Let's Ride

Ali Abdel-Maksoud, EE, Benjamin Ledoux, CSE, Syed Ali, CSE, and Xavier Farrell, EE

***Abstract*—As Covid-19 continues to shutdown or limit gyms many workout enthusiasts are turning to working out at home. Peloton and other internet connected stationary bicycles have seen huge success by allowing users to connect and workout with others virtually while tracking their workouts and progress. The problem with the currently available solutions is that they are expensive, difficult to move or transport, and cannot bring their experience to the outdoors. Our product, Let's Ride, solves these problems by providing an inexpensive system that can be easily installed on a user's own bike to communicate live ride data to an iOS app. Let's Ride allows users to work out with others virtually either indoors on a stationary bike stand or outdoors on any terrain.**

I. Introduction

Americans are looking for ways to remain active and healthy amongst the changes due to the COVID-19 pandemic. According to the Washington Post, "Bicycle sales nationwide surged by 50 percent in March (2020), according to the NPD Group, a market research company. It reported a 121 percent increase in adult leisure-bike sales…" [1]. Furthermore, a survey drawn from 1000 reports "nine in 10 Americans who exercise regularly say they will continue with at-home workouts even after they feel comfortable returning to a gym in the future" [2]. These two statistics indicate there is a large market for new fitness products that are tailored to biking and workout enthusiasts

A. Significance

Americans are looking for ways to remain active and healthy amongst the changes due to the COVID-19 pandemic. According to the Washington Post, "Bicycle sales nationwide surged by 50 percent in March [2020], according to the NPD Group, a market research company. It reported a 121 percent increase in adult leisure-bike sales…" (Davies, 2020). Furthermore, a survey drawn from 1000 reports "nine in 10 Americans who exercise regularly say they will continue with at-home workouts even after they feel comfortable returning to a gym in the future"(King, 2020). These two statistics indicate there is a large market for new fitness products that are tailored to biking and workout enthusiasts.

B. Context and Competing Solutions in Marketplace

There are several other products on the market that allow users to track their rides and there are many Internet-connected stationary bikes available. Strava and other workout apps allow users to record their rides and compare stats with others. The ride tracking for these apps is done with the phone's GPS and a maps API. Peloton and its derivatives track users performance on a stationary bike and connect with other workout groups and classes for live workouts. What makes our product different is that it is the same system for both stationary stand biking and outdoor rides. Purely stationary bikes cannot do outdoor workouts and are large and hard to move, as well as expensive. Out product is much less expensive and can be installed onto a bike the user already owns, saving a considerate amount of space and is much easier to move.

C. Societal Impacts

Our users are biking enthusiasts who want a way to record their rides and have connected rides with friends or random players. It also includes general workout enthusiasts who want the stationary bike/Peloton/spin class experience in their own home but can't use the competing solutions due to cost, space concerns, or wanting to be able to take the experience outside. One concern for us is safety for users in cities, where they may need to wait at lights during their ride but could be tempted to run through them or make unsafe decisions to get ahead in a ride. Besides this we believe our system will have no negative societal impact, it will serve to provide an alternative to the existing solutions that may not work for certain users. Our main focus is making a system that is easy to use and much cheaper than connected closed systems like Peloton. With standard Bluetooth communication and a simple and published messaging system anyone could create another app to access the bike data, preventing the Let's Ride system from becoming e-waste if the official app and service were discontinued.

D. System Requirements and Specifications

For our rider's experience we want the system to be frustration-free and easy to use. The system must be able to be set up by a user with little tool skills and bike repair skills in less than one hour. A major frustration for users with anything that uses Bluetooth are connection issues. When the user approaches the bike with attached system and starts a race the Bluetooth module should connect to the iPhone on the first attempt >90% of the time. Riders should not have to worry about rain or dust from the road destroying their system, so the completed system must be IP34 complaint. The entire system must weight less than 2 lbs and not interfere with the riders normal riding motion to keep users from getting annoyed with the physical system.

The power system should keep the users from having to worry about whether their system will stay on for a long ride.

Ali Abdel-Maksoud from Amherst, MA (e-mail: author@ boulder.nist.gov).
Benjamin Ledoux, from Lowell, MA (e-mail: bledoux@umass.edu).
Syed Ali, from Chelmsford, MA (email: smali@umass.edu)
Xavier Farrell from Springfield, MA (e-mail: xfarrell@umass.edu).

So the power system should be able to keep the system on for at least 8 hours, and provide usable electric power at speeds over 3.2 m/s (~ 7mph). For the system to be useful it must be accurate in its measurements. We define the accuracy requirements for our sensors system in Table 1.

