University of Mississippi

Department of Electrical and
Computer Engineering
Oxford, Mississippi, USA

Cp E 462 Final Project Report

SharkCycles - Shark Park

Smart Parking Station for SharkCycles

Submitted by

Team Proton
Vincent Bruno,
Mason Glenn,
Canaan Lockridge,
Ramon Tillman

on

May 4, 2022

List of Acronyms

GPIO	General-Purpose Input/Output. 4
LED	Light-Emitting Diode. 6
mA	Milliamp. 3
MCU	Micro Controller Unit. 11
PCB	Printed Circuit Board. 3
RFID	Radio-Frequency Identification. 2–4
RPi	Raspberry Pi. 9
RX	Receiving. 4
SPI	Serial Peripheral Interface. 4
TX	Transmitting. 4
USB	Universal Serial Bus. 4

1 Project Overview

SharkCycles is a larger project being developed by many teams of undergraduate seniors. The goal of this project is to bring to the University of Mississippi campus an E-bike sharing system in hopes of reducing on-campus traffic and shorten long, cross-campus walks for students, faculty, etc. The hilly nature of the University campus makes conventional bikes less appealing to users and drives up the amount of vehicle traffic between classes. Our client, the University of Mississippi Department of Parking and Transportation, believes that this network of E-bikes could solve this issue by reducing the work required to traverse the campus. With that as an overall goal, multiple teams across multiple engineering disciplines are working to bring the SharkCycles network to life and satisfy the client's needs.

2 Team Proton - Project Goals

The goal of Team Proton over the 2021-2022 school year was to design and develop a smart parking station for the SharkCycles. After multiple meetings with Sam Patterson, the Director of Parking and Transportation of the University of Mississippi, we had the basic requirement for our final design. The basic design specifications for the parking station (in no particular order) are the following:

1. Energy Independent
2. Orderly
3. Scalable
4. Easy to Use
5. Secure

With those requirements in mind, Team Proton could get to work on designing a smart parking station for the SharkCycles.

3 Design Phase

We began our design by splitting off into two teams so that we could research different methods for this parking station. Keeping the project specifications in mind and wanting to build off of the SharkCycles from last year, we wanted our designs to be energy independent, orderly, scalable, easy to use, and secure. The designs we came up with originally had their perks, but they also had drawbacks and needed to be weighed heavily before deciding on what route to take. The two designs that we ended up picking between were the geofence and the puck to puck. The geofence was a nice way to easily check off the scalable, easy to use checkmarks that we were looking for, but as we were researching we found that there was no good way to keep it orderly or secure. That led us to our second option which was the puck to puck design. This design checked off the secure, orderly and easy to use checkmarks as well, but came with the task of how to scale it. In the end we ultimately decided to go with the puck to puck design as it hit all of the crucial parts we needed and we could find a way to scale it later.

3.1 Geofencing

The idea of a geofence is as the name implies. A geofence creates a virtual "fence" that connects a geographical region of concern. For our case, we were thinking of uses this as an area in which a person could park the SharkCycle. This posed questions like, what would this look like if there was an excess of bikes or if a person could just come and drop of the bike in any kind of way within the geofence. Also, the question of how would we receive and possibly send information pertaining to a specific bike. Sam Patterson made it very clear that he wanted the parking station to have some factor of structure. This was a viable option, but there was no straightforward answer to these questions, so we kept looking for other directions to go.

3.2 "Puck to Puck"

The puck to puck design as we initially called it actually consisted of an Radio-Frequency Identification (RFID) and tag reader. This process entails digital data which is encoded within a tag. When this tag is placed near a reader, the reader captures the encoded data. The data capture can be used to specify the user and initiate other functions for a given sequence. There are two primary forms of RFID technology consisting a passive and active communication. Of course here, we wanted to be able to send information of the bikes position to allow users to check out and turn in the e-bike. This methodology led to a more user friendly way to authorize, send, and receive information. It would also allow for a more structured parking area. For our case, we used this RFID technology to allow a user to open and close a lock once the bike was checked in or out.

