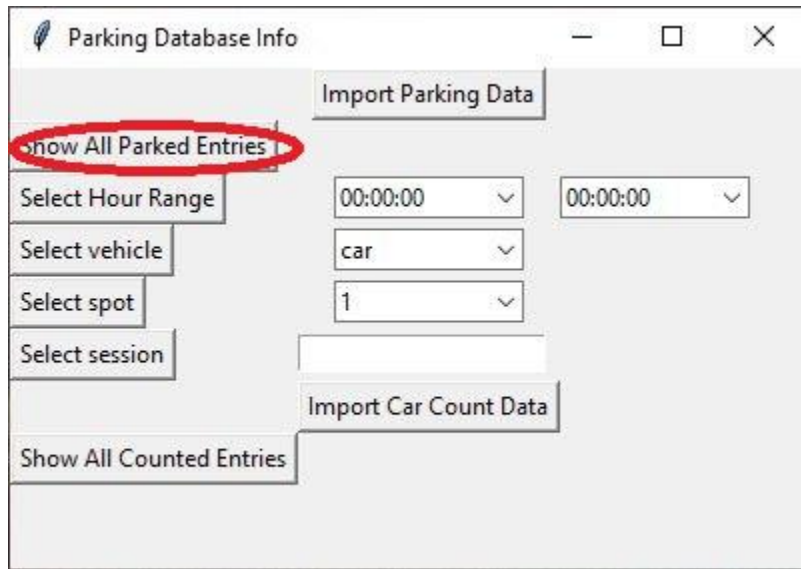


Figure A 12: Team4 Show All Parked Entries [11]

- 2) To show entries that are parked during a certain time range, click the scroll boxes to the right of the “Select Hour Range” button to select your start and end hours. Then press the “Select Hour Range” button.

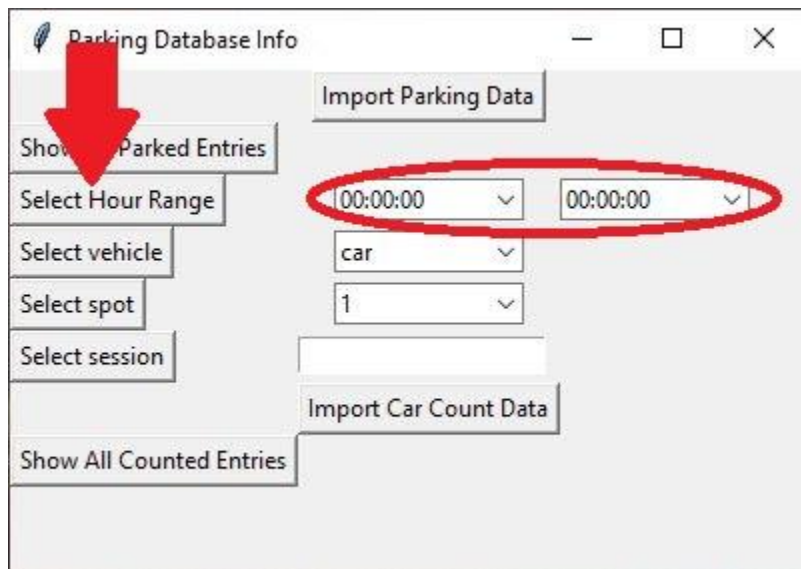


Figure A 13: Team4 Select Hour Range [11]

- 3) To show entries of a certain vehicle type, click the scroll box to the right of the “Select vehicle” button, select the vehicle you want to view and then press the “Select vehicle” button.

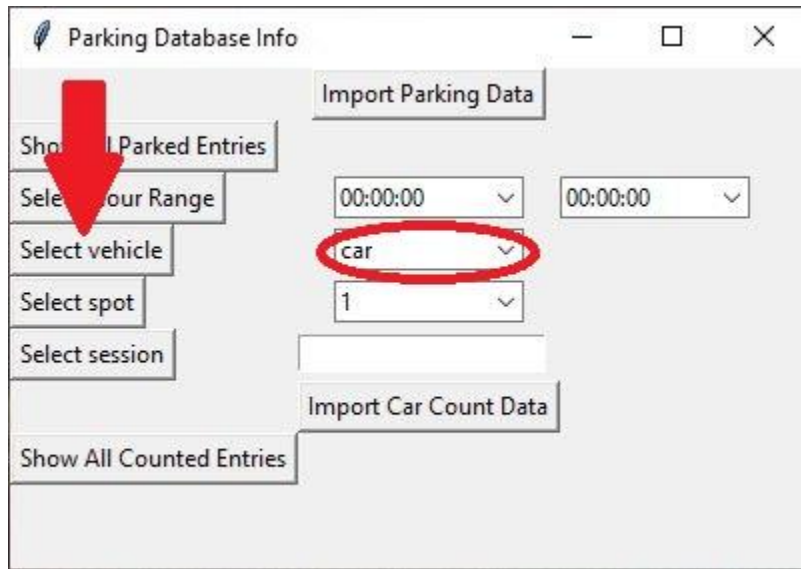


Figure A 14: Team4 Select Vehicle [11]

- 4) To show entries of a certain spot, click the scroll box to the right of the “Select spot” button, select the spot number you want to view and then press the “Select spot” button.

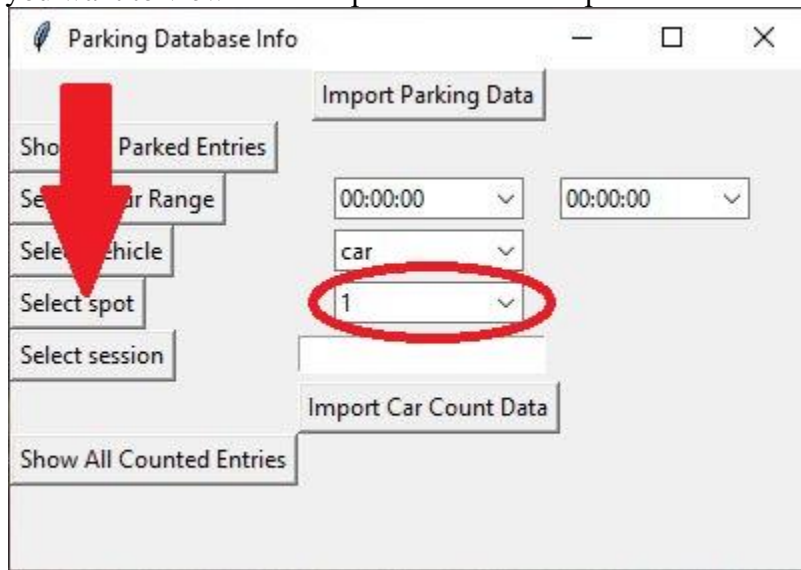


Figure A 15: Team4 Select Spot [11]

- 5) To show entries of a certain session, click the box to the right of the “Select session” button and input the session number you want to view and then press the “Select session” button.

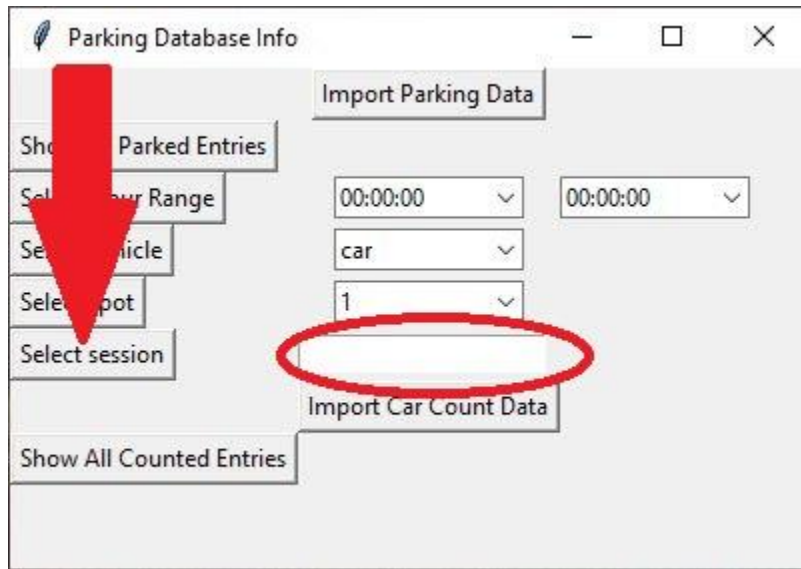


Figure A 16: Team4 Select Session [11]

- a) Note: The session numbers are based off how many data collection sessions you have.
 - i) To view your first session, type “1” into the input box and then click the “Select session” button.

D. Charging

Charging the system is a very simple task. Simply connect a 9V power adapter into the female barrel DC jack. Flip the right switch to turn on the voltmeter and charge the system until the voltmeter reaches 7.1 volts. The device can also be used while charging.

APPENDIX B. HARDWARE

The hardware used for this project consisted of 12 Samsung 25R 18650 cells and a 2S BMS, which make up the power bank; an NVIDIA Jetson Nano as the main processor; a USB camera to capture the footage for the system; a DC voltmeter to display the voltage available in the power bank; a 5v Noctua NF-A4x20 cooling fan for processor protection; and an LM 2596 DC voltage regulator to ensure the processor isn't fried. All the hardware components are enclosed in a 3D printed shell.

