SMART Rack

Andrew J. Jewers, EE, Alessy Leblanc, EE, Fedor N. Panikov, EE, Arthur E.Singas, CSE

Abstract— **One of the most common methods of transportation, especially for students, is biking. Not only is it environmentally friendly, but it is cheaper than owning a vehicle, and can be much more reliable than taking public transportation. However, a big issue, especially on campus, is figuring out where to leave a bicycle. Often times, when one walks around campus, a common sight to see is overcrowded bike racks. These bike racks fill up very quickly throughout the early day, especially in very busy areas. This leads to cyclists being unsure of where they will be able to lock their bikes. The SMART Rack offers a solution to this issue. The SMART Rack is a bike rack that is connected to the internet, and is operated through a mobile application. The app allows users to reserve spots at the rack of their choice, giving users peace of mind knowing that their bike is secure.**

I. Introduction

A. Significance

The US market for bicycles is very big, being valued at around six billion dollars, and around 15-20 million bikes are sold annually in the United States [1]. However, many densely populated areas in the United States are struggling to keep up with the growing market, meaning there are not enough spots for bikes to be locked. This is very evident on college campuses. There are many recorded issues on campuses of bike rack overcrowding, such as University of Texas, where there are only 7,000 recorded spaces for the 9,200 registered bikes [2]. This issue can even be seen on our own campus here at UMass Amherst, at places such as the library, where there are commonly near to no spaces available throughout the day. The lack of designated bike spots leads to several issues, one of them being bikes locked to undesignated areas, such as railings on handicap ramps, which of course causes accessibility issues.

B. Context and Existing Products

There are currently only a couple of major attempts at creating products that attempt to solve this issue. The most prominent company that attempts to solve this issue is BiKeep, a company based out of Estonia [3]. BiKeep sells bike racks which are also connected to the internet. Like our product, they also have an app that is connected to the product, but it does not have as many features as our app. The BiKeep app can help locate a bike rack on a map, as well as being able to unlock a rack when a user is at the location. However, a user cannot reserve a spot. This means that there is no confirmation that a user will have a bike spot available to them when they arrive at the rack. Additionally, if a user does not have their phone on them, they must purchase a specific BiKeep RFID card from the company. This means that a user can potentially be stranded if their phone dies and they do not have an RFID card. The BiKeep bike rack comes with a secure metal bar which will open and lock the bike straight to the rack, leaving no need for an external bike lock. They also have additional features, such as adding solar power to the system, however that is still just in conjunction with the AC power. This is also more expensive, with a single bike rack with no additional features coming in at over $1000.

The second major attempt at a smart bike rack is by a company called Vadebike [4]. This is also a “smart” bike rack, however it does not come with an app. Vadebike has a website on which users can find the locations of the bike racks, but there is no information on whether or not a spot is available, and just like BiKeep, there are no reservations available. It also incorporates the use of an RFID card, which is used to unlock and lock the bike on site. There are two chains per bike spot, one meant for the back tire and other for the front tire. There is also a small storage space meant for a helmet. Additionally, this service costs money, and users must pay per hour of use. Lastly, this service currently has a geographical limitation, as they are based in Barcelona, Spain, and do not have any locations outside of the Barcelona area.

C. Societal Impacts

The people that would be most impacted initially by our product would be UMass community members, as this would be first deployed on our college campus. Having our bike rack set up on campuses would be a huge bonus to students, especially students who live off-campus. Students would be more likely to bike to campus, because they would be guaranteed a bike spot. Furthermore, if there would be many of these racks set up around campus, students could be guaranteed a spot that is very close and convenient. Other than students, there are still many people that would benefit from this product. Essentially anyone who would like to use a bike as a mode of transportation would benefit from this product. It would not only make the logistics of a commute simpler, it also helps by coming with its own lock. Because there is a built-in lock at the bike rack, there is no longer a need to bring an external personal lock. This will save a commuter the hassle of carrying a potentially heavy lock around, and can even save the user money, because they will not be forced to carry a lock around. A great benefit of our product is that it will reduce the number of bikes locked up to non-designated areas. One of the more common places for a bike to be locked is a handicapped ramp. This is visible on our own campus, as ramps such as the ones in from of Hasbrouck and Engineering Lab almost always have bikes locked to them. Therefore, people with handicaps could benefit from the SMART Rack, as less bikes will block the handicap ramp pathways.