Feature	Description	Spec	Result
Portable	Weight	< 2lbs	System < 0.9lbs
	Size	< 6 cubed inches	System < 4.5 cubed-inches
	Mountable	Cannot cause significant immediate damage to bike	Clamps and zip-ties
Safe	Electrically	Short circuit and/or overvoltage protection Off/ON Button	Achieved: Voltage protection board limits battery voltage to 4.18V and Zener limits input voltage to 24V
	Physically	No direct contact with electronics while in use No sharp or pointed edges	Achieved: Physical Enclosure
	Functionally	Cannot induce a sudden stopping event	Achieved: Robust mounting and detachment scheme
Ride Condition Insensitive	Output power in varied weather	5V 300mW for speeds > 7mph in dry and moderately wet conditions	Achieved: Battery powered the system from with IP34 rated box
Durable	Physical Strength	Withstand 2 drops from mounting height	Untested: IP34 rating and metal dynamo encasing suggests achievable
Reusable	Can supply power at multiple separate and distinct times of use	Estimate product life expectancy	Achieved: 20,000 workouts over more than a year

Table 1: Requirements and Specifications

Portable: Based on bicycle price ranges and target customer skill level, we assume our product consumers will own bicycles between the weights of 20 lbs (~9.1 kg) to 25 lbs (~11.4 kg) [3], [4]. For this bicycle weight range, our final system must not increase the unloaded bike weight by a maximum of 7%. The cumulative weight of the entire system is < 0.9 lbs and will increase an anticipated unloaded bicycle weight by < 5% satisfying this specification. The system is also mountable. The generator is mounted on the rear bike fork via clamps secured by screws. The PCB and battery are placed within an off-the-shelf electrical box and zip-tied to the underside of the bike seat. Wiring is zip tied to the frame of the bike. Specifications regarding size and shape are aimed at minimizing increased drag experienced when riding. Therefore, the system must not extend beyond the width of the bicycle's saddle. This requires that the system width must be less than 4.7 if centered along the axis of the bicycle's length [5], [6]. As a system that is not intended to contribute to the visual experience of the product, discretion is of value. This translates to size specifications that maximize discretion without prohibiting functionality. Therefore, the power system should not exceed 6 inches in height or length. Our system is placed within an off-the-shelf 4"x4"x4" electrical box, satisfying both requirements.

Safe: Electrical safety measures such as over-voltage protection and short circuit protection are implemented. The charge control circuit used for the lithium-ion battery is dependent on the sensed charge status of the battery which is broken into three regions: the over-voltage, nominal voltage and under-voltage region. If the battery voltage is in the nominal voltage region, power is allowed to flow from the generator to the battery and the load. If the battery voltage is in the over-voltage region, the electrical power from the generator is prohibited from flowing to the battery and load and the battery supplies power to the load. In both of these operating conditions, power can flow from the battery to the load. If the battery voltage is in the under-voltage region, the battery is electrically disconnected from the load and no power can flow to the load; therefore, the system will shut down. The primary functional safety measures are implementing robust mounting infrastructure such as metal clamps and zip-ties to prevent abrupt stopping events. The box is mounted underneath the seat which is above the rear wheel. It is advised that in mounting the box underneath the seat with zip-ties, one side should have twice the zip-ties as the other. This has beneficial safety impacts in the event of a zip-tie failure, causing the box to swing toward the strong side and away from the rear wheel as opposed to falling directly onto the tire. It is advised that the strong side be opposite to the side where the gears are located. Lastly, under normal system operation, power system components near the rider within an enclosure with a durability rating of IP34 [7]. Therefore the rider is not exposed to the electrical components while they are under use. Furthermore, the enclosure does not have sharp edges and is mounted to the bike frame in a way that does not obstruct rider movement or bike functionality.

Functional Across Ride Variety: The battery size is able to supply power to the system under normal operating conditions for a minimum of 8 hours. The system is anticipated to be able to supply power for longer periods when the generator is operating under nominal conditions. Therefore the system is insensitive to various ride durations. The power system is able to supply usable power over the range of target consumer riding speeds at a minimum of 7 mph and upwards of 25 mph. Lastly, the system can supply usable power in moderately wet conditions.

Reusable: The wear item of this system is the rechargeable lithium-ion battery with its life expectancy quantified by the number of charge cycles before the effective energy capacity reduces to 80%. The equations below details the parameters used to estimate the number of workouts and full days of use our product will offer before replacing the first wear item.

Assumptions

- Workouts are 30 minutes long (2 workouts/hour)
- 200 charge cycles (worst case estimate as compared to healthy battery use of 400-500 charge cycles) [8].

Using the parameters and assumptions defined above, the minimum estimated number of workouts offered by this product without buying new equipment is 20,000 workouts and 400+ full days of use. Therefore the system is reusable.

$$Hours\ of\ Operation = \frac{Battery\ Rating}{Load\ Current}$$

$$Number\ of\ Workouts = \frac{2\ Workouts}{Hour} * Hours\ of\ Operation$$

$$Total\ Number\ of\ Workouts = \frac{Number\ of\ Workouts}{Charge\ Cycle} * Number\ of\ Charge\ Cycles$$

$$Total\ Operable\ Hours = \frac{Hours\ of\ Operation}{Charge\ Cycle} * Number\ of\ Charge\ Cycles$$

$$Total\ Number\ of\ Days = \frac{Hours}{Day} * Total\ Operable\ Hours$$

II. DESIGN

A. Overview

Our design can be broken down into three distinct subsystems. The power system, the microcontroller and sensor system, and the iOS app with Google Firebase backend. Figure 1 shows our system block diagram.

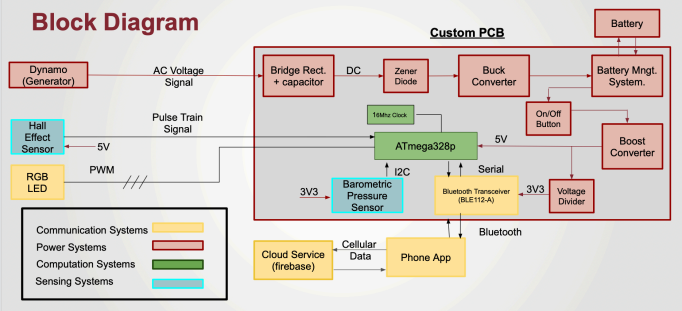

Figure 1:Block Diagram Describing our System.