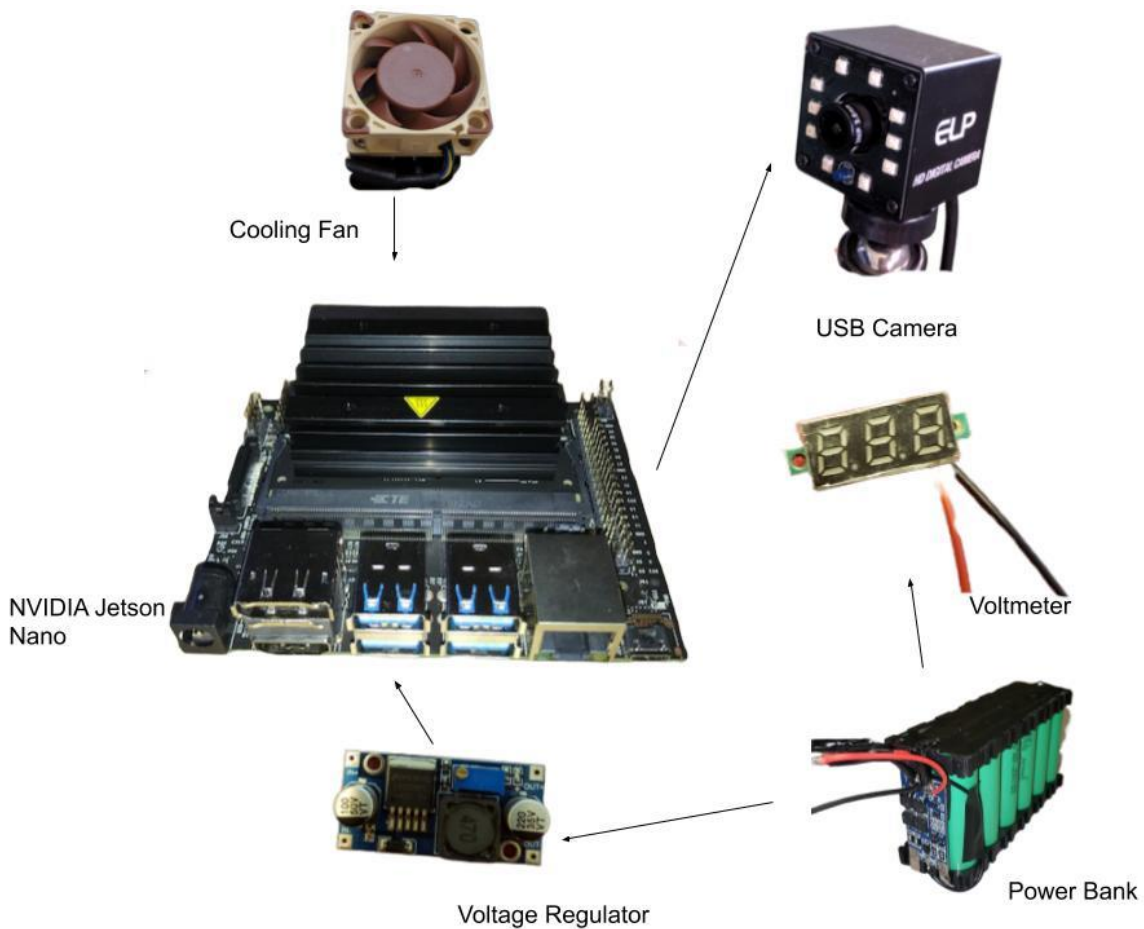


Figure B 1: System Hardware [29]

The prolonged use trials were conducted by running the car detection program from the lab prototype on an endless loop. While the video looped endlessly the voltage displayed on the voltmeter would be collected at intervals of ten minutes. Two tests were conducted, one of them ran for 1 hour and yielded the results shown on table 3. The second test conducted ran for 3 hours and offered a much better understanding of the power bank's capacity. Both tests were conducted with the development board operating at max power, meaning it was operating at 20 watts. Our program, however, only requires 5 watts to operate, thus the power bank can offer around four times more usage per charge than tested. The power bank can reliably power the system at 20 watts for 3 hours and 30 minutes; but since our program only requires 5 watts to operate, the power bank can reliably power our system for over 10 hours. And as a safety measure, the power bank can also be charged while being operated.

Table B I.
Prolonged Use Test 1 [10]

Voltage	Time (min)
6.82	0
6.78	10
6.66	20
6.54	30
6.38	40
6.24	50
6.11	60

Table B II.
Prolonged Use Test 2 [10]

Voltage	Time (min)
7.04	0
7.00	10
6.97	20
6.92	30
6.89	40
6.87	50
6.86	60
6.85	70
6.83	80
6.81	90
6.78	100
6.68	110
6.58	120
6.45	130
6.34	140
6.20	150
6.05	160
5.92	170
5.80	180

APPENDIX C. SOFTWARE

Software Flowcharts

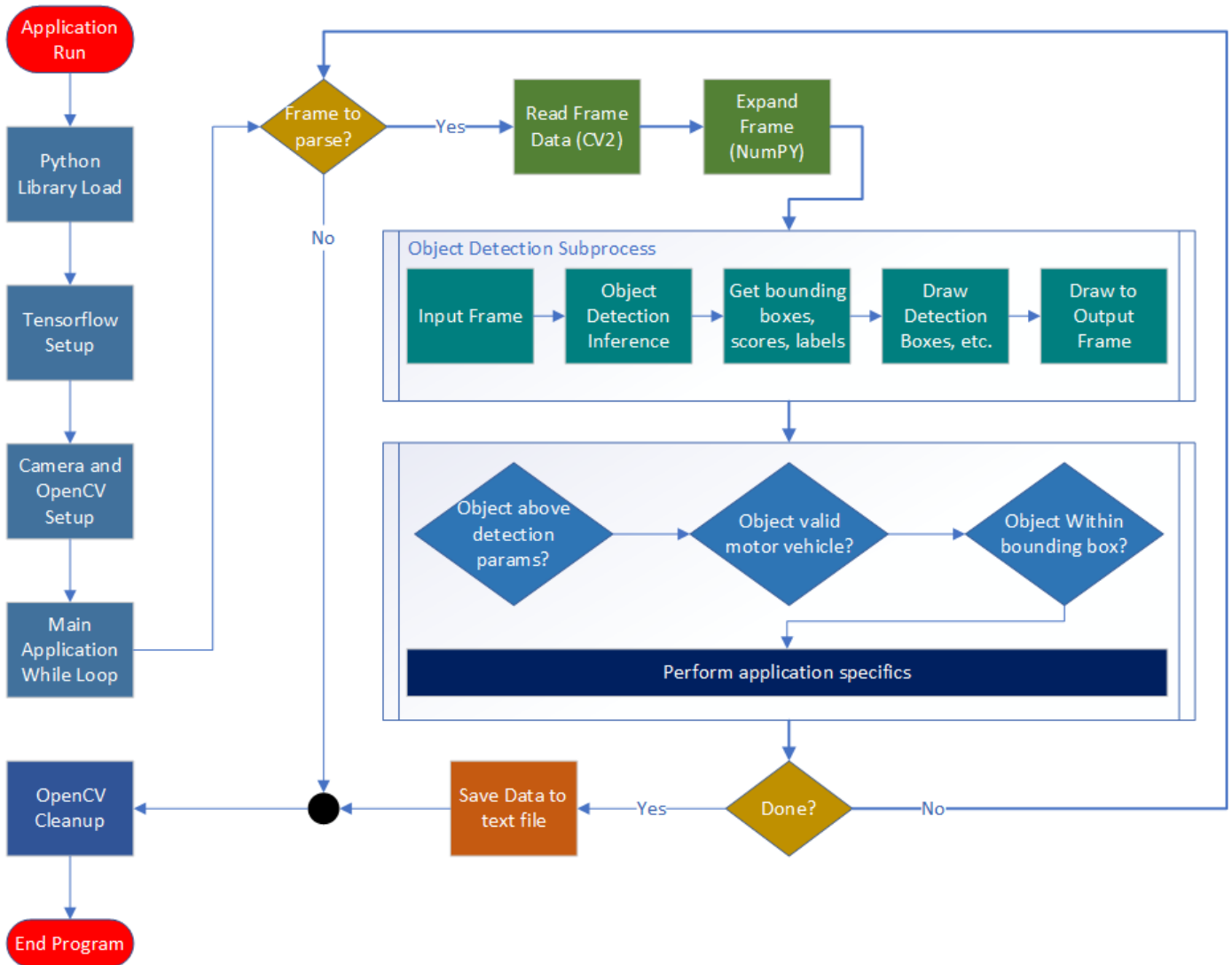


Figure C 1: Team4 Python Application Workflow [7]

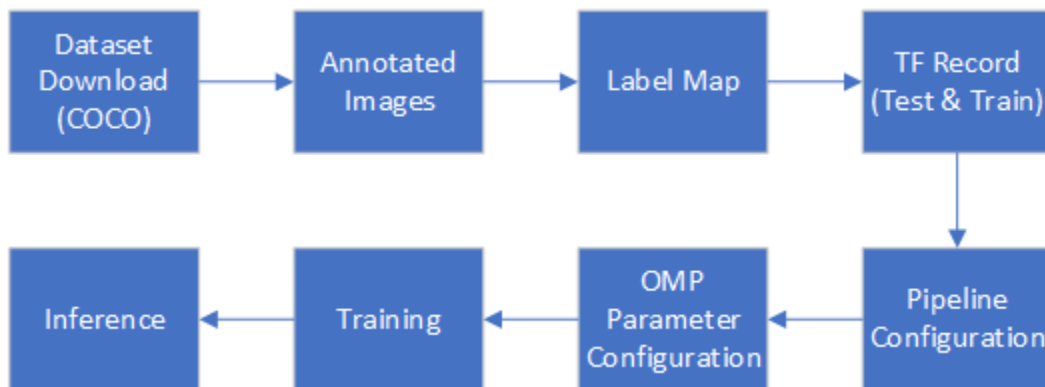


Figure C 2: TensorFlow Object Detection API [30]

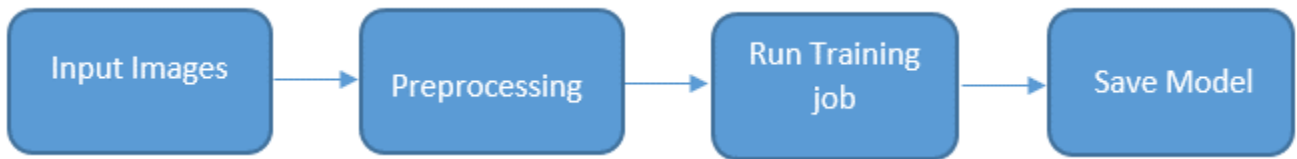


Figure C 3: Training Pipeline [30]