4 Work Accomplished

The goal over the 2021-2022 school year was to create a prototype of a smart parking station as a continuation of the SharkCycles project. Team Proton managed to set goals and complete them each week to track progress as our prototype became more fleshed out. The fall semester we managed to get the Raspberry Pi to communicate with our RFID reader. This was one of our main goals because we needed the RFID to act as our authenticator to lock or unlock a station when a bike is being used. Our next major goal was disassembling the LINKA lock to use as our own lock in the prototype. We disassembled it and began running tests on it to figure out what the voltage and current draw would be for the lock and ultimately found out that it began operating a one volt. The ideal voltage was about three volts with a current draw of about eighty Milliamp (mA). The final work that we did for the fall semester was creating a KICAD schematic of our modified lock. This was done as way to try and piece together what the original printed circuit board looked like for the LINKA lock and create our own. We ultimately failed in creating our own though because most of the components of the LINKA lock's Printed Circuit Board (PCB) had been scratched off. That concluded our work in the fall semester and left a lot to be done in our spring semester. The spring semester started off strong though and we managed to get an H-bridge to regulate the current flow to the motor of the lock. This allowed us to begin sending RFID signals to turn the motor clockwise or counterclockwise depending of if we wanted it opened or close. The next goal we set out to achieve was to make the parking station completely self sufficient. We decided to go with solar power since the current draw was not very high and the sun was out most of the year. After the solar panel we needed a solar charge controller so that we did not fry our Raspberry Pi by supplying it with to much voltage. The solar charge controller was connected to a battery that powered our Raspberry Pi and our lock. The battery would be charged by the solar power from the solar panel and ultimately be self sufficient. We then needed to get a few adaptors to connect everything together. After all of our adaptors came in we began to piece our final prototype together. Our final prototype consisted of the Raspberry Pi acting as our brain and controlling the lock. The lock and Raspberry Pi were connected to the H-Bridge to send signals back and forth to open and close the lock. The Raspberry Pi, the battery, and the solar panel were connected to the solar charge controller. The battery was simultaneously storing power that it was receiving from the solar panel while also supplying power to the Raspberry Pi. After we got it all pieced together, we had to work iron out some bugs in our code. Towards the end of the semester we began to looking at using an Arduino between the lock and Raspberry Pi using serial communication. This would allow us to create a larger scale project by having one Raspberry Pi connected to multiple locks via the Arduino's. We managed to get it slightly working using a Universal Serial Bus (USB) connection, but shifted our focus to using the Transmitting (TX) and Receiving (RX) pins on the Arduino. This was done so that we could use the USB for debugging purposes but ultimately ran out of time in the middle of implementing it.

5 System Composition

The final working product consists of a Raspberry Pi 4 as the main driver behind the majority of the components. This Pi is powered by a battery through a solar charge controller. The battery is charged by a 12V solar panel. The Pi reads a RFID input from the RFID MRC-522 module connected via Raspberry Pi Serial Peripheral Interface (SPI) General-Purpose Input/Output (GPIO) pins. After reading this input, the Raspberry Pi sends a digital signal to either open or close the locking mechanism on the parking station to either park or release a theoretical bike. This signal is received by our H-bridge/motor controller which then sends a digital high to either to either open or close the motor. A final diagram of the design is show in Figure 1.