The microcontroller and sensor system uses the power from the power system. The ATmega328P and hall effect sensor run off the 5V rail and the BLE112A Bluetooth Low Energy [9] module and BMP388 barometric pressure sensor run off the 3.3V rail. The ATmega328P controls the handlebar RGB LED via three pulse width modulation lines (labeled PWM in Figure 1). The hall effect sensor sends its output through a Schmitt trigger which then arrives at the ATmega328P in the form of a pulse train. On the rising edge of this pulse train an internal counter increases in the ATmega328P to count the amount of times a magnet on the wheel passes the hall effect sensor.

The battery size will be able to supply power to the system under normal operating conditions for a minimum of 8 hours. The system is anticipated to be able to supply power for longer periods when the generator is operating under nominal conditions. Therefore the system is insensitive to various ride durations. The power system will be able to supply usable power over the range of target audience riding speeds at a minimum of 7 mph and upwards of 25 mph. The final design will detail power dissipation to internal heat generation and ensure temperatures do not induce system failure. Lastly, the system will include a slippage reduction component to increase friction between the generator and tire sidewall in moderately wet conditions.

Durability is achieved through the use of high quality materials and cushioning. The generator features metal encasing and mounting attachments increasing durability. The PCB and battery will be enclosed in a high quality off-the-shelf biking pouch designed to secure phones and other possessions.

The power system utilizes a rechargeable lithium-ion battery. The generator and IC components are purchased and/or designed with the intent of reusability. The final system has an off site charger that can be used to recharge batteries overnight. These components have a lifespan that allows for multiple ride experiences and therefore the system is reusable.

The iOS app communicates with the Bluetooth module through Apple's Core Bluetooth Library [10]. This library is the only official way for iOS devices to utilize Bluetooth peripherals. Player sign in, matchmaking, and the live game are all handled by Apple's GameKit service. [11] Sign in is done with the Apple GameCenter account so users do not have to create a new account. GameKit utilizes Apple's servers to do matchmaking and invite other riders, as well as send data to all other riders in a race. User data and race history is stored in Google's Firebase database service. The iOS app is programmed in Apple's own Swift programming language which is based on Objective-C. None of us knew Swift or UI Design with the Xcode IDE so we had to read a lot of documentation and go through multiple tutorials provided by Apple.

B. Power System

The power system used in our project consists of a generation source, power electronics for conditioning power flow and an energy storage device. The generation source is a sidewall bicycle dynamo that outputs AC power and is referred to as the generator. The generator converts rotational energy from the rotating bike tire during a ride to electrical power. A bridge rectifier and capacitor is used to convert the AC output to a DC input and a zener diode is used for over-voltage protection to prevent damage to the following buck converter. The DC input is stepped down by a buck converter and used to charge the battery through an over-voltage protection circuit. The battery is a 4.2V lithium-ion 18650 battery cell. A boost converter raises the output of the battery to a voltage within the input voltage range of the load.

C. Microcontroller

We are using the ATmega328P microcontroller [12] with firmware written in C++. Our microcontroller reads in data from our two sensors and messages from the bluetooth module, and from this input data decides how to control the handlebar mounted LED and what message to send back to the bluetooth module. Designing this block involved reading the data sheet for the ATmega328P to learn how to utilize different features on the board and communicate with the other parts of our system. The microcontroller receives and sends data to the Bluetooth module using a serial UART connection. To read data from the Bluetooth module the microcontroller pops the next available byte from the UART FIFO and decides what to do next based on the value of that byte. To send data to the Bluetooth module the microcontroller we write a string to the serial connection. The microcontroller uses I2C protocol to communicate with the BMP388 altitude sensor, the microcontroller sends a read request to the sensor and adds the float value returned to the message to go to the Bluetooth module. To read in the distance traveled data we set the internal 16-bit TCNT1 counter on the ATmega328p to increment on the rising edge of the PD5 pin. When a magnet passes the hall effect sensor the signal from the hall effect goes high. We have a Schmitt trigger between the microcontroller and the hall effect sensor to convert the output of the hall effect into a digital signal. We used the Microchip Studio IDE and an Atmel ICE to develop, load, test and debug the firmware.

D. Sensors

For the sensing system we used two main sensors to gather all required information, a hall effect sensor and a barometric pressure sensor (BMP388). The hall effect sensor is able to sense magnetic fields as they pass by its sensing area. We took advantage of this effect to measure the distance traveled on the bike. We did this by clipping six neodymium magnets equidistant from each other on the outer end of the spokes. This helped us achieve the initial goal of being ∓0.6m of desired accuracy. Since the circumference of the wheel was known, as one magnet went by a certain distance was traveled and we were able to calculate the overall distance traveled that way. We used the SPI protocol to communicate with the sensor [13].

The BMP 388 was used to measure the change in elevation. The change in elevation was used as a form of subjective rewarding system to those who decide to bike on incline surfaces rather than just a straight line. The BMP 388 was able to measure the air pressure and the temperature and from that it was able to output the local elevation. We used I2C protocol to communicate with this sensor [14].