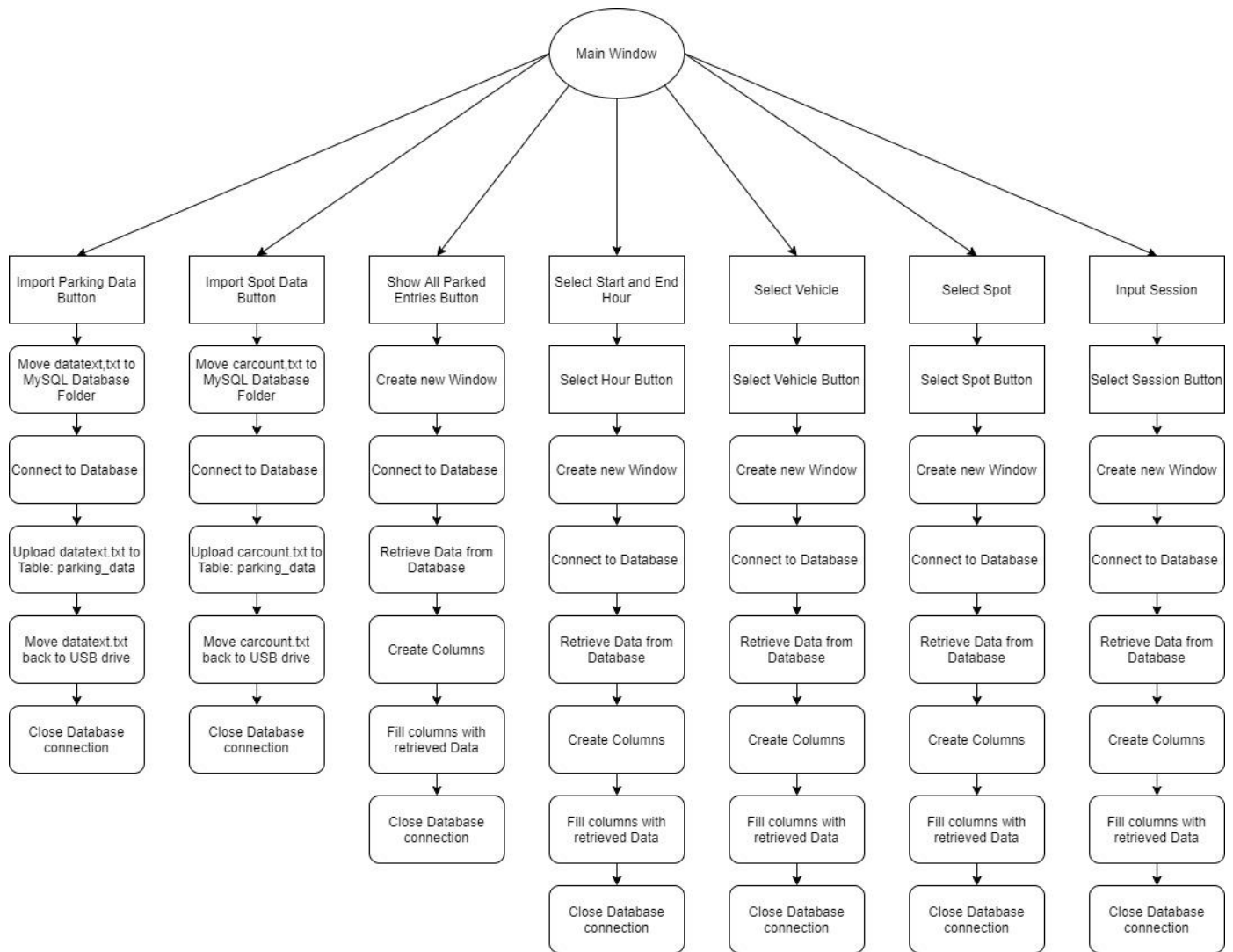


Figure C 4: GUI Flowchart [11]

Application Specific Code Review for the Car Parking Detection Program

The following is code from **app_car_park.py**. The subroutine shown here shows how the detected object checks whether the vehicle is within the confidence level threshold, then checks if the detected object is a valid motor vehicle (determined by the label map), and finally checks if the detected object is within the coordinates of the detection boxes.

```
153
154 # Check if detected object is a valid motor vehicle, (1 = Sedan, 2
      Truck, 3 = Bus, 4 = Motorcycle)
155 # Then get the center x and y position:
156 i = 0
157 while i < objects_num:
158     object_class = int(classes[0][i])
159     if (float(scores[0][i]) > AVG_CONFIDENCE_THRESHOLD):
160         if((object_class == 3 or object_class == 4 or object_class == 6 or
            object_class == 8)):
161             x_temp = (int(((boxes[0][i][1]+boxes[0][i][5])/2)*IM_WIDTH))
162             y_temp = (int(((boxes[0][i][0]+boxes[0][i][4])/2)*IM_HEIGHT))
163             cv2.circle(frame, (x_temp, y_temp), 5, (75,13,180), -1)
164             j = 0
165             while(j < spot_count):
166                 if ((x_temp > parking_spot_tl[j][0]) and (x_temp <
                    parking_spot_br[j][0]) and (y_temp > parking_spot_tl[j][1]) and
                    (y_temp < parking_spot_br[j][1])):
167                     park_inspace[j] = True
168                     park_counter[j] = park_counter[j] + 1
169                     buffer_counter[j] = 0
170                     if grab_vehicle[j] == 0:
171                         grab_object_class[j] = object_class
172                         grab_vehicle[j] = 1
173                     j = j + 1
174 i = i + 1
```

Line 167 to line 173 increments an arbitrary counter for each parking spaces detection box. This continues to tick up as long as the detected object is still within the detection box. At this point, the program will grab the vehicle type (or class) and the parking spot number.

The next line of code executes after the detected vehicle leaves the parking detection box after a certain amount of frame ticks (this is the `buffer_counter` variable). This check executes for every parking spot.

```
177 while i < spot_count:
178     if park_counter[i] > 0:
179         buffer_counter[i] = buffer_counter[i] + 1
180
181     # If no vehicle is within the spot (false alarm), buffer by counting up to 50
        frames then reset counters
182     if buffer_counter[i] > 50:
183         buffer_counter[i] = 0
184         park_counter[i] = 0
185         pause[i] = 0
186         grab_vehicle[i] = 0
187         park_detector[i] = False
188         end_time_val = t
189         end_time = t.strftime("%H:%M:%S")
190         datafile = open("datatext.txt", "a+")
191         idfile = open("entryid.txt", "r+")
192         entry_id = int(idfile.read())
193         datafile.write("%d\t" % entry_id) #Unique key (for SQL)
194         spot = i + 1
195         datafile.write("%d\t" % spot) #Parking spot
196         datafile.write("%s\t" % start_time) #Need to set entry_time when car occupies
            spot
197         datafile.write("%s\t" % end_time) #Need to set exit time when car leaves spot
198         datafile.write("%s\t" % vehicle_kind[i]) #Vehicle type
199         car_count[i] = car_count[i] + 1
200         datafile.write("%d\t" % car_count[i]) #car count
201         datafile.write("%d\t" % session_id) #session
202         datafile.write("%s\n" % str(end_time_val - start_time_val)[:7]) #duration
203         datafile.close()
204         entry_id = entry_id + 1
205         idfile.seek(0)
206         idfile.truncate()
207         idfile.write(str(entry_id))
208         idfile.close()
```

The code resets the values for the parking spot and issues a buffer counter. This will prepare the code for the next detected vehicle that enters the detection box for the same parking spot. The code also writes all the data collected onto the `datatext.txt` file.

Application Specific Code Review for the Car Counting Program

The following code is for the **app_car_count.py** program. This code is similar to the **app_car_park.py**, but only checks for the two detection boxes: entrance and exit. This is executed as separate if-statements for every validly detected object.

```
153
154     i = 0
155     while i < objects_num:
156
157         # Check to see if detected object is equal to or above the threshold
158         # And check if detected object is a valid motor vehicle, (3 = car, 4 = bus, 6 =
            truck, 8 = motorcycle)
159         object_class = int(classes[0][i])
160         object_scores = float(scores[0][i])
161         if((object_scores >= AVG_CONFIDENCE_THRESHOLD) and (object_class == 3 or
            object_class == 4 or object_class == 6 or object_class == 8)):
162
163             # Get the center x and y position
164             x = int(((boxes[0][i][1]+boxes[0][i][5])/2)*IM_WIDTH)
165             y = int(((boxes[0][i][0]+boxes[0][i][4])/2)*IM_HEIGHT)
166             cv2.circle(frame, (x,y), 5, (75,13,180), -1)
167
168             # If passes entrance, draw a circle at center of object, and increment entrance
            counter
169             if ((x > entrance_tl[0]) and (x < entrance_br[0]) and (y > entrance_tl[1]) and (y
            < entrance_br[1])):
170                 entrance_detection_counter = entrance_detection_counter + 1
171                 entrance_buffer_counter = 0
172                 if grab_vehicle[0] == 0:
173                     grab_object_class[0] = object_class
174                     grab_vehicle[0] = 1
175
176             # Do the same for exit area
177             if ((x > exit_tl[0]) and (x < exit_br[0]) and (y > exit_tl[1]) and (y <
            exit_br[1])):
178                 #cv2.circle(frame, (x,y), 5, (75,13,180), -1)
179                 exit_detection_counter = exit_detection_counter + 1
180                 exit_buffer_counter = 0
181                 if grab_vehicle[1] == 0:
182                     grab_object_class[1] = object_class
183                     grab_vehicle[1] = 1
184
185
186     i = i + 1
187
```

Application Specific Code Review for the GUI Program

The following section is an explanation of the GUI program. It uses the tkinter library to create the GUI objects (windows, displayed text, buttons, etc.), the mysql.connector library to connect to the MySQL database and utilize the SQL language, and the shutil library to move files. Upon running the GUI program, the user is presented with a home window. From here the user can decide to do the following:

- 1) Import parking data from *datatext.txt* to the database in the table: *parking_data*
- 2) Display all entries currently in the database table: *parking_data*
- 3) Display entries that fit user selected time range in the database table: *parking_data*
- 4) Display entries that fit user selected vehicle type in the database table: *parking_data*
- 5) Display entries that fit user selected spot number in the database table: *parking_data*
- 6) Display entries that fit user inputted session number in the database table: *parking_data*
- 7) Import car count data from *carcount.txt* in the database table: *count_data*

*1) Import parking data from *datatext.txt* to the database in the table: *parking_data**

Pressing the “Import Parking Data” button on the GUI main window will move the *datatext.txt* file into the MySQL Uploads folder. Once this is done, the GUI will connect to the MySQL database. Once connected, an SQL command is run to upload the data in *datatext.txt* to the *parking_data* table. The text file is then moved back to the USB drive and the update to the database is committed. Connection with the database is then closed.