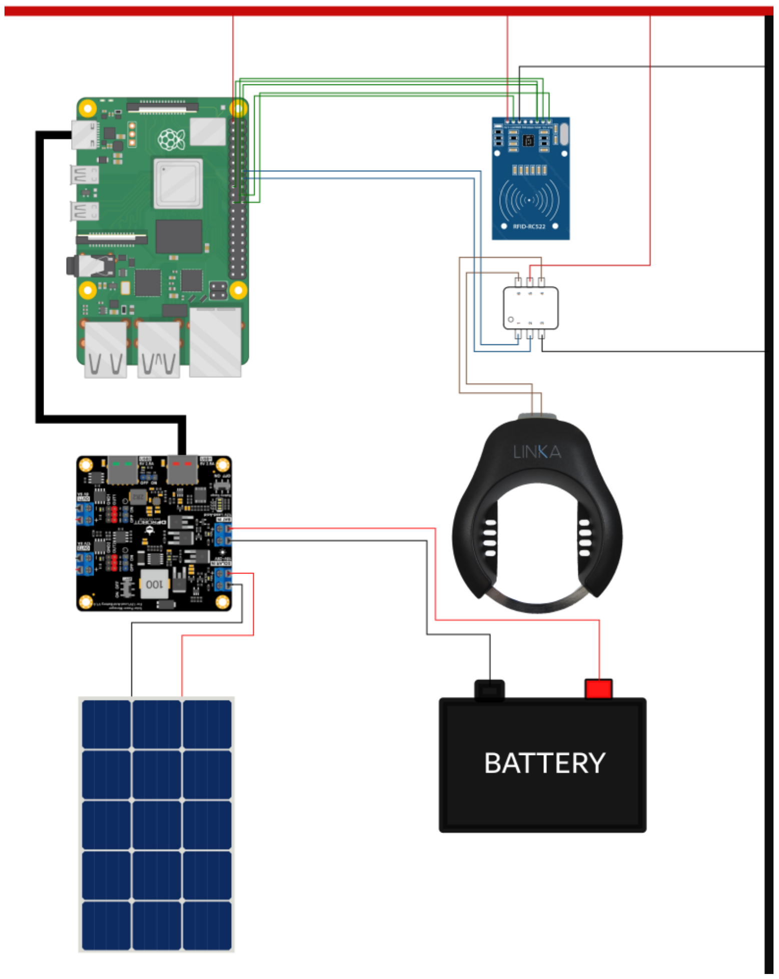
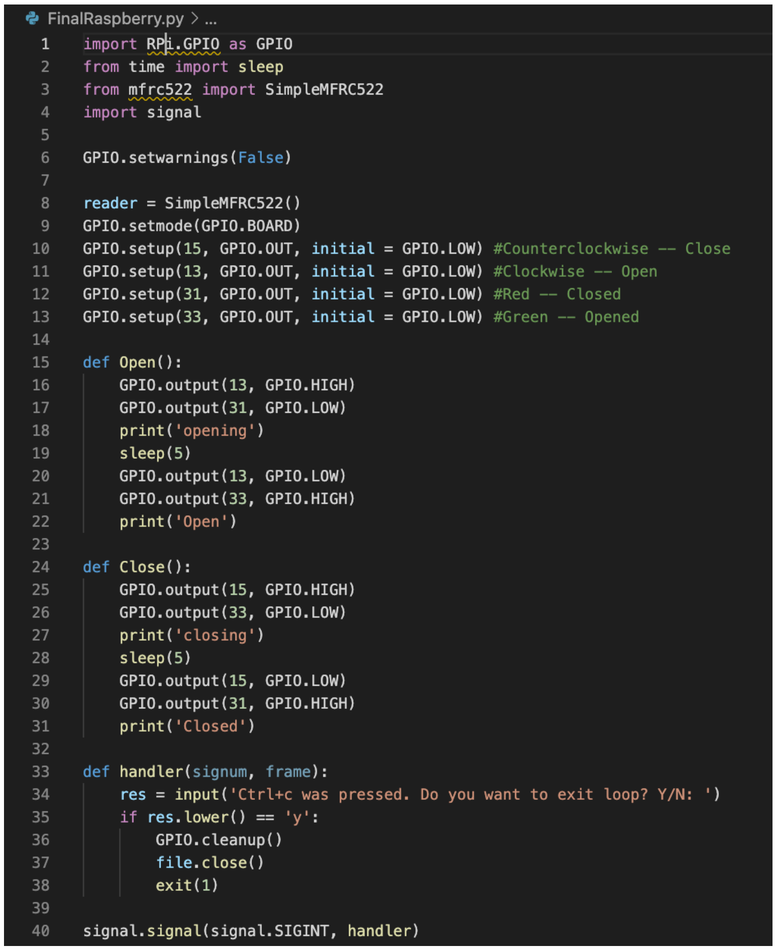