E. Bluetooth Module

To enable Bluetooth connectivity we are using the Silicon Labs BLE-112A-V1 module which utilizes the Bluetooth Low Energy standard [15]. This modules includes its own antennae. To implement the Bluetooth module we had to learn about the Bluetooth 4.0 standard, GATT profiles, and Bluetooth services and characteristics. We chose the Bluetooth Low Energy (also known as Bluetooth 4.0) protocol because it is the protocol that iOS supports for apps to use for connecting to peripheral devices (on iOS Bluetooth Classic 2.0 can only be used for headphones and speakers). The Bluetooth modules communicates with the board through a serial UART connection. To test the Bluetooth module we started with communicating with a Bluetooth serial terminal on an Android phone, and then moved to using an iOS app we developed to use as a test bench for the Bluetooth communication between the iPhone and the Bluetooth module. Once the Bluetooth communication code was functional and tested we integrated it into the main iOS app and did additional testing.

F. Let's Ride iOS App and Data

Our app design is mainly predicated on using developer API's provided by apple, called GameKit, and along with that our cloud infrastructure was supported by Google Firebase. Our app design was split into 5 main Views, also called ViewControllers in Xcode. These consisted of:

1. SignUpViewController
2. HomeScreenViewController
3. ReadyUpViewController
4. GameViewController
5. EndGameViewController

We also designed a gameKitHelper singleton class which implemented the necessary methods for communication between the ViewControllers and Apple's gameKit API. We will discuss this helper class more towards the end but first let's explore the design and functionality of the ViewControllers.

SignUpViewController is the view that users see when they first open the app for the first time.

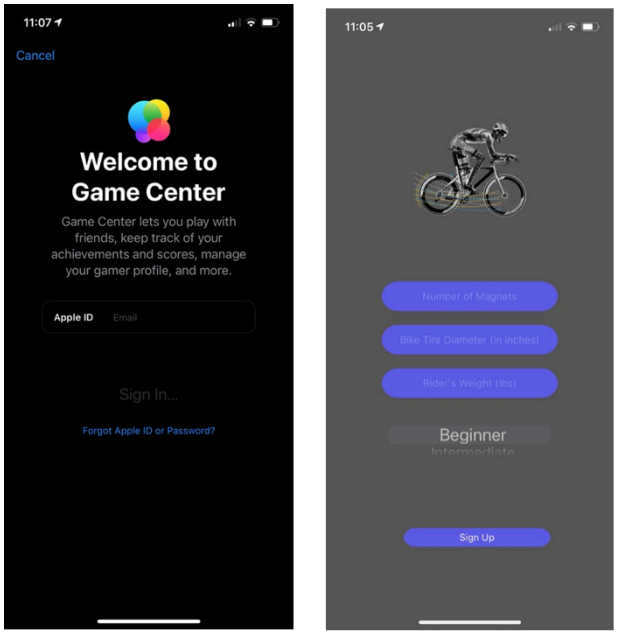

Figure 2: iOS app start screens

First time opening app: If users are not signed into their Game Center account on their IOS device they are first asked to either log in or sign up for one as it is necessary for matchmaking (Figure 2), we will be doing later in the app. Once logged into their Game Center, the app will automatically take the credentials provided by Game Center and create a user account in firebase authentication. This only gets done once, the first time the user opens the app. Once that is done users are asked to enter the number of magnets on their bike, the diameter of their bike tire in inches, their rider Weight (optional), and lastly their skill level (Beginner, Intermediate, Advanced) (Figure 2). Once users hit sign up this information is uploaded the firebase Firestore database and assigned to the users UUID that Game Center provides. This data is later retrieved automatically from the database every time the user opens the app and stored in a bike data struct.

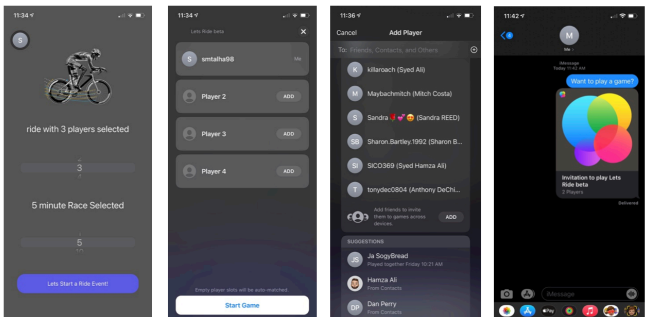

Figure 3: Ride invite process

Opening the App after signup: Once users have all the information entered the screen the proceeding time, they open the app, once game center authenticates with its servers and passes the authentication to firebase where the bike data is retrieved this screen automatically will transition to the HomeScreenViewController. So, users will have no need to keep entering their information.

HomeScreenViewController: This view represents how riders can start a match with other users and can configure what type of match they would like to start.

As shown in Figure 3 users are asked how many people, they would like to ride with in real time along with the length of race they would like. As users scroll through the value the text fields above them changes to indicate the number of riders they want to ride with and the length of race. Once "Lets Start a Ride Event" is pressed, users are shown a matchmaking screen (Figure 3). Here they will see the number of users they selected to be in the race along with an ADD button next to each player. This lets users invite their friends through a iMessage invite link that is sent (Figure 3) and also as a push notification to the users IOS device alerting them that their friend wants to play. Once this invite link is clicked it automatically opens the app, authenticates the user, and puts them directly in the matchmaking screen shown in figure 4 along with the other users in the game. However, if the user wants to play with any stranger looking to ride with, instead of adding a user they can just press start Game and they will be automatically paired with other users with similar skill levels, race type, and number of users. Once matchmaking is complete the users are transitioned to the readyUpViewController.