*2) Display all entries currently in the database table: *parking_data**

Pressing the “Show All Parked Entries” button on the GUI main window will create another window to display the soon to be retrieved data. Next, the GUI will connect to the MySQL database. Once connected, an SQL command is run to retrieve all the data entries in the *parking_data* table. The data is saved to a variable. The columns and scroll wheel are then created to display the retrieved data. The columns are filled in via a for loop with the retrieved data. The connection to the database is then closed. The user can either close the window and start a new query or keep the window open and start another query.

*3) Display entries that fit user selected time range in the database table: *parking_data**

After a user selects a start hour and end hour in the dropdown boxes, a new window is opened when the user clicks the “Select Hour” button. The hour values are saved to a variable to be used later. The GUI will then establish a connection to the MySQL database. Once connected, an SQL command is run to retrieve data from the database using the variables that stored the start hour and end hour selected from the main window. The retrieved data entries are then saved to a variable. The columns are filled in via a for loop on the retrieved data variable. The connection to the database is then closed. The user can either close the window and start a new query or keep the window open and start another query.

*4) Display entries that fit user selected vehicle type in the database table: *parking_data**

After a user selects a vehicle type in the dropdown box, and the user clicks the “Select Vehicle” button, a new window is opened. The vehicle type is saved to a variable to be used for later. The GUI will then establish a connection to the MySQL database. Once connected, an SQL command is run to retrieve data from the database using the variable that stored the user selected vehicle type on the main window. The retrieved data entries are then saved to a variable. The columns are filled in via a for loop on the retrieved

data variable. The connection to the database is then closed. The user can either close the window and start a new query or keep the window open and start another query.

5) Display entries that fit user selected spot number in the database table: parking_data

After a user selects a spot number in the dropdown box, and the user clicks the “Select Spot” button, a new window is opened. The spot number is saved to a variable to be used for later. The GUI will then establish a connection to the MySQL database. Once connected, an SQL command is run to retrieve data from the database using the variable that stored the user selected spot number on the main window. The retrieved data entries are then saved to a variable. The columns are filled in via a for loop on the retrieved data variable. The connection to the database is then closed. The user can either close the window and start a new query or keep the window open and start another query.

6) Display entries that fit user inputted session number in the database table: parking_data

After a user inputs a session number in the entry field, and the user clicks the “Select Session” button, a new window is opened. The session number is saved to a variable to be used for later. The GUI will then establish a connection to the MySQL database. Once connected, an SQL command is run to retrieve data from the database using the variable that stored the user selected spot number on the main window. The retrieved data entries are then saved to a variable. If the variable holding the retrieved entries is 0, then the GUI pops up a window that says an invalid session number was entered. If a variable holding the retrieved entries is not 0, the columns are filled in via a for loop on the retrieved data variable. The connection to the database is then closed. The user can either close the window and start a new query or keep the window open and start another query.

Software Test Results

Accuracy and usefulness of the object detection is very much related with the combination of the software and camera hardware. For this section of the software test results, we've included test results of the lighting. In order to test light conditions, a circuit was built employing an Arduino Nano and a TEMA6000 light sensor to measure the lux at the important areas of the testing location. The datasheet claims the TEMA6000 measures in lux, but cross tests will be performed with a different lux meter to confirm this [31]. The ambient light was measured at the position of the camera, at the position of the vehicle, and recording the ambient light that makes its way into the parking structure from outside. These tests were conducted at the third floor of parking structure 3 and parking structure 5 in California State University, Sacramento. As well as the third and fourth floor of parking structure 1. Parking structure 5 has plenty of lighting while parking structure 1 is the darkest of all the structure.

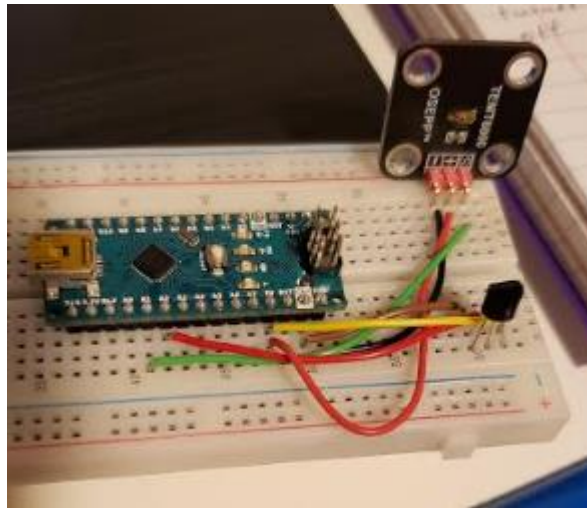


Figure C 5: TEMA6000 circuit [10]

Through these tests we found that the best lighting for image is along 1,000 lx, which is the value for an overcast day. This level of light minimizes shadow and glare but also allows for a clear image. None of the structure at Sacramento State present these ideal lux levels but that was expected. Parking Structure 1 only presented an average lux level of around 3 lux, so it is the worst option for our system.

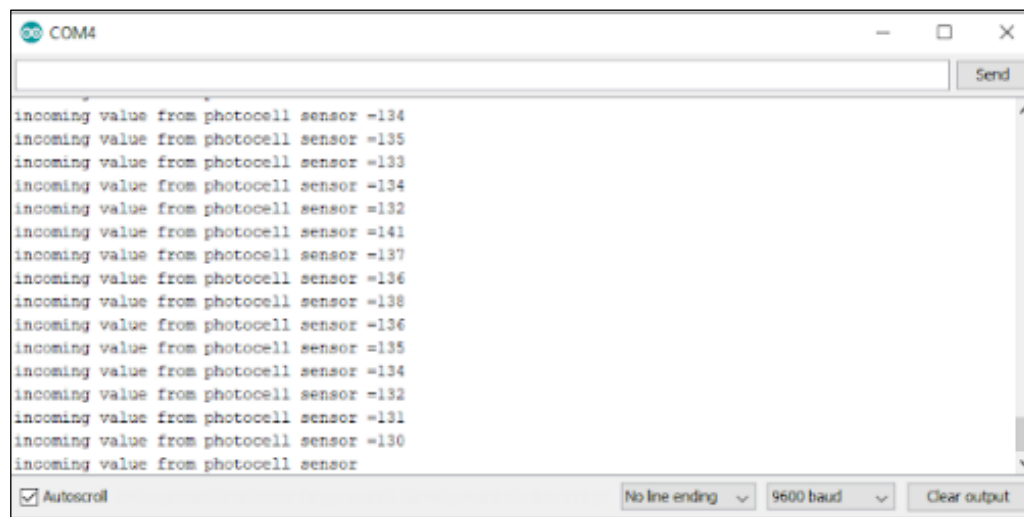


Figure C 6: Structure 5 lux level [10]

Parking Structure 5 offered the best lux levels, however, due to the LED lights the white vehicles almost blended in with the surroundings. Darker vehicles were more easily detected.

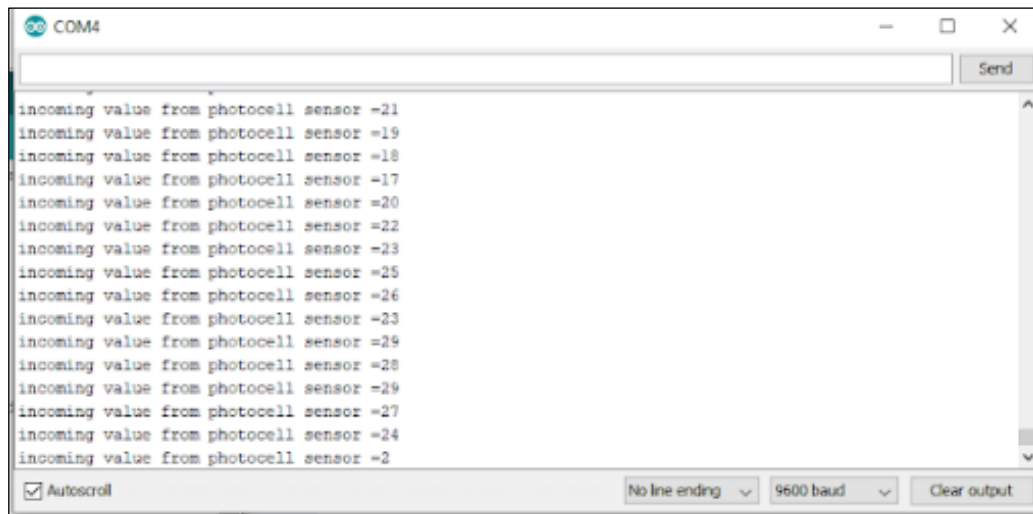


Figure C 7: Structure 3 lux level [10]

Parking Structure 3 offered less total lux levels, but the lighting worked nicely with white vehicles. There was enough lighting to clearly view a vehicle and white vehicle didn't get lost in the lux.

So, from our tests we concluded that when being placed in an external parking area the ideal lighting is around 1000 lux, so an overcast day. But if placed in a parking structure, the higher the lux the better, and LED lights will cause errors.

Testing was done on the system's GUI and database. We tested the GUI's ability to transfer data from a flash drive which recorded data from the NVIDIA Jetson Nano. Specifically, we tested the robustness of the GUI which should be able to transfer data from a text file while simultaneously preventing the transfer of entries that have the same primary keys as entries that are already in the database. We also tested the GUI's runtime when transferring text files with large amounts of entries to the database. We also tested the runtime of the GUI when pulling entries from the database. Finally, we tested the GUI's ability to correctly filter entries in the database. Testing of the GUI and database was done on a laptop with an Intel(R) Core (TM) i7-4710HQ CPU running at 2.50 GHz. The GUI was coded in Python 2.7 and the database is a MySQL Community Version 8.0.