D. Requirements Analysis and Specifications

The combination of RFID car compatibility along with fast locking show that one of our main priorities for users is convenience. The locking time constraint is reflective of our choice of hardware, and a simple solenoid locking system will ensure fast locking/unlocking. Our system will be connected to the grid, so the rack will always be powered and available for users to access.

Requirement	Specification	Value
Security	Database encryption	All services supplied by Amazon Web Services (AWS) have a default encryption. The encryption is called "Encryption at rest", and our application will meet strict encryption compliance and regulatory requirements.
Convenience	RFID compatibility	8 digit student ID Unique ID for every student Allows for spot authentication
	Always powered	AC Power Source, 110V
	Grace Period	Must be present at desired spot within 15 minutes of authenticating your spot on the mobile application
Responsive	Locking Engagement Time	<3 sec

Table 1: Requirements and Specifications

II. Design

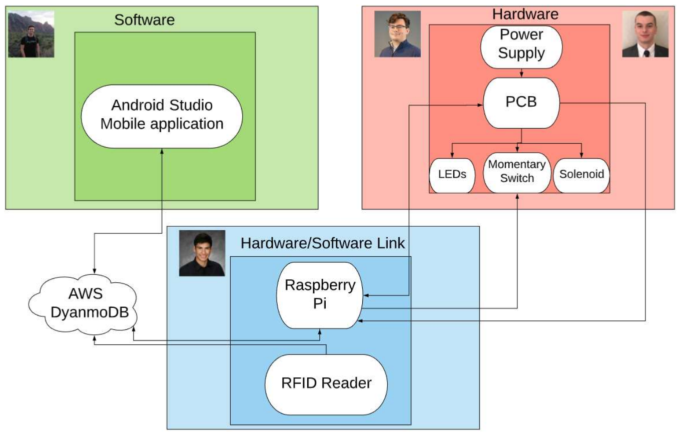

Figure 1: Block Diagram Overview of Design

A. Overview

Our solution to the issue of bicycle congestion is our SMART Rack. The SMART Rack features an Android mobile application, network connectivity, a Raspberry Pi, and an embedded chain locking system. This technology will solve the problem of bicycle congestion as it will allow users to choose an open parking spot with our mobile application ahead of time. The network connectivity will allow users to be able to reserve spots from the comfort of their own homes, so they do not have to worry about trying to get a spot once they arrive on campus. Another technology we had considered was using an ESP32 instead of a Raspberry Pi, which is talked about more in Appendix A. When attempting to meet all specifications with our design, one tradeoff we had to make was having an AC power source available in order for SMART Rack to operate. Ideally, this design could be adapted to use a renewable energy source such as solar, however this would require some improvements to the code for the Raspberry, such as entering a low power mode when cards are not being scanned by users.

B. Block 1

The SMART Rack's backend system will be able to manipulate incoming user data provided by the android application and will further determine whether a user within the local database is authorized to use a specific slot within the SMART Rack. Currently, the program (written in Python) is running on a Raspberry Pi 3 B+[5]. This device allows for smooth communication between the program and the android application (which is being developed by Arthur). The Raspberry Pi will act as the intersection between the entire system's components. It will link the Android application to the PCB and hardware components (such as the solenoid) and will be able to report sensory data back to the application. Currently, the program is able to recognize whether a user within our local database is authorized to use a single slot. This means that the program has not been scaled to the point where it can differentiate between different slots on the rack.