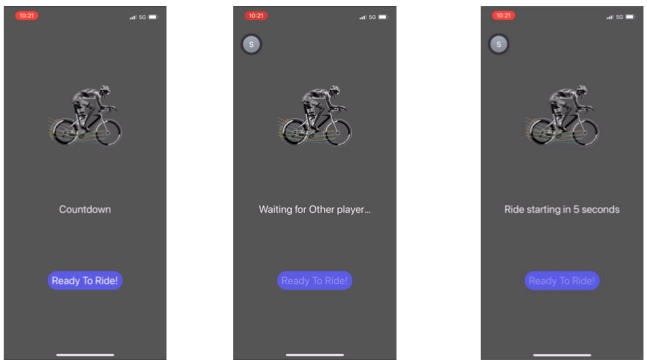

Figure 4: ReadyUpViewController - ride countdown progress

ReadyUpViewController: This view occurs after all users have been put into the match and allows each to ready up before starting the ride.

As shown in Figure 4, the users first see this view which asks them to press "Ready to Ride!" button. Once this button is pressed, they get a message saying, "Waiting for Other Players to ready up" and the Ready to Ride! Button is greyed out so users cannot force start the game (Figure 4). Once all users have pressed "ready to Ride!", a 5 second countdown timer start alerting the users that the ride is starting. This countdown gets decremented every second so users visually know how much time is left before the ride starts. Once the countdown ends users are transitioned to the GameViewController.

GameViewController: This view is the main match view and shows real time data between the user and other riders.

As shown in Figure 5, once the match starts users can see a countdown timer for the time remaining in the match along with their current score that is updated continuously as soon as it is received through Bluetooth. We also get the score for the opponent that has the highest score so that you know exactly how far you're either from first place or how far the closest rider is to you if you are in first place. Once the timer has reached 0, the match is over and the EndGameViewController is shown to the user with the match statistics.

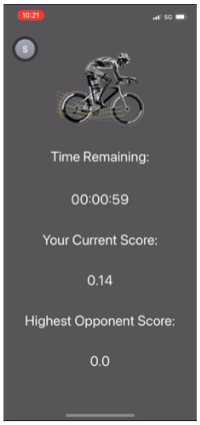

Figure 5: Live game view

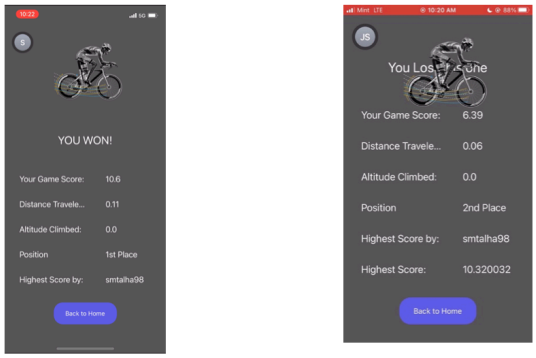

Figure 6: End game view

EndGameViewController: This view shows the match statistics after the match is over and provides a way to get back onto the home-screen.

As shown in Figure 6, the user is displayed a custom message based on whether they won or lost, as well as their game score, distance they traveled in miles, altitude climbed during the ride, their position, and the username of the player with the highest score. If you lost the match, you are shown the same information as when you win but a custom message tells you "you lost this one" along with the highest score of the user who won. All this data is also processed and uploaded to FireBase Firestore database. The "Back to Home" button takes you back to the home-screen.

GameKitHelper class: This class represents the main communication between Apple's Game Center servers and the rest of the view controllers. This was designed to be a

singleton so that all views could access and write and read from the class without instantiating a new class. The main purpose of this class is to

1. Access game center matchmaking
2. Communicating game data between users
3. Handling invites (sending and receiving)

It does this through what swift language calls delegates. Delegation is a design pattern that enables a class or structure to hand off (or delegate) some of its responsibilities to an instance of another type. In our GameKitHelper class we inherit different delegates that are invoked by the game center servers when a certain specified action occurs. Our gameKitHelper specifically extends GKMatchmakerViewControllerDelegate, GKMatchDelegate, and lastly GKLocalPlayerListener. The GKMatchmakerViewControler delegate has us implement methods that relate to the matchmaking aspect of creating a match. The delegate of a GKMatchmakerViewController object implements this protocol to handle when players accept invitations, the player cancels matchmaking, or an error occurs. Once matchmaking is done it creates a match object. This receives connection status and data transmitted in a multiplayer game via the GKMatchDelegate. Lastly, the GKLocalPlayerListener protocol is used to listen for and handle a variety of Game Center events for the local player such as the GKInviteEventListener which is what we use to handle invites to matches.

VI. The Refined Prototype

A. Prototype Overview

Figure 7: Refined prototype mounted on bike

Our refined prototype sits inside a weather proof enclosure as seen in Figure 7. It is mounted underneath the seat of the bike and then the hall effect sensor comes out as a wire and is mounted on a the bar next to the wheel. Magnets adhered to clips go on the spokes of the wheel lining up with the hall effect. The bike dynamo connects at the top of the wheel and its connected to the leads from the system. The RGB LED is connected to the system and mounts on the handlebars.