The first GUI and database test were to determine whether or not entries in a text file, to be transferred to a database, will transfer if the primary keys of the entries in the text file conflict with ones in the database. To do this test, we used a set of sample data that has entries that conflict with the primary keys of entries in the database seen in figures 6 and 7. The primary key is the left most column.

1	1	070000	070010	truck	1	1
2	2	070000	071505	bus	1	1
3	3	070100	071606	sedan	1	1
4	4	070150	071707	bus	1	1
5	2	071650	071808	sedan	2	1
6	4	071850	071906	truck	2	1
7	4	072000	072101	truck	3	1
8	1	070150	072202	sedan	2	1
9	2	072150	072500	bus	3	1
10	4	072300	072630	sedan	4	1
11	1	073000	074501	bus	3	1
12	2	072700	080340	sedan	4	1
13	3	072004	090402	bus	2	1
14	4	072805	090450	truck	5	1
15	1	075500	120030	sedan	4	1
16	2	070050	070902	truck	1	2
17	1	071043	073503	sedan	1	2
18	3	071402	080010	bus	1	2
19	2	070050	090306	truck	2	2
20	4	070404	100305	sedan	1	2
21	1	074509	080302	sedan	1	3
22	2	074007	080520	truck	1	3
23	3	074105	120904	bus	1	3
24	2	060530	063400	sedan	1	4
25	1	060720	073021	truck	1	4
26	4	082546	083801	sedan	1	5
27	1	083851	093211	bus	1	5
28	1	094543	094838	sedan	2	5

Figure C 8: Primary Key Test: Sample Data in Database [11]

1	1	140000	150000	truck	4	3
2	1	170000	183000	bus	4	3
3	3	174503	184620	bus	4	3
29	1	210507	220406	car	1	7

Figure C 9: Primary Key Test: Sample Data to Transfer to Database [11]

Included in the sample data is also an entry with a valid primary key, “29”. This entry should be added to the database while the first three should not. After trying to upload the data in figure 4, the database was not updated with the conflicting primary key entries. However, the valid entry was also not uploaded to the database. Uploading the text file in the MySQL command prompt will accept the valid primary entry, and it will also deny the invalid primary key entries. However, to accomplish this through our GUI, we need to include the “IGNORE” SQL command when transferring data from the text file. The “IGNORE” command will ignore any invalid primary key entries but will transfer entries with valid primary keys.

Regarding transferring data to the database, our next test determines the average runtime the GUI needs to transfer large amounts of data. The first test consisted of the transfer of a text file with 400 entries. This transfer was done 28 times. In order to get the runtime, the “time” library was used to get the system time at the beginning of the transfer. This time was then compared to the system time pulled at the end of the transfer and their difference represents the runtime of the transfer, which is in seconds. The average

runtime was calculated to be approximately 0.19806 seconds. The runtimes and average runtime can be seen in table 3. 28 transfers were also done with a text file with 1,000 entries. The average runtime was approximately 0.23599 seconds. The runtimes and average runtime can be seen in table 3. It should be noted that the version of MySQL we are using for the database, MySQL Community Version 8.0, needs to disable “Safe Update” in the preferences in order to delete all the entries in the database with a single line, “DELETE FROM parking_data”, where “parking_data” is the name of our table that holds all the entries in our database.

Table C I.

Runtimes in seconds of GUI to upload 400 entries (left) and 1,000 entries (right) [11]

	Upload 400 runtime	Upload 1000 runtime
	0.0686345100402832	0.186171293258667
	0.2431592941284170	0.154010057449340
	0.1899352073669430	0.163047313690185
	0.1264197826385490	0.163047313690185
	0.2002503871917720	0.291187286376953
	0.4227843284606930	0.176358938217163
	0.4227843284606930	0.216327190399169
	0.2037813663482660	0.182073831558227
	0.1291139125823970	0.165365457534790
	0.2182135581970210	0.196045160293579
	0.1309826374053950	0.386857748031616
	0.1702976226806640	0.173931121826171
	0.1351537704467770	0.189947128295898
	0.1490693092346190	0.160087347030639
	0.1958782672882080	0.343049526214599
	0.1506774425506590	0.382721424102783
	0.1459155082702630	0.181707859039306
	0.1998605728149410	0.207238912582397
	0.2758593559265130	0.249595642089843
	0.1606729030609130	0.219284057617187
	0.1157140731811520	0.163952589035034
	0.1962301731109610	0.170810937881469
	0.4152402877807610	0.140136003494262
	0.2060668468475340	0.446177959442138
	0.1321496963500970	0.413292407989501
	0.1446413993835440	0.413292407989501
Average	0.1980571746826170	0.235989112120408

The average runtime from pulling data from the database was also calculated. The method to calculate the runtime is the same method used for the test that transfers large amounts of data to the database. The GUI pulls all the entries in the database. We had 28 trials of the runtime for both 400 entries returned from the database and 1000 entries returned from the database. The average runtime for returning 400 entries was approximately 0.01987 seconds. For returning 1,000 entries, the average runtime was approximately 0.03125 seconds.

Table C II.

Runtimes in seconds of GUI to upload 400 entries (left) and 1,000 entries (right) [11]

	Return 400 runtime	Return 1000 runtime
	0.0315198898315429	0.0312476158142089
	0.0199971199035644	0.0155742168426513
	0.0239598751068115	0.0312447547912597
	0.0159795284271240	0.0319230556488037
	0.0185804367065429	0.0312452316284179
	0.0162780284881591	0.0468981266021728
	0.0199456214904785	0.0359673500061035
	0.0183289051055908	0.0279915332794189
	0.0159766674041748	0.0311832427978515
	0.0160090923309326	0.0396280288696289
	0.0160088539123535	0.0279390811920166
	0.0199289321899414	0.0312471389770507
	0.0180828571319580	0.0279912948608398
	0.0200097560882568	0.0279455184936523
	0.0239098072052001	0.0279445648193359
	0.0199456214904785	0.0280070304870605
	0.0302193164825439	0.0319416522979736
	0.0232706069946289	0.0312411785125732
	0.0199453830718994	0.0311801433563232
	0.0199327468872070	0.0311965942382812
	0.0160088539123535	0.0312590599060058
	0.0199444293975830	0.0311977863311767
	0.0168595314025878	0.0321385860443115
	0.0159931182861328	0.0312478542327880
	0.0159466266632080	0.0359907150268554
	0.0239589214324951	0.0312016010284423
Average	0.0198669433593750	0.0312528060032771

The last set of GUI and database tests were the filter tests. Our GUI allows the user to filter the entries in the database to only return particular entries. The time range filter will return entries that were parked in the monitored spots within a specified time range picked by the user. In order to test the time range filter, we uploaded sample data, seen in figure 8, and tested the GUI's ability to accurately pull entries from the database based off the parameters selected by the user.

1	1	070000	070010	truck	1	1
2	2	070000	071505	bus	1	1
3	3	070100	071606	sedan	1	1
4	4	070150	071707	bus	1	1
5	2	071650	071808	sedan	2	1
6	4	071850	071906	truck	2	1
7	4	072000	072101	truck	3	1
8	1	070150	072202	sedan	2	1
9	2	072150	072500	bus	3	1
10	4	072300	072630	sedan	4	1
11	1	073000	074501	bus	3	1
12	2	072700	080340	sedan	4	1
13	3	072004	090402	bus	2	1
14	4	072805	090450	truck	5	1
15	1	075500	120030	sedan	4	1
16	2	070050	070902	truck	1	2
17	1	071043	073503	sedan	1	2
18	3	071402	080010	bus	1	2
19	2	070050	090306	truck	2	2
20	4	070404	100305	sedan	1	2
21	1	074509	080302	sedan	1	3
22	2	074007	080520	truck	1	3
23	3	074105	120904	bus	1	3
24	2	060530	063400	sedan	1	4
25	1	060720	073021	truck	1	4
26	4	082546	083801	sedan	1	5
27	1	083851	093211	bus	1	5
28	1	094543	094838	sedan	2	5

Figure C 10: Sample Data Used to Test GUI SQL Filters [11]

In the case of the time range filter, the user defines a start hour and an end hour. All entries that have vehicles that are parked in this hour range will be pulled from the database. As shown in figures 9 and 10, the GUI accurately pulls all applicable entries that have vehicles parked from the user selected time range: “08:00:00 – 09:00:00”.

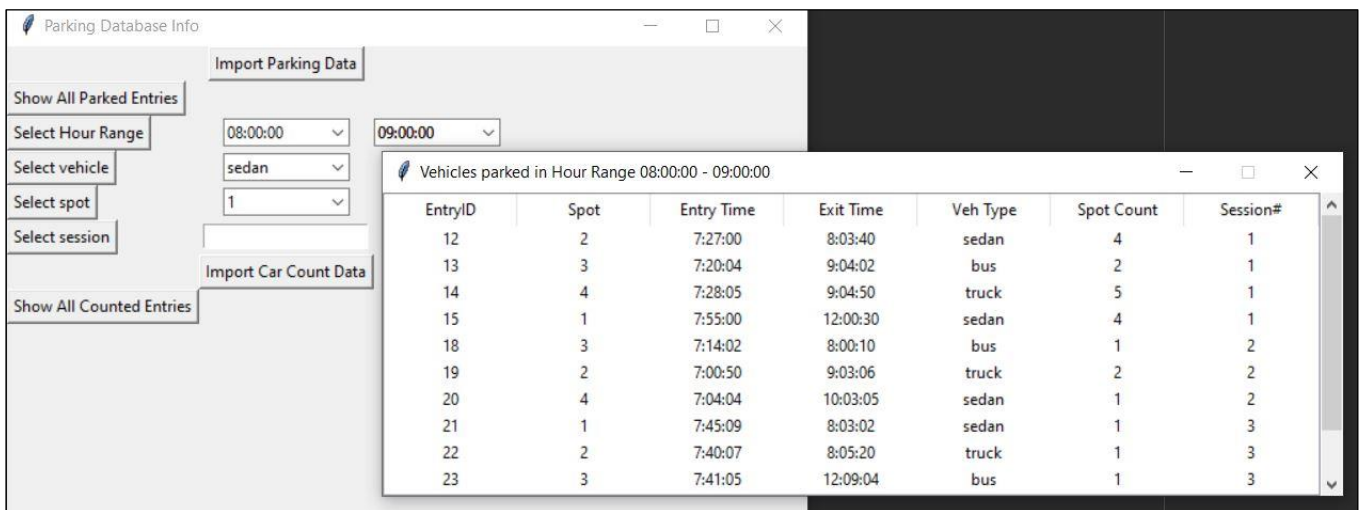


Figure C 11: Time Range Filter of Parking Data: 08:00:00 – 09:00:00 part 1 [11]

Originally, there was a problem with our logic when filtering for a specific time range. The GUI would not return entries with vehicles that were parked before the user selected start time and left after the user selected end time. Entries with EntryID 20 and 23 show that the logic now works since these entries enter before the user selected start time, 08:00:00, and after the user selected end time, 09:00:00.