The local database is a CSV file that is continuously updated through the Android application as well as the Python program. Below I outline the different scenarios in which the program will recognize and respond accordingly:

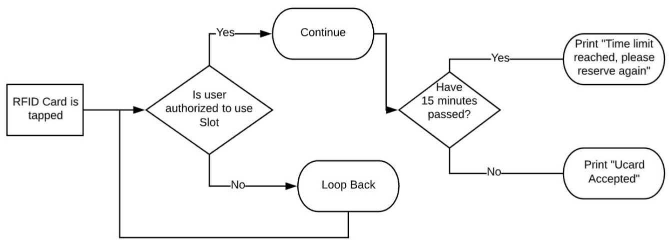

Figure 2: User ID Control Diagram. How our backend software manipulates user data

C. Block 2

The solenoid and supporting circuit will be responsible for locking bikes into the rack. The solenoid operates on 12VDC, making it the highest voltage component for the whole system. Because of this, and the large amount of current the solenoid can draw, power consumption is important to keep in mind when designing how the solenoid will operate. To design this circuit, basic principles of DC circuits from Circuits I and II will be applied, in addition to DC analysis of transistors from Electronics I. A comprehensive understanding of solenoids will be necessary to build this block. Solenoids can be thought of as an inductive element, since they are essentially an electromagnet used to push or pull in a metal plunger. It is important to know how a large inductive load will behave when applying a significant voltage across it, as there are some important details relating to this process such as how current is able to pass through this load.[6] Figure 3 shows a schematic of the solenoid circuit, with the solenoid being represented as an inductor in series with a resistor. When the BJT first enters saturation, almost all of Vcc is across the solenoid, which pulls in the plunger due to the high voltage. As the capacitor begins charging, the voltage across the solenoid lowers to a suitable hold-in voltage, where the voltage is just high enough that the solenoid with stay pulled in, but low enough such that it will not heat up and dissipate excess power. To test this system, a simple external voltage was applied to the output. This was eventually switched out for an output pin on the Raspberry Pi's GPIO. See Appendix B for more information on this testing.

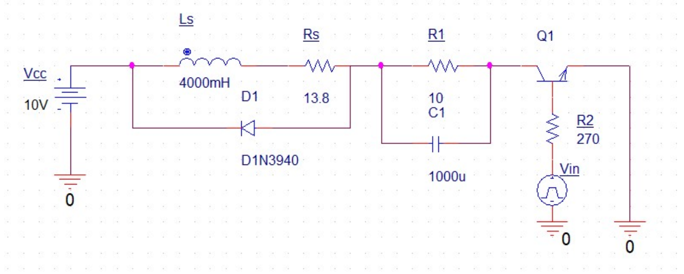

Figure 3: The solenoid driving circuit. The input voltage represents the voltage level sent from the Raspberry Pi that controls what position the solenoid is in.

In addition to the solenoid driving circuit, there is also an LED Lock Status Circuit that goes along with it, which can be seen in Figure 4. It serves two purposes, providing the user with visible feedback in the form of an LED on whether or not their lock is in place to be locked, and at the same time providing that information back to the Raspberry Pi. It takes in a high or low input from both the Raspberry Pi and the momentary switch. The circuit is set up with two NPN transistors in series, with the Emitter of one connected to the collector of the other, The top transistor receives 5VDC to its collector. This circuit is an AND gate, with the output at the emitter end of the bottom transistor. The output will only be high when both of the inputs are high, sending a voltage of approximately 3.3V. The transistors will go into Saturation in this case, and the LED will be lit up, and a high signal will be sent back to the Raspberry Pi.