Figure 1: Final System Level Schematic

6 Control Software Explanation

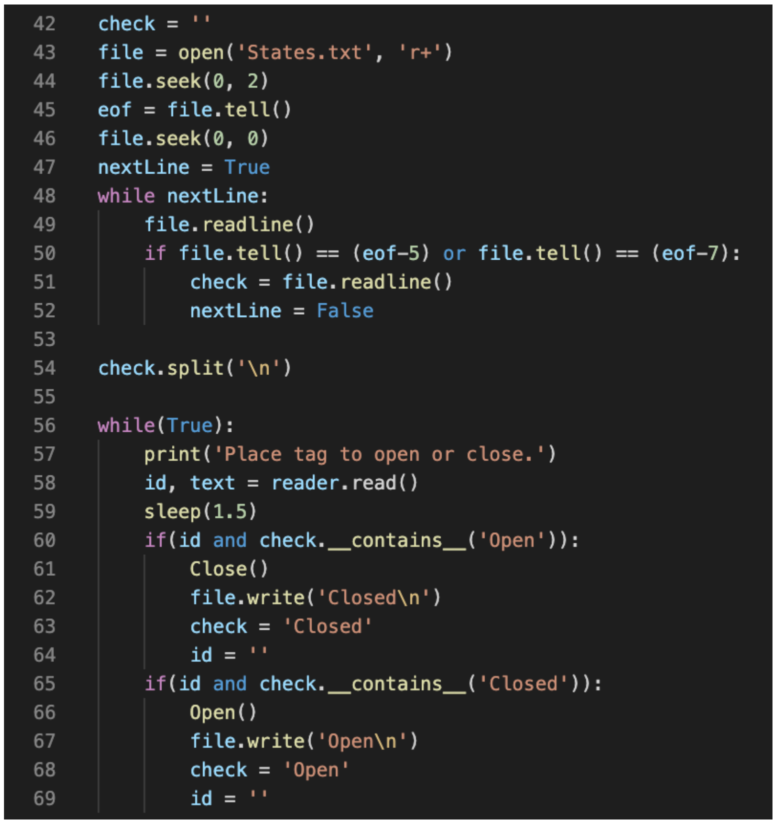
The software present in the final stable version of this project was written for the Raspberry Pi in Python. Stepping through its function, it initializes a reader variable that represents the RFID module as well as sets up all GPIO pins and states what they are for. After setting up these initial variables, three main functions are defined for later use. The first of these three functions is the Open function. Open sends a digital high signal to the h-bridge pin responsible for opening the lock, waits for 5 seconds, then turns it off. This time allows the lock time to open. The open function also turns off the red Light-Emitting Diode (LED) that indicates the lock is closed and turns on the green LED indicating the lock is open. The close function does the opposite, setting the pins for close high and turning on the red LED. Finally, the handler function is an exit function that listens for the user to input Ctrl+c and asks to confirm the intention to exit. If y (or Y) is input, the GPIO pins are cleaned up and the output file is closed and the loop is exited. These functions and the pin setup can be seen in Figure 2.