B. List of Hardware and Software Used

Software used:

- Apple's Xcode IDE
- Microchip Studio IDE
- Swift, C++ programming languages
- Google Firebase
- Apple GameKit

Hardware used:

- ATmega328P microcontroller
- Bosch BMP388 Barometric Pressure and Altimeter
- Silicon Labs BLE112A Bluetooth Low Energy Module with Antenna
- Tung Lin 4 Pole Bike Dynamo
- LM2596 Buck Converter
- 8205A Chenbo Battery Charging Board
- TPS63060 Boost Converter
- KBP2005G Full Bridge Rectifier
- Zener Diode 24V 5W
- EBL 18650 Li-Ion Battery

C. Custom Hardware

The custom PCB layout shown in Figure 8 contains all the ICs for the power system as well as the ICs for the data collection and communication system as well as the BMP 388. From top left to top right it goes bridge rectifier, buck converter, a PMOS followed by NMOS and slightly below them is the voltage protector. To the right of that stage is the boost converter. In Between them is an assortment of capacitors, inductors, and resistors. The middle and bottom of the board is populated with the bluetooth controller on the far mid right and directly below it is the ATmega328p. Below the ATmega328p to its farthest right is the BMP388. There are pin headers that lead to the Hall effect as well as ones that lead to the battery and the dynamo.

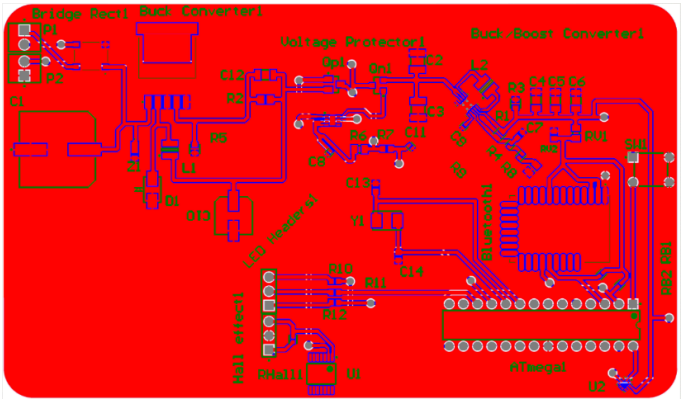

Figure 8: PCB Layout

D. Prototype Functionality

Our refined prototype at FPR was able to be mounted easily to stationary and moving bikes, and came in a robust and weatherproof enclosure. The system was able to be powered by the battery for extended periods of time and riding the bike with the dynamo attached resulted in charging the battery. The hall effect sensor collects data from counting the amount of magnets passing the sensor detected. The altitude sensor reports an altitude but it is stuck at a single altitude currently. The microcontroller is able to package this data and reliably send it via Bluetooth to the iOS app. The iOS app is

able to process these messages, and update scores during the game. The iOS app also allows for users to sign in with their Apple GameCenter account and authenticate with FireBase to download their user settings. Users can then start races and either invite their friends or be automatically matched up with a stranger that is looking for the same type of race. The app sends data in real time to the other riders in the race and sends messages to the microcontroller via Bluetooth on how to update the handlebar status LED on the bike. The app also uploads the match data to firebase after the match is completed.

E. Prototype Performance

At FPR we were able to reach ~87% accuracy of counting bike wheel rotations with the hall effect sensor. Our tests were slightly undercounting the amount of magnets passing the hall effect sensor. Our altitude sensor stopped reporting any change in altitude after transitioning to the soldered protoboards. Our Bluetooth module was able to automatically connect to the app 94% of the time, and when not automatically connected would connect the next time after toggling power to the system. The RGB LED would change within 2 seconds of a place change coming into the local iOS app.

A timed installation of the system took 22 minutes to complete, and the system did not interfere with the rider's motions when biking. The system was able to function after being left in the rain for 4 hours.

The iOS app ran reliably over cellular data and WiFi connections. When losing connections the app would continue to connect data from the microcontroller, and send the newest data once reconnected to the internet. When the Bluetooth lost connection the microcontroller would continue to read in data, and then send the latest data once the iOS app was able to reconnect with the Bluetooth module.

The app supported up to 5 riders in a single ride.

VII. Conclusion

At MDR we were slightly behind our goals, with the microcontroller not being able to communicate with an iOS app yet. We were also short on the matchmaking deliverable. At CDR we were able to meet all of our deliverables and present a fully integrated and usable system that was on breadboards. We had the blank PCB available for our first revision, however another revision was needed after CDR to fix missing address select and mode select lines to our altitude sensor. The altitude sensor pads site below the device and are extremely small, so another revision was necessary as trying to modify the board manually would be very difficult. We were able to get part of the board working but we had problem getting a switch on the voltage protector to activate so we couldn't supply power to the system. For FPR we had to settle for using soldered protoboards in our system. When transitioning to the protoboards we started having problems with our altitude sensors, where they would only give us a constant altitude and not change. Our app and microcontroller system worked as planned for FPR, and we were able to use the system on both stationary bikes and on road rides.

Acknowledgment

We would like to thank our project advisor Professor Soules for helping us through this project, and Professor Epstein for helping us gather materials and solder our PCB. We would also like to thank our project evaluators Professor Kundu and Professor Janaswamy.