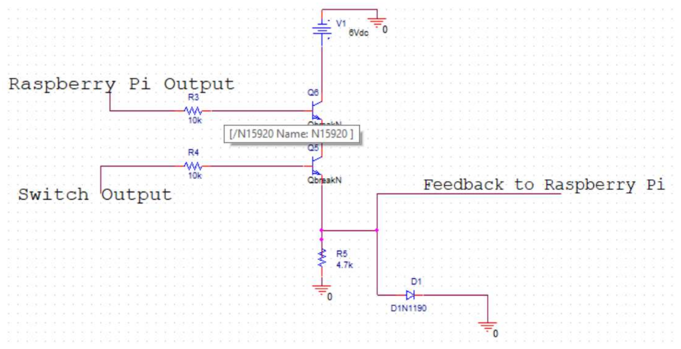

Figure 4: LED Lock Status Driving Circuit

III. The Product

A. Overview

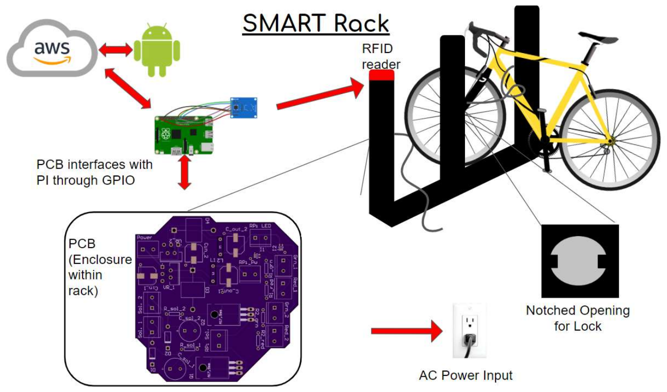
SMART Rack
aws
RFID reader
PCB interfaces with PI through GPIO
PCB (Enclosure within rack)
Notched Opening for Lock
AC Power Input

Figure 5: Product Sketch

Our SMART Rack product sketch has gone through a plethora of changes throughout the semester. Ultimately, we landed on what was most efficient and most suitable for our end users. In the top right corner of our product sketch, we have a simplified version of what our SMART Rack looks like. The way we designed our SMART Rack allows for one pole to be used for two slots on the rack. Our android application is used for users to reserve their slot on the Rack. All user data goes through DynamoDB, a database service ran by Amazon Web Services (AWS). This information is then used to determine authentication using a Python script held on a Raspberry Pi 3 B+. The raspberry pi is also used to control the solenoid for locking/unlocking purposes. All of this information travels back to the android application in order to inform the user that their bike was successfully locked using the embedded locking system.

B. Electronic Hardware Component

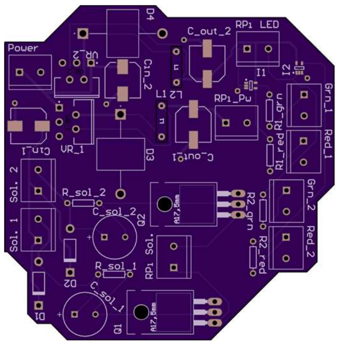

Figure 6: Final Fabricated PCB Design

The design of the principal electronic hardware component went through many different variations throughout the year. Initially, the Power Regulation Circuit was going to be a linear voltage regulator. After our meetings with advisors and evaluators, we updated our design, to be more power efficient. Our final design used a step down buck voltage regulator, as it has a much greater power efficiency, at approximately 80% versus a linear regulator which would have been approximately 40% efficient. The LM2596 Step-Down Voltage Regulator was used, alongside a 12V 10A AC/DC Adapter. The output of the regulation circuit is 5V 3A, which is enough to power both the Raspberry Pi and the LED Circuit.

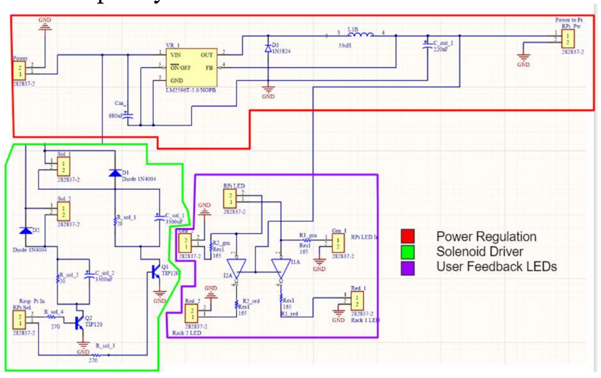

Figure 7: Electronic Hardware Schematic

Another part of the design that went through several iterations was the LED User Feedback Circuit. In its final design, it consisted of an two inverters instead of transistors, as it consumed less power, was much less complex, and had much quicker feedback. The LEDs turn on different colors depending on the status of the lock. The blue LED will light up if the spot is locked, and white if the spot is unlocked.