The next filter tested was the vehicle filter. This filter will pull any entry that has the user defined vehicle type. The results can be seen in figure 11, which pulls all entries that were trucks. Figure 12 shows all entries that were buses. Figure 13 and 14 shows all entries that were cars.

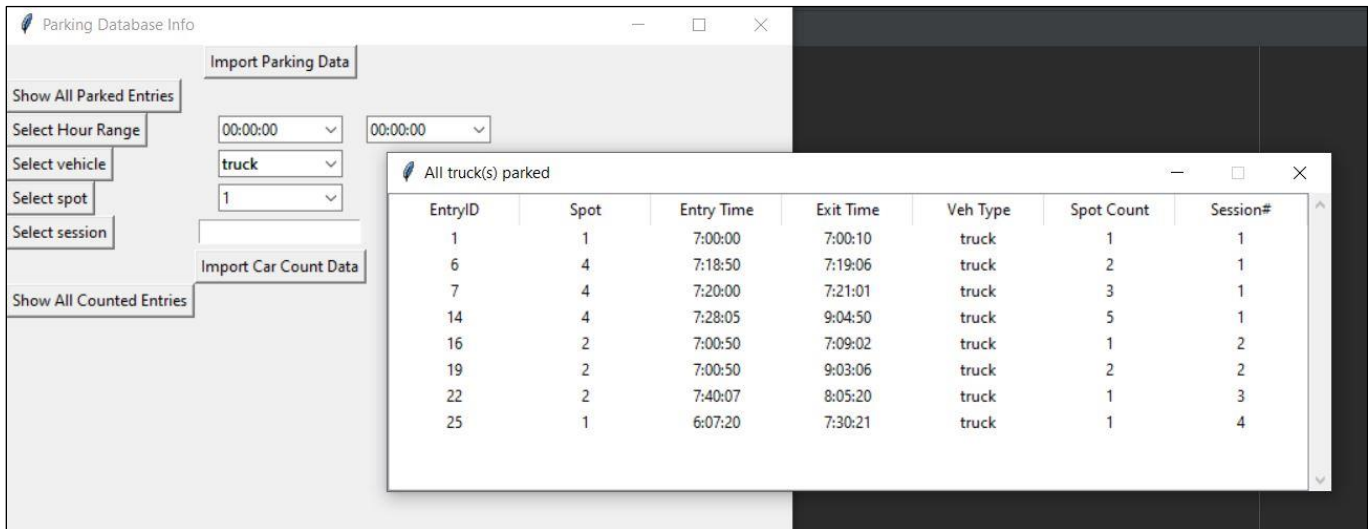


Figure C 12: Vehicle Type Filter of Parking Data: Truck [11]

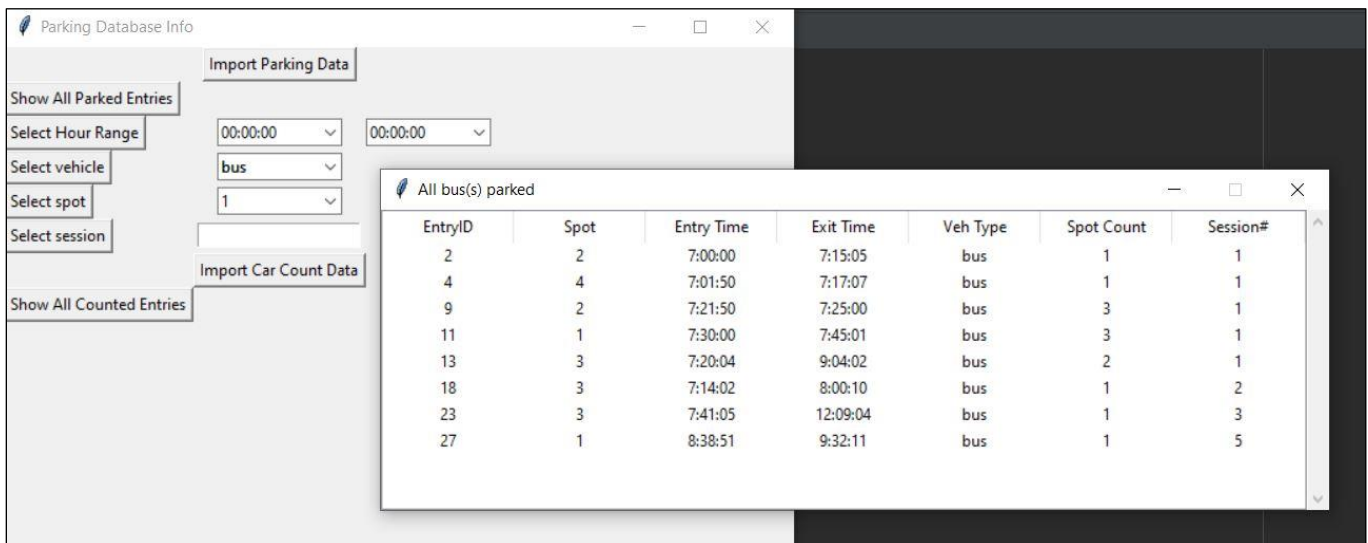


Figure C 13: Vehicle Type Filter of Parking Data: Bus [11]

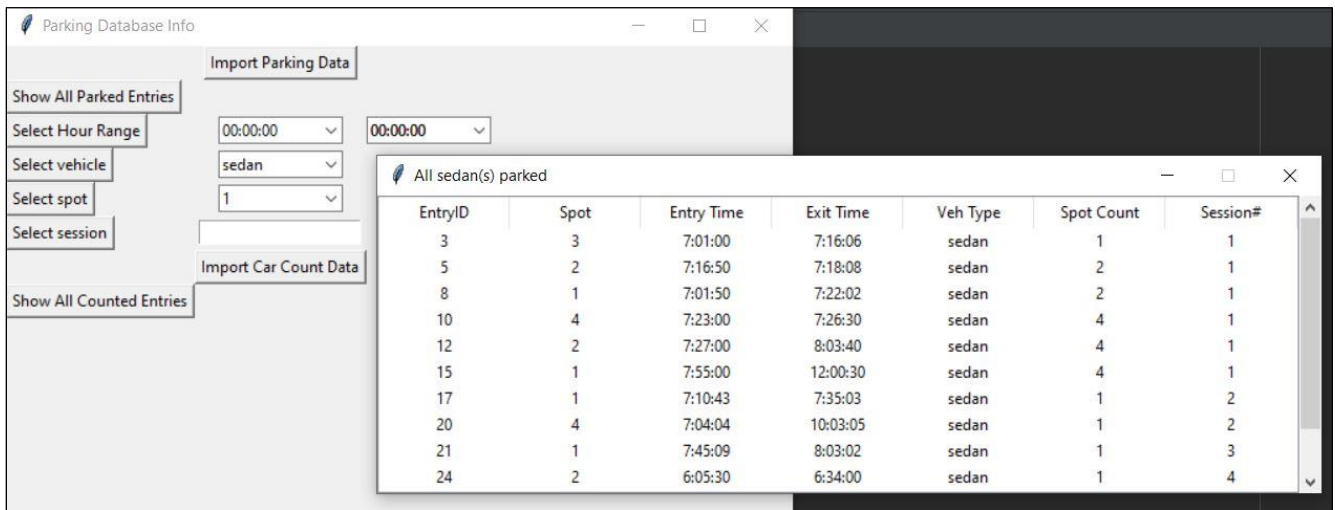


Figure C 14: Vehicle Type Filter of Parking Data: Car part1 [11]

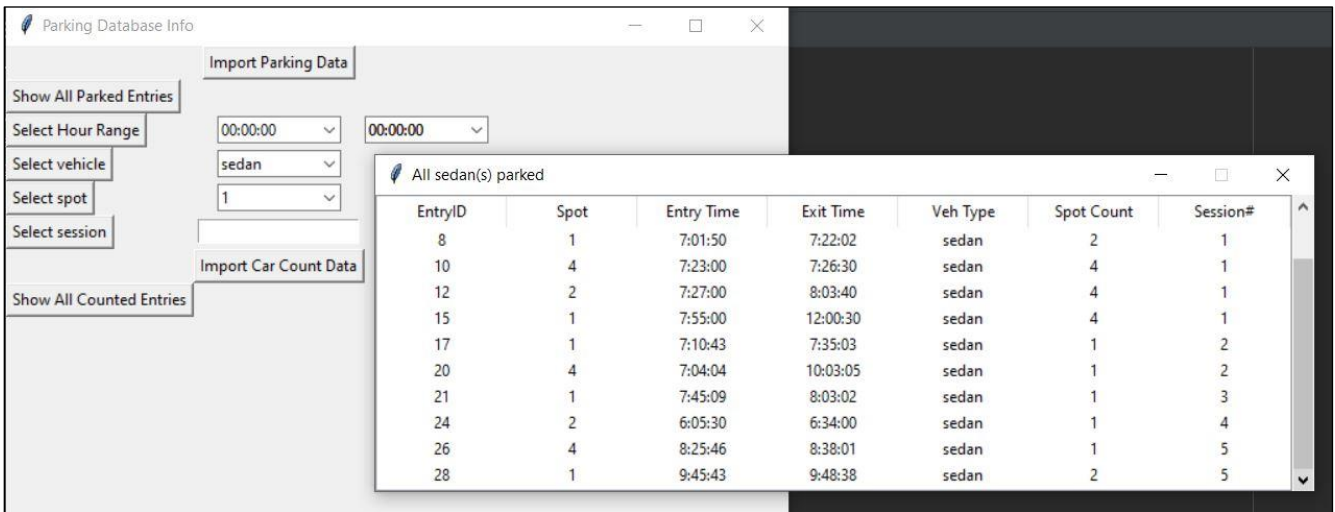


Figure C 15: Vehicle Type Filter of Parking Data: Car part 2 [11]

The combining entries returned from figures 11, 12, 13, and 14 has every, since the combined queries pull every possible vehicle type stored in the sample database.