References

[1] E. Davies, "What Do Bikes and Toilet Paper Have in Common? Both Are Flying out of Stores amid the Coronavirus Pandemic." *The Washington Post*, WP Company, 28 May 2020, [Online] www.washingtonpost.com/local/what-do-bikes-and-toilet-paper-have-in-common-both-are-flying-out-of-stores-amid-the-coronavirus-pandemic/2020/05/14/c58d44f6-9554-11ea-82b4-c8db161ff6e5_story.html. [Accessed May 3, 2021]

[2] R. King, "Most Americans Plan to Continue at-Home Workouts Even Once Gyms Fully Reopen." *Fortune*, Fortune, 17 Aug. 2020, [Online] fortune.com/2020/08/17/covid-workouts-at-home-gyms-reopening-coronavirus-fitness/. [Accessed May 3, 2021]

[3] B. Lin, "How Much Should You Spend on Your First Bike?" *The Pro's Closet*, 24 Sept. 2019, [Online] www.theproscloset.com/blogs/news/how-much-should-you-spend-on-a-bike. [Accessed May 3, 2021]

[4] T. Whitehouse, "How Much Should a Road Bike Weigh?" *Road Bike Basics*, 5 Jan. 2021, [Online] roadbikebasics.com/how-much-road-bikes-weigh/. [Accessed May 3, 2021]

[5] BikeFit. "Sit Bones Width Measurement and Bike Saddle Selection." BikeFit, 16 June 2020, [Online] blog.bikefit.com/sit-bones-width-measurement-and-bike-saddle-selection/. [Accessed May 3, 2021]

[6] Bikinguniverse. "How to Measure Your Sit Bones – Bike Saddle Size & Fit." *Bikinguniverse*, 14 Oct. 2019, [Online] www.bikinguniverse.com/measure-sit-bones-bike-saddle/. [Accessed May 3, 2021]

[7] IEC 60529, "degrees of Protection Provided by Enclosures (IP Codes)," Ed. 2.1 (Geneva: International Electrotechnical Commision, 2011), [Online] https://www.iec.ch/ip-ratings [Accessed May 3, 2021]

[8] I. Buchmann, "BU-808: How to Prolong Lithium-Based Batteries." *Battery University*, 29 Jan 2016, [Online] batteryuniversity.com/learn/article/

how_to_prolong_lithium_based_batteries. [Accessed May 3, 2021]

[9] BLE112A-V1, Data Sheet, *Silabs*, https://www.mouser.com/pdfdocs/BST-BMP388-DS001-01.pdf

[10] Developer.apple.com. 2021. *Apple Developer Documentation*. [Online] https://developer.apple.com/documentation/corebluetooth

[11] Developer.apple.com. 2021. *Apple Developer Documentation*. [Online] https://developer.apple.com/documentation/corebluetooth

[12] ATmega328p, Data Sheet, *Microchip*, https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[13] Unipolar Hall Switch, Data Sheet, *Melexis*, https://cdn-shop.adafruit.com/datasheets/US5881_rev007.pdf

[14] BMP 388, Data Sheet, *Mouser*, https://www.mouser.com/.pdfdocs/BST-BMP388-DS001-01.pdf

[15] Bluetooth® Technology Website. 2021. *Core Specification 4.0 | Bluetooth® Technology Website*. [Online] https://www.bluetooth.com/specifications/specs/core-specification/

APPENDIX

A.Design Alternatives

A different approach for this type of product could be to just use the GPS available on smartphones and a maps API to get distance and altitude change data over the course of a race and not bother with a physical system. While this would be cheaper up front maps APIs can become expensive and also lack the ability to be used on a stationary bike stand. Maps APIs also do not have the data to find more exact altitudes. For example, someone biking up a bridge on a bike path that goes above a highway will not have that reflected in their immediate scoring the way someone with our system would.

We had also considered different ways for alerting the users of their progress in the race/workout. We brainstormed mounting the phone itself to the handlebar and having the user see their

The power system also went through many revisions over time with different levels of features. Eventually we reached the simplified design in the final product.

B.Technical Standards

The Bluetooth standard used is maintained by Bluetooth Special Interest Group and is called the Bluetooth Core Specification 4.0, and our module specifically uses the Bluetooth Low Energy protocol from this specification. Bluetooth used to be maintained by IEEE but is now exclusively maintained by Bluetooth SIG.

The Swift programming language is a standard maintained by Apple. It is used for developing applications for iOS and macOS. Swift was used to develop the iOS app.

ANSI C++ was used to program the ATmega328P microcontroller. This is IEEE standard 60599.

C.Testing Methods

To test the Bluetooth connection between the microcontroller and the iOS app we tested initial connection to the board by turning the system off then back on multiple times and approaching the system with the iPhone and checking for a connection.We will also tested moving away from the system while running and testing reconnection. Live rides were tested for disconnections via checking the iOS app log files for missing or corrupt messages.