In the solenoid circuit, the general topology remained the same, however the resistor values changed to optimize the solenoid driver.

Unfortunately, due to COVID-19, the fabrication of the PCB was delayed, and we did not receive it until the last day of classes, and were not able to populate it before we had to leave campus. However, as you can see in Figure 6, we have a finalized, fabricated, but not populated PCB. In order to present our demo during CDR, we had a fully working prototype, made of our hand soldered protoboard.

C. Functionality

Using *Figure 1*, only one part of our block diagram was not functioning as we simply did not have that part yet (the PCB). Although ordered on time, due to COVID-19 complications, we were not able to test our PCB. In order to still have a functioning product, we used protoboards and soldered our own components that were going to be originally done remotely. We had multiple complications with the DynamoDB Database as I had to use JSON to receive and manipulate user data. Another complication we encountered was false triggering of certain GPIO pins that would result in both our solenoid and LEDSs malfunctioning. In order to fix those errors, we added a line of code (GPIO.cleanup()) that does exactly as the name signifies, it cleaned and reset any set GPIO pins once the script finished running for one users RFID tag.

D. Performance

The first specification that we listed was database encryption in order to provide security of our user's data. AWS provides "Encryption-at-rest" services for all tables, which means that Amazon owns the key to decrypt data. The second specification we listed was RFID compatibility in order to provide convenience for our users. This allows everyone to identify themselves to our system with their UMass student ID. We did this by programming one of our teammate's student ID numbers to a RFID test card. We were successfully able to lock and unlock a bike with the associated RFID card, all while leaving another bike locked on the rack with a different ID number associated with it. The third specification that was listed was “always powered”. We were not able to test this specification due to COVID-19. Our plan was to leave the rack running over a long period of time to see if it could still service requests consistently, but we never had the chance to do so. The fourth specification that we listed was the grace period. This specification has been renamed to “15-minute post-reservation window”. We were able to meet this specification by making use of a service called Time-To-Live (TTL) in AWS. TTL is a service that allows you to specify a time that an entry in a table should be deleted. We set the TTL to 15 minutes, and were able to show that the entry was deleted from the table. The fifth and final specification that we listed was a sub-3 second lock engagement time. At the time, our implementation of the lock used a linear servo. Now, we are using a solenoid that locks and unlocks on the order of milliseconds.

IV. Conclusion

Since our MDR, we have developed a functional product, with an updated enclosure. Alessy and Arthur implemented Amazon Web Services to add superior security and reliability to the communication between the Raspberry Pi and the Android mobile application. Fedor and Andrew were able to optimize and finalize their hardware design, and design a custom PCB. Due to the COVID-19 complications, the only major work left unfinished was the population of the PCB.

However, our group was able to demonstrate SMART Rack using hand-soldered protoboards for our CDR.

ACKNOWLEDGMENT

We would like to thank our advisor, Professor Xu, and our evaluators, Professor Soules and Professor Xia, for the continuous support and advice on building a successful senior design project. We would also like to thank Alden Michaels for taking time to speak with us and give us advice on our project.

REFERENCES

[1] Wagner, I. (2019). *Topic: Bicycle Industry in the U.S.*. [online] www.statista.com. Available at: https://www.statista.com/topics/1448/bicycle-industry-in-the-us/ [Accessed 13 Dec. 2019].
[2] Chu, B. (2019). *UT needs more bike parking, students say - The Daily Texan*. [online] Dailytexanonline.com. Available at: http://dailytexanonline.com/2019/04/24/ut-needs-more-bike-parking-students-say [Accessed 13 Dec. 2019].
[3] "Commercial Bike Racks & Bicycle Parking Systems: Manufacturer," Bikeep. [Online]. Available: https://bikeep.com/. [Accessed: 19-Dec-2019].
[4] Vadebikebcn.com. (2019). *Smart biking. Cycling & parking*. [online] Available at: https://www.vadebikebcn.com/en [Accessed 13 Dec. 2019].
[5] Meyer, Robinson. "The Unbelievable Power of Amazon Web Services." *The Atlantic*, Atlantic Media Company, 23 Apr. 2015.
[6] P. Millett, "What's the Best Way to Drive a Solenoid?," *ElectronicDesign*, 31-May-2018. [Online]. Available: https://www.electronicdesign.com/industrial-automation/article/21806574/whats-the-best-way-to-drive-a-solenoid. [Accessed: 19-Dec-2019].
[7] Pimylifeup.com. (2020). *How to setup a Raspberry Pi RFID RC522 Chip*. [online] Available at: https://pimylifeup.com/raspberry-pi-rfid-rc522/