APPENDIX D. MECHANICAL ASPECTS

The mechanical aspect of this project is simply the 3D printed enclosure; and possibly even a tripod if the user wishes to use one. The enclosure was design using SolidWorks 3D Design Software and printed through two different methods. The center piece that works as the system's harness was printed with polylactid acid. The five walls surrounding the rest of the system were printed using standardized photopolymer resin 6mm thick.

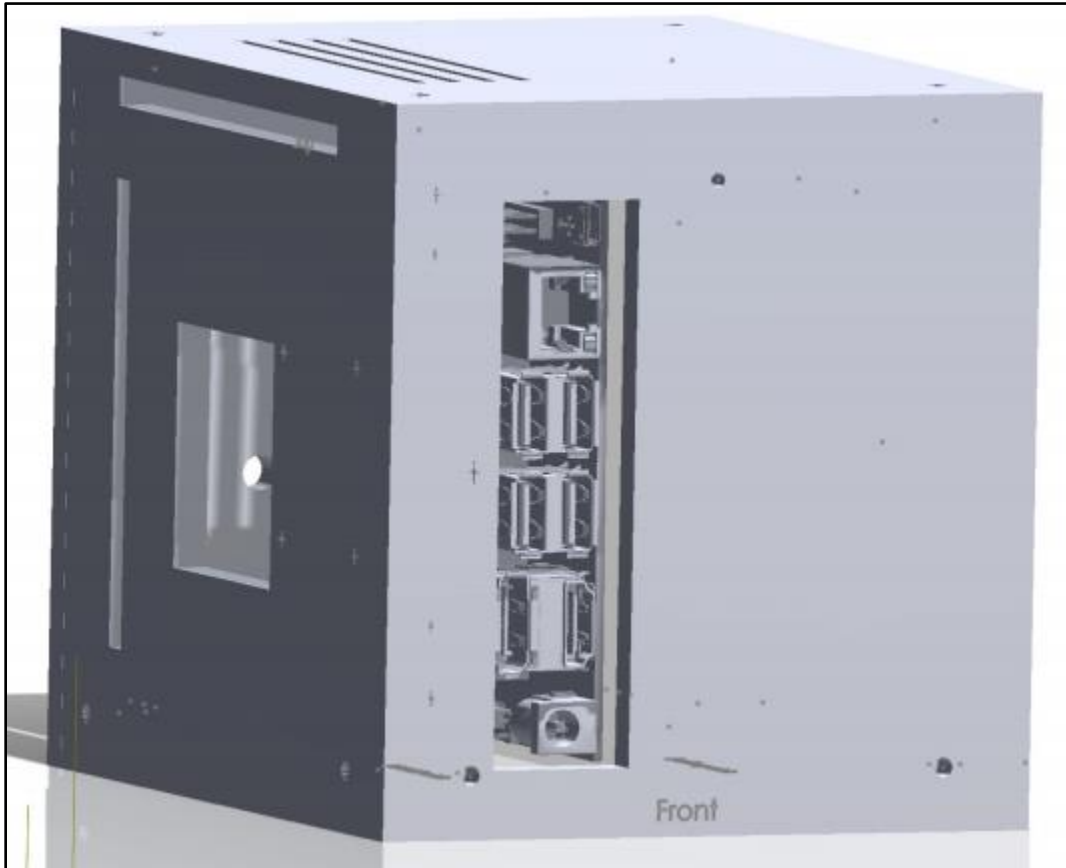


Figure D 1: Full enclosure [14]

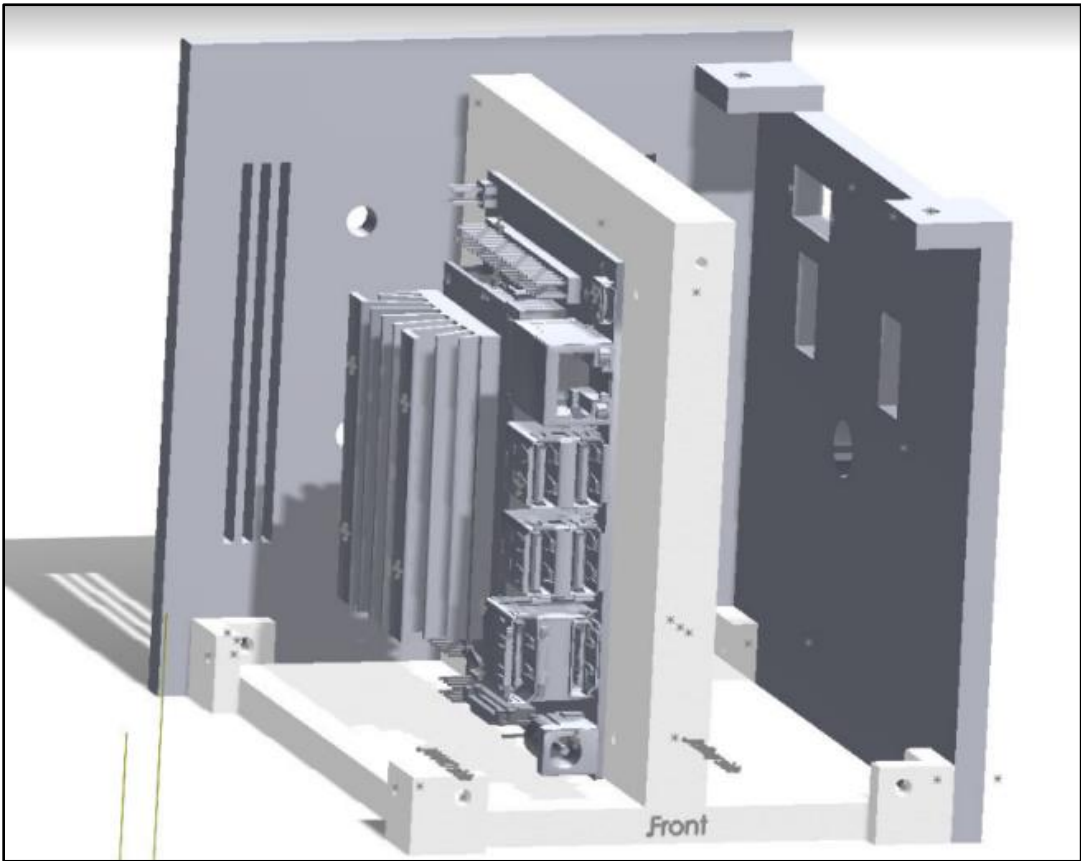


Figure D 2: Enclosure with Nano exposed [14]

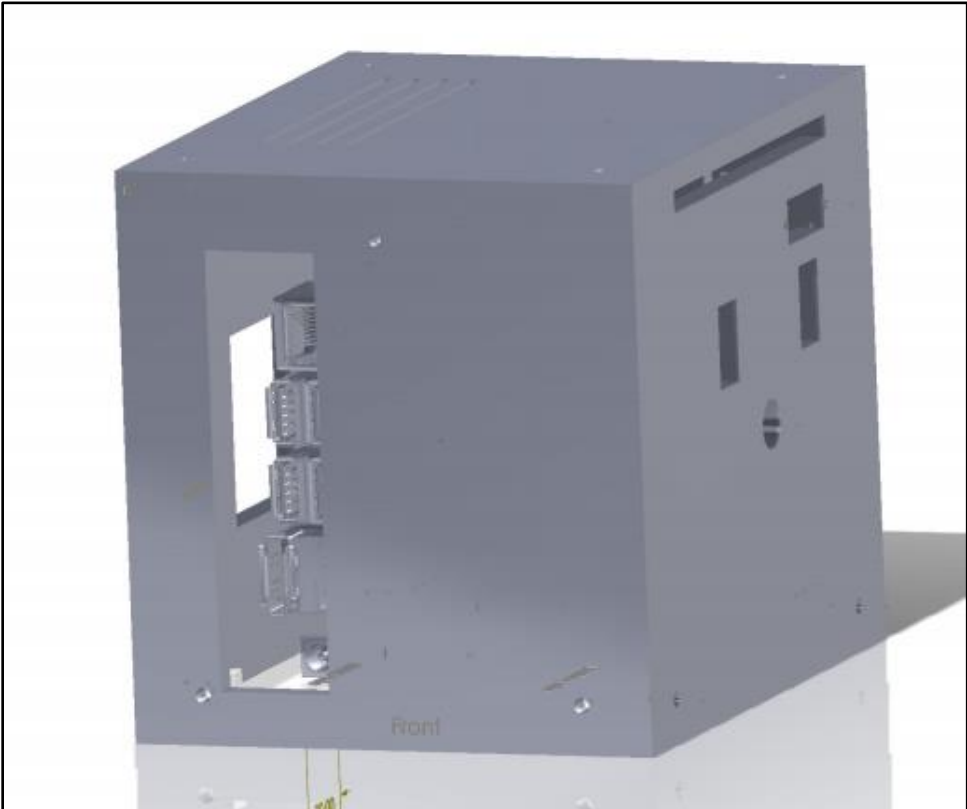


Figure D 3: Full enclosure profile view [14]