The next half of the control software is the main loop and where the logic takes place. It begins by opening States.txt which stores the current state of the lock and navigating to the last line of the .txt file. This last line shows the most recent state of the lock as either "Open" or "Closed" and sets the variable check equal to the last line. The main loop is then entered and prompts the user to present an RFID tag to the reader. Once the tag is held to the reader, the id variable is populated with the tag's id. The two if statements check to make sure id is populated and that the check variable contains either "Open" or "Closed". If those conditions are met, the Open or Close function is called based on the check variable, a new value is written to States.txt, check is set equal to the opposite of its current value, and id is set to an empty string and the loop continues. This loop continues until the user inputs Ctrl+c to exit the program. This main functionality can be seen in Figure 3.


FinalRaspberry.py > ...

```python
import RPi.GPIO as GPIO
from time import sleep
from mfrc522 import SimpleMFRC522
import signal

GPIO.setwarnings(False)

reader = SimpleMFRC522()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(15, GPIO.OUT, initial = GPIO.LOW) #Counterclockwise -- Close
GPIO.setup(13, GPIO.OUT, initial = GPIO.LOW) #Clockwise -- Open
GPIO.setup(31, GPIO.OUT, initial = GPIO.LOW) #Red -- Closed
GPIO.setup(33, GPIO.OUT, initial = GPIO.LOW) #Green -- Opened

def Open():
    GPIO.output(13, GPIO.HIGH)
    GPIO.output(31, GPIO.LOW)
    print('opening')
    sleep(5)
    GPIO.output(13, GPIO.LOW)
    GPIO.output(33, GPIO.HIGH)
    print('Open')

def Close():
    GPIO.output(15, GPIO.HIGH)
    GPIO.output(33, GPIO.LOW)
    print('closing')
    sleep(5)
    GPIO.output(15, GPIO.LOW)
    GPIO.output(31, GPIO.HIGH)
    print('Closed')

def handler(signum, frame):
    res = input('Ctrl+c was pressed. Do you want to exit loop? Y/N: ')
    if res.lower() == 'y':
        GPIO.cleanup()
        file.close()
        exit(1)

signal.signal(signal.SIGINT, handler)
```

Figure 2: Control Software Setup



```python
check = ''
file = open('States.txt', 'r+')
file.seek(0, 2)
eof = file.tell()
file.seek(0, 0)
nextLine = True
while nextLine:
    file.readline()
    if file.tell() == (eof-5) or file.tell() == (eof-7):
        check = file.readline()
        nextLine = False

check.split('\n')

while(True):
    print('Place tag to open or close.')
    id, text = reader.read()
    sleep(1.5)
    if(id and check.__contains__('Open')):
        Close()
        file.write('Closed\n')
        check = 'Closed'
        id = ''
    if(id and check.__contains__('Closed')):
        Open()
        file.write('Open\n')
        check = 'Open'
        id = ''
```

Figure 3: Main Functionality Loop

7 Confounding Factors and Regressions

There were things that went wrong or that were considered regressive to our work before we were even hands on with the project. In addition to common setbacks such as wiring complications and circuit issues that were solved easily, there were other confounding factors that occurred such as trouble with time management, issues with the LINKA Lock PCB components, as well as interfacing problems between the Raspberry Pi and RFID and also establishing serial communication between an Arduino Nano with the Raspberry Pi (RPi).

7.1 Time management

The main regressive factor early in the fall semester was the time it took to choose which approach we wanted to build off of for the project. Eventually this was solved after weighing the pros and cons of the different approaches and choosing the one that best suited the specifications that we were looking for. This lead to another set back during the fall semester and that was the time management of the project, we were not able to find consistent times that we were all able to work on the project and that slowed down the progress we made during that semester, this was not as much of a setback in the spring semester.

7.2 LINKA PCB components

Our original plan with the LINKA Lock was for us to deconstruct it and replicate its functionality on a PCB of our own design. However due to the difficulty of identifying the specific parts of the LINKA Lock we were unable to design our own PCB with the necessary parts. We ended up solving this setback by modifying the LINKA Lock instead of designing our own PCB.