APPENDIX

A. Design Alternatives

One of the design alternatives that we considered was using an ESP32, a microcontroller, instead of the Raspberry Pi that we are currently using. The ESP32 has the ability to connect to the internet, which is one of the reasons why we had considered it. However, the main reason we considered it versus the Raspberry Pi is because of power consumption. The Raspberry Pi has significantly more power consumption than the ESP32. The ESP32's power consumption is very low, coming in at under 300mA, at 3V. On the other hand, the Raspberry Pi always needs to be powered on, and it takes in 5V at 2.5A, which means the power consumption is around at least 12.5W. However, we concluded that the Raspberry Pi was the better option. On one hand, we are using an AC wall outlet to supply power to our device, therefore a Raspberry Pi's power consumption is very minimal compared to supply. But on a more important note, it was a lot less feasible to use the ESP32. If we used the ESP32, it would be almost impossible to connect to the UMass Wi-Fi.

Originally the locking mechanism we chose was a linear actuator, which would be supplied a voltage to extend through a hole in a locking pin to lock bikes into place. This seemed like a desirable choice due to its ease of use, low power consumption, and resistance reading. This resistance reading would indicate the position of the actuator, allowing us to read this resistance to determine information such as if the lock engaged properly and which spots are available on a given rack. The disadvantages of the actuator were its cost and slow locking speed. An actuator takes time to move into place, which is determined by how it is geared internally. Best case the actuator could move into position in approximately a second, but that would require the user to hold the lock in place that whole time to ensure the lock closes properly. A solenoid provides a more instant movement of the locking pin, and the solenoid proved to be more durable than an actuator. While the solenoid does not have the resistance reading to indicate its position, various inexpensive sensors exist to accomplish this, which is a goal for Fedor and myself for CDR. Solenoids are generally much cheaper than linear actuators and far less complex. These factors are an added benefit for choosing the solenoid which makes the rack easier and less expensive to maintain long-term.

Another design we considered was having a metal bar to lock over the bike instead of a chain lock. The main advantage of this would be improved security. However, there were several problems with this. One main issue was that a company in Barcelona has already developed a product that has a bar lock and designing a completely new and improved system with a bar lock seemed too mechanically complex. Another disadvantage of a solid bar is that it would not be a universal design. Surely it could be made to accommodate most bicycles, but exceptionally small or large bikes, or oddly shaped bikes would not be able to be locked in our rack if we designed it with this type of lock. These factors lead to our decision to implement a chain lock, which would be simple to design a locking mechanism for, and universal for all types of bikes.

B. Technical Standards

Wi-Fi – IEEE 802.11 details protocols for the implementation of wireless local networks for computers. Wi-Fi is perhaps one of the world's most widely used standard and is incorporated to almost all of the electronics we use in modern times. Our product utilizes the Wi-Fi standard for communication between the Raspberry Pi and Amazon Web Services. Although not part of the final design, early stages did feature direct, wired LAN connection for testing purposes, widely covered under IEEE standard 802.

RFID – Both the International Organization for Standardization(ISO) and the International Electrotechnical Commission(IEC) have standards for RFID transmissions. Low and high frequency RFID tags do not require any licensing for worldwide use, however, ultra-high frequency tags are not globally standardized. Our RFID card reader operates at 13.56MHz, which is defined as high frequency, so there would not be any standardization issues if SMART Rack were to be used in other countries.