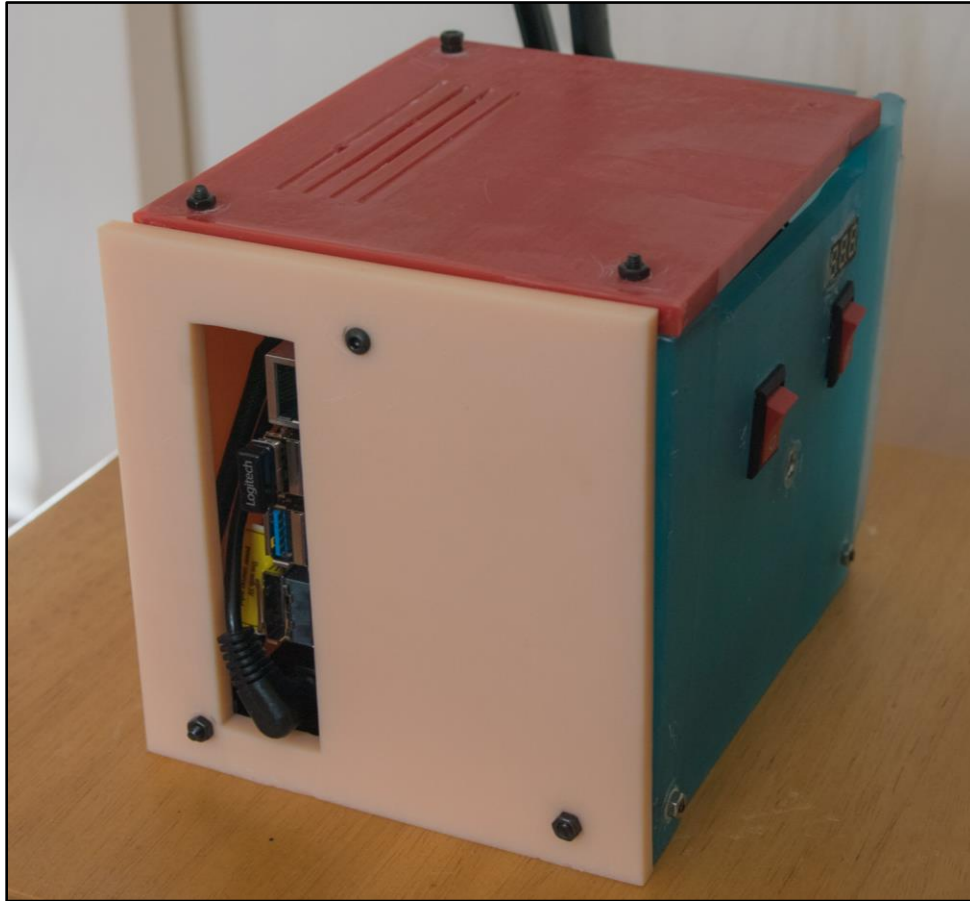


Figure D 4: Team 4 Full Printed Enclosure Design [14]

APPENDIX E. VENDOR CONTACTS

On behalf of Team D.c.M.V., thank you Professor Ghazan Khan, from the Civil Engineering Department at Sacramento State University. He provided invaluable insight on the direction our project should take. Dr. Khan was gracious enough to offer sponsoring the project, which we did not take up the offer, but we do appreciate his generosity and kindness.

APPENDIX F. RESUMES

Sergio Cortes

Objective

-Electrical and Electronics Engineering student determined to gain experience in his career field. Fluent in Spanish, both written and oral.

Education

- California State University, Sacramento
Bachelor of Science, Electrical Engineering
GPA: 2.95

*August 2015 - *Expected December 2020*

Relevant Coursework

Circuit Analysis
Computational Methods and Apps
Logic Design
Network Analysis
Applied Electromagnetics

Signals and Systems
Data Structures and Algorithms
Database Management Systems
Electronics I and II
Microprocessors

Feedback Systems
Power Systems
Senior Project Design I and II
Semiconductor Physics
Machine Vision

Work Experience

Shift Lead
• Manage a team of up to 7
• Manage a safe
• Communicate store status

Jamba Juice – Citrus Heights, CA

January 2017 – Present

Team Member
• Customer service
• Food preparation

Little Caesars - Orangevale, CA

January 2016 - December 2016

Project Experience

Wheeled Robot

- Programmed and built a wheeled robot that continuously added new sensors and tasks.
- Installed, soldered, and wired electrical components.
- Programmed sensors and motors.

August 2019 – December 2019

Seizure Warning Headband

- Designed a headband that would send an email whenever the person wearing it dipped below a certain angle.
- Programmed a Parallax Propeller board to send a signal to a Raspberry Pi 3 whenever an accelerometer reached a certain value.

February 2019 – March 2019

Data Collection of Motorized Vehicles

- Developing an inexpensive parking data collection system
- CAD design
- Building 18650 battery pack

August 2019 – present

Knowledge and Skills

Circuit Building

-Amplifiers, Diodes, Sensors, Wires, Breadboards

Computer Languages

- C, C++, Python

Software

- SolidWorks, Cadence, Matlab, PSPICE

Machines/Microcontrollers

- Arduino, Parallax Propeller, Raspberry Pi, Oscilloscope

Michael Khoo

Objective

Seeking to obtain a part-time, a full-time, or an internship position where I can use my engineering knowledge to do hardware benchmarking and design.

Education and Related Courses

Computer Engineering, California State University, Sacramento 2015-2020 (Graduated May 2020)

- **Advanced Computer Organization** (Fall 2018)
- **Operating System Pragmatics** (Fall 2019)
- **Computer Interfacing** (Spring 2018)
- **Senior Design II** (Spring 2020)
- **Electronics I** (Fall 2018)
- **CMOS and VLSI** (Spring 2020)

Computer Technical Skills

Software: Adobe CC, Altera, Arduino Platform, AutoDesk, Cadence Virtuoso, Multisim, Symantec Ghost, Vivado

Programming Languages: Assembly (Intel x86, MIPS, AT&T), C, C++, Java, Python, Verilog, VHDL

Operating Systems: OS X, PartedMagic, Raspbian, Ubuntu, UnRAID, Windows

IT Platforms: Altiris, Microsoft SharePoint, ServiceNow, SpiceWorks

Work Experience

ECS IT Consulting Assistant

September 2016 - Present

- Setup and deploy machines, software, and equipment for computer labs, servers, faculty and staff members.
- Troubleshooted desktop/laptop hardware components, network switches/routers, printers, monitors, TVs, docking stations, projectors.
- Promoted to Help Desk Coordinator (*Jan 2018 - Present*): organized/ran weekly work meetings, trained new help desk members, and updated manuals and training materials.
- Accomplishments: Migrated training materials to Canvas which is now the primary source of training for new help desk members, completed over 200 tickets on SpiceWorks.

Academic Awards

Dean's Honor Roll (2018-2019)

Project Experience

Pipelined Datapath

- Under the MIPS architecture, simulated a multi-cycle pipelined datapath in Verilog that can execute various types of instructions as well as handle hazards detect and exceptions.

Senior Design

- Created a machine vision device that collects parking data for professionals to create useful models for solving parking problems.

Java App using Scrum Process

- Created an electricity usage calculator app from the ground up with a Scrum team of five.

Extra Activities

Computer Technology Club (CTC)

- Club President (*January 2018 - present*). Took lead in running a student computer repair shop; taught members how to build, diagnose and repair desktop computers.
- Lead workshops for overclocking and benchmarking desktop hardware on both Intel and AMD.

Christian Students at Sac State

- Club President (*August 2019 - present*). Lead general meetings, organized activities and outings, and reached out to students via tabling and passing fliers.

Ryan Uda

OBJECTIVE

- To obtain a position related to the field of Computer Engineering or Computer Science.

EDUCATION

- California State University, Sacramento
Bachelor of Science, Computer Engineering

*August 2012 - *Expected May 2020*

RELEVANT COURSEWORK

Advanced Computer Organization	Computer Networks and Internets	Operating System Pragmatics
Advanced Logic Design	Data Structures and Algorithms	Signals & Systems
CMOS & VLSI	Database Management Systems	Senior Project Design I
Computer Hardware System Design	Electronics	
Computer Interfacing & Microprocessors	Operating System Principles	

WORK EXPERIENCE

- | | | |
|--------------------------|--|------------------------------|
| Legal Assistant | Law Office of Robert Masuda - Sacramento, CA | <i>July 2016 - Present</i> |
| Monorail Usher(Seasonal) | California State Fair - Sacramento, CA | <i>July 2013 - July 2016</i> |

PROJECT EXPERIENCE

- Pipelined Datapath** *February 2018 - May 2018*
 - Simulated a datapath with a control unit for a pipelined system in VHDL that can execute various types of instructions as well as handle hazards and exceptions.
 - Validated its functionality by debugging individual components through a stimulus test bench then combined through a top level module.
- Facial Recognition Security System** *October 2017 - December 2017*
 - Designed a security system that uses facial recognition as a means of authentication which will inform the user of unrecognized users by Email.
 - Programed a Raspberry Pi and Raspberry Pi Camera to take a photo of a user and determines whether or not the photo is of a recognized user.

KNOWLEDGE AND SKILLS

Computer Languages

- C, SQL, System Verilog, VHDL, Assembly x86

Software

- Windows, Mac OS, Linux/Unix, VMware, Cadence, Matlab, PSPICE, Multisim, Microsoft Office

Machines/Microcontrollers

- Arduino, Raspberry PI, Propeller, FPGA

Practical/Useful Experience

- While working on projects, developed ability to troubleshoot setbacks and errors through concise and methodical steps.
- Demonstrated ability to accomplish multiple tasks and projects with strict deadlines.
- Worked part-time while going to school as a full-time student.
- Familiar with conversing and corresponding with large bureaucratic entities in order to accomplish various tasks.
- Officiated recreational and competitive soccer games from freshman year of high school until freshman year of college.
- Experienced in customer service and working with abrasive clients.
- Volunteered as an assistant coach for a youth girls soccer team for the past 4 years.
- Volunteered as a coach at Basic Hoops Summer Basketball Camp for the past 10 years.
- Volunteered at the Nichiren Buddhist Church of Sacramento bazaar for the past 10 years.