7.3 Interfacing Problems

When testing the Raspberry Pi and the RFID module, there were some interfacing problems where the RFID module would not read the tag and operate correctly. This was solved by replacing the RFID module, as the original seems to have malfunctioned.

7.4 RPi and Arduino Nano

At the end of the semester we considered establishing serial communication between the Raspberry Pi and an Arduino Nano. We were able to get some functionality by using a USB connection but when focusing on using the TX and RX pins on the Arduino we were unable to implement this we the remaining time we had so ultimately this was a factor that was not solved and did not make it into the final build of the project.

8 Key Learning Takeaways

This project was a great learning experience overall. It helped us all learn to work better in teams with conflicting schedules as well as work together based on different team members' strengths. We all grew in technical knowledge as well as written/oral communication and presentation in a semi-professional environment. For technical skills, we are all now very familiar and comfortable with different types of GPIO with Raspberry Pi as well as how to connect and manage solar power charge methods for small-scale projects. Though it wasn't completely functional at the end, the serial communication between MCUs taught us a lot about serial communication as a whole and the different communication methods between devices. Throughout the project, we also learned the importance of examining the detailed specifications of project parts, as it occasionally led to issues with our parts we ordered and almost burned out our Raspberry Pi later on. The most valuable takeaway from this project is the team-building aspect and how to play to individuals' strengths along with the gained experience with the software and hardware methods we used.

9 Future Work

For future teams developing the Shark Park, this semester's Team Proton has laid a decent groundwork and have some solid ideas for future improvement.

More information for future teams is included in the README.md of the Shark Park GitLab repository. The improvements mentioned are primarily improvements to software function, hardware integration, and general structure. The next few subsections will detail the ways in which these categories could be improved.

9.1 Software Suggestions

The FinalRaspberry.py file is the final stable code that functioned as our demo in the Spring semester. It works well with the lock we implemented and is stable for the foreseeable future. However, there are some improvements that could be made (including some rather easy fixes). These fixes and improvements will be discussed in the order they would appear in the file. First, it would be much better for the States.txt file to only ever store one line either "Open" or "Closed". Currently it writes every time there is an update. For our purposes, this is fine, but could result in storing potentially megabytes of just Open, Closed, Open, Closed... The ability to rewrite that and have one line would be best. Also, for the further future, teams will need States1.txt, States2.txt, etc. to manage multiple pucks as well as reader1 (RFID reader), reader2, etc. This leads into the next major piece of work that will not be as easy. If there is a parking rack with 5 pucks, that rack will have 5 RFID readers and 5 States.txt's. Future teams will need to create a way for any of those to read and open/close at any time. Another improvement would be a queue system in the edge case that two users present tags at the same time. Those are the main software improvements we suggest.

9.2 Hardware Suggestions

Our current hardware configuration is what is shown in Figure 1 connected out over a breadboard. It worked for our purposes, but breadboard connections are spotty at the best of times and the wires were occasionally problematic. These issues could be resolved by consolidating all the functionality to a single PCB. This will likely be one of the primary focuses of future teams. The first hardware issue that needs to be solved however is the connection and serial communication with the Arduino Nano (or another small Micro Controller Unit (MCU) if teams choose). This will help scaling the parking station to more parking spots. We have in the GitLab repository serial communication tests that worked over USB Serial, but need to be able to use TX/RX pins for communication instead. These two improvements are what will need to be implemented next.

9.3 General Suggestions

This section will be a sort of catch-all for suggestions we have. The project would benefit from a pseudo-bike rack made of cardboard or some other easy-to-use material. This would help future teams get a better idea of how the bike rack would work and how to make longer-distance connections between the Raspberry Pi and each smaller MCU (currently Arduino Nano). As SharkCycles teams expand, another suggestion would be to meet with other teams across disciplines to make sure everyone is on the same page. This might be difficult with conflicting schedules, but we think it would benefit all SharkCycles teams.