C. Testing Methods

Solenoid/Momentary Switch Integration with Raspberry Pi:
After the driver for the solenoid was completed, we began testing our system by connecting the base of the NPN

transistor to one of the general purpose input/output(GPIO) pins of the Raspberry Pi. This would allow for the Pi to supply a 3.3V high signal to the transistor, which would allow us to operate the solenoid through the code on the Pi. Alessy and I configured the Pi so that the solenoid would be sent a signal to retract as soon as an RFID card is tapped on the card reader. This continuous high voltage on the output pin would keep the solenoid retracted as long as we wanted, which would allow the user to insert the lock piece to lock into the solenoid's plunger. Now we needed a way to control turning off the signal to the solenoid so it would know when to close. The solution was a momentary switch that would be activated when the lock piece was inserted and aligned with the solenoid plunger. We connected the switch between another GPIO pin and ground. This pin was configured as an input, and we used the internal pull-up resistors to operate the switch. We then coded the Pi such that grounding the input pin with the switch would cause the Pi to stop sending the solenoid circuit a high level and thus extend the solenoid again and lock the rack. After coding the inputs and outputs required, we first began by scanning an RFID card and reading the voltage at the output pin. When we first saw no voltage being supplied, we knew we had to make some corrections in the code to get the behavior we wanted for the output pin. After the output pin was supplying a voltage, we then tested to see if the switch input could control the output pin. The same voltage was read at the output pin to determine if pressing the switch was turning off the high on that pin. Once the input and output were working as expected, the solenoid could be controlled via the output voltage and switch. After the solenoid and switch were mounted into the test enclosure, we tested the operation of the solenoid with the 3D printed lock to ensure that the lock would consistently and easily operate for users.

D. Team Organization

Our team is composed of four members working on different tasks for our project. Alessy and Arthur are working closely together, with Alessy working on RFID and Database development, and Arthur working on developing our user-friendly Android application, which users will use to reserve their bike slots. Andrew and Fedor will also be working closely together on more of the hardware side, with Andrew working on the hardware development such as the solenoid, and Fedor working on the PCB design. Even though those two groups, split essentially by software and hardware, work closely in their own groups, there is still good communication between the two groups, which allows for everyone to learn from each other. Team communication started off slow during the beginning of the semester, due to some extenuating circumstances. One of our team members was a part-time student for the Fall semester, as they were taking part in a co-op in Boston. This meant that they could only be on campus from Friday-midday on Sunday, meaning that it was difficult to set up many meeting times. However, with the help of Professor Xu, we were able to set up strict weekly goals for each team member, and had weekly meetings with our advisor, and the team communication and productivity increased greatly.

E. Beyond the Classroom

Alessy Leblanc - My duty for this project is to implement a robust backend system that will be able to manipulate and respond to all user data being fed. In order for me to have accomplished what we have developed so far, I had to develop certain Python skills, as well as a basic understanding of how the Raspberry Pi works. Personally, I believe that Python is quite similar to Java, and since I do have more experience utilizing Java in and out of academic courses, the transition from one programming language to another was not problematic at all, and in fact quite fun. I was successful in implementing an AWS service that took all of our user data and stored it remotely on an AWS server. We utilized DynamoDB, a service that allows us to get user data from the AWS server and manipulate it to authentic users on our SMART Rack. This allowed more space on the Raspberry Pi and consequently allowed users to interact with the app securely without having to worry about whether their data is kept safe and private. The DynamoDB service has an internal option for encryption of any raw data coming in and automatically de-encrypt it going out. A very valuable skill set that was briefly introduced to me was how the solenoid interacts to the Raspberry Pi using GPIO connections.

Andrew Jewers - One of the main skills I needed to learn for this project was 3D modeling and printing. I had briefly used AutoCAD software in the past, but had never designed and printed my own custom parts. Not only did I have to learn how to design 3D parts, but also the limitations of 3D printing, and how different printing parameters can be altered to optimize printing performance. I gained much of my knowledge on printing through M5 and Du Bois Library staff, who all gave me advice on my part and how I should design it for my application. I decided to work in Autodesk's Inventor to create my .stl file for printing, since it was easy to download, free for students, and at least somewhat familiar to me. I also learned how to match the dimensions of a mesh model file with the printer to ensure that the part is printed correctly in the proper scale. I found this portion of the project an enjoyable way to gain some interdisciplinary engineering experience. In terms of electronics knowledge, I had to learn about the different types of solenoids and how to drive them. Solenoids come in a myriad of different designs for various applications, so I had to learn about the differences and pick the best design for our application. I knew that we would want something that would be able to withstand lots of use and be as robust as possible, and I also knew that we would likely want a solenoid that is normally extended out when no voltage is applied, since the locked position is likely the position the lock would be in for almost all of its time. Since the metal plunger is a fairly heavy-duty piece of metal, I knew the drawback of this would be that it would take significant power to pull it in. So, for designing my driver, I knew that I would have to consider power consumption as one of the main design parameters. There is endless information on solenoid drivers on the internet, so I took many different designs and approaches into consideration and assessed the advantages and disadvantages of each. Solenoids are an extremely versatile component, so I was eager to gain experience with them as I saw this as a valuable skill to have as a professional for when I am involved with more projects in the future.

Fedor Panikov - There were several skills that I needed to learn or develop further in order to move forward with this project. One of those skills was circuit design. In our Circuit and Electronics classes, we did do some circuit design, but we were given many constraints by our professors, or lead the process in some sense by instructions. However, for our project, I had to come up with circuitry by myself, with my own set of constraints and guidelines. I spent a good amount of time researching the most important things that I need to consider when designing circuitry, which some had been taken for granted before, when much of the information had been given to me in class. As well as circuit design, another skill I have learned is PCB design. Aside from our freshman year class, in which we learned some basics of Altium, we did not do much PCB design. However, as this is my role, I must learn it. I spent a good amount of time learning the basics of PCB design and trying to find my way around several programs. YouTube has been a great resource, as well as Alden, who we talked to several times as a group. M5 provided several workshops for Altium Designer, which came in very handy, as well as several helpful members of M5. There are many connections between this skill and my life as a professional, as I actually helped develop a PCB at my co-op position, where I developed a PCB shield to go on top of an Arduino, that would perform some specific calculations based on an input from a thermistor.

Arthur Singas - The skills necessary to begin working on the server/client implementation are skills I have learned from previous programming courses, specifically ECE 374 Computer Networks & The Internet. Learning how sockets work and how to transmit data through them was something we briefly covered in that course, so it only took a little bit of review to implement. The main thing I spent time learning that hasn't been covered in a course is using Android Studio and Android Development in general. Android Studio offers so much that it felt almost impossible to start writing code. It took a few different YouTube videos for me to know where to look to implement a basic UI and how to program different elements within the UI to perform specific actions. The good thing about Android Studio is that there is so much documentation for anything that I needed to implement. I ended up using Java documentation for references throughout the entire project, and was able to achieve everything I needed to by CDR.

F. Budget

Below is a list of items that have been purchased for our project. Most of the expenses since our MDR have involved developing our hardware, such as our new enclosure, PCB, and components to populate the PCB.

Adafruit HUZZAH ESP-32	21.95
Raspberry Pi kit	69.99
2pcs Solenoid	36.74
RFID Reader and Tag (x2)	17.97
HDMI Cable	6.99
DP to HDMI Cable	8.59
32GB SanDisk microSD	7.64
Raspberry Pi	35.00
Pi Power Supply	8.49
Breadboard Wires	5.79
Power Supply	26.99
Infrared Obstacle	7.99
Mouser Inductor (x2)	3.74
Digi-Key Voltage Reg (x2)	9.54
20pcs Diode Schottky (x2)	19.98
PCB (w/ $25 expedited shipping)	75.00
Digi-Key TE AMP Connectors (x35)	19.25
Mouser Capacitor Nichicon 220uf (x10)	4.40
Mouser Capacitor Nichicon 680uf (x10)	7.00
Mouser Texas Instru. Inverter (x10)	1.80
Stainless Steel Oval Locking Clip	17.39
Raspberry Pi Transparent Case (x2)	12.98
Low Carbon Steel Chain 25'	20.85
Left Angled Micro-USB	5.99
Total	452.05

Table 2: Project Budget