Extensive testing was be conducted to ensure the system is ride variation insensitive. We have tested the system to acquire the minimum speed for which the generator produces usable power from rotation. The indications of proper operation of the system was the LED being fully lit on the appropriate PCBAs. The components used are as follows:

- Measurement: tachometer to extract the linear speed from rotational speed
- Rotation Actuator: Drill to rapidly rotate the generator head
- Generator: Bike dynamo to convert rotational energy to

For testing the sensor system we stationed the wheel and spun it for 10 full rotations. We then compared the received count through the app to the expected count (which was 60, 10 rotations and 6 magnets per rotations). We repeated that test 10 times and got a range of 50-55 detected out of the 60 expected. This led to us being able to record distance with a minimum accuracy of 83.3% and a max of 91.6%. We didn't get a chance to test the BMP388 in the final system due to shortage of time for FPR. However, our initial testing that was done for MDR did prove that its accuracy of recording the change in altitude being with in ∓0.5m was indeed correct. We had a meter stick and as we rose in elevation it changed accordingly.

Much of the testing for the app was done manually as a lot of the backend (such as game center communication) is already handled by Apple. Each View Controller was tested with common cases that the users would perform. We also performed testing for matchmaking with multiple users along with how communication is handled between all users. Since most of the backend is handled by apple's servers, we did not need to write code for unit testing rather we did that part manually as specified.

D.Project Expenditures

The price per blank PCB with shipping from JLCPCB comes out to $3.23 per board. Our bill of materials per unit from Digikey was $62.57.

Our costs for prototyping materials we bought totaled $143.57, however we did use many parts from M5 which helped keep out project expenditure low.

E.Project Management

Xavier was the Team Coordinator, Syed was the Team Secretary, Benjamin was the Team Treasurer, and Ali was the Altium Lead. Xavier specializes in power systems, Benjamin specializes in system software design and verification/testing and systems engineering. Syed specializes in backend systems. Ali specializes in sensor systems. The specialties lined up nicely with the different subsystems of our project, and allowed for a very neat organization of work which made the transition to mostly online work much easier. We broke our project down into modular parts so each member could work mainly on our own system and integrate them with little issue, which we succeeded in when bringing the integrated system together.

F.Beyond the Classroom

Benjamin - I was the treasurer for this team and in charge of the microcontroller and Bluetooth communication. I also worked on the iOS app and assisted with the PCB design. The most important skill I worked on during this project was time management when there is few exact deadlines and the problem and planning was mainly up to me. While SDP did have some due dates, it was mostly up to me on how to use my time and break the project into pieces and assign priorities. Working remotely with a team was also useful experience. Doing my own research and reading many different data sheets to find the best options for our project and comparing options was also a useful skill to practice. These are all skills I would know will be very useful in my professional career.

Xavier - A key skill that I developed through this design project was the ability to communicate challenges as they affect product timeline. At the start of the spring semester, I was displaced from my home and have been displaced for its entirety. This caused several unforeseen challenges with accessing campus, designing and building a prototype.

Although it was challenging, I informed my advisor of the hardships I faced and he provided invaluable guidance on how to produce a working system given the unique circumstances.In this design project I learned the value of preliminary research. While presenting a system design in a team meeting with our advisor, I was shown that there were many critical characteristics of my design that were not addressed that could have been accounted for with deeper preliminary research. As I progressed in designing the system, I learned to be mindful of assumptions, justify engineering decisions and document the design process.

Ali - As the Altium lead and the sensing system engineer my job was coming up with data collection methods that gave us the information we needed to provide an accurate experience for the user. While some of our solutions were subjective (like using the BMP 388 to reward incline bikers). They achieved the end result of being able to reward the user in the way we wanted. I also designed, populated and tested both PCBs necessary for our project. When hiccups arised and we couldn't use the PCBs in the final project I was the one who soldered the ICs and necessary connections for the power system and the microcontroller for both protoboards. I'm sure the skills acquired regarding debugging and design related to PCB design as well as being able to quickly prototype will be helpful one day but I have yet to use these experiences outside of class. I am however grateful for them and SDP21 as I'm sure they will come in handy in my future graduate experience.

Syed - I was tasked with implementing the front end and the backend design of the IOS app. I oversaw the cloud infrastructure as well as implementing apple's gameKit API properly with our app so we could pass data between users in real time. I received help from Ben on the communication aspect of the app as well as calculating the game score aspect of the app. I think there were many aspects of real-world software engineering that I had to employ to get my work done. For one I learned that while preliminary design and research is very important it's also very important to be flexible. I realized this when my original plan was to implement all the matchmaking code by myself using firebase database to communicate between users. I soon realized that while I had accomplished matchmaking with strangers, it would be much more difficult to implement matchmaking with friends along with implementing the invitation handlers for such a task. I realized I would not have the time to have a fully well tested matchmaking system done and so I had to deviate and start overusing apple GameKit api. This ended up being a great decision as it handled matchmaking perfectly the way I had originally hoped for and since most people already use game center on IOS for multiplayer games, they would have friends in that system already. This made me realize the importance of figuring out how much work a task would take initially so that in the future I don't have to deviate halfway through the year. This also made me realize the importance of setting weekly goals for myself and a set schedule to work throughout the week. Since I was running behind, I set a 9-5 schedule for myself where I would get all the work, I had laid out done for the week and the rest of the time would spend working on the app regardless of how much I had gotten done that week. This fast tracked me in getting a fully working app done as I struggled especially during the first semester in setting time aside for working on SDP. Lastly, I really enjoyed working on something completely new and something I'd never actually done before and had experience with and realized how much I loved making apps and will most likely pursue this more in my free time after college as well. All in all, I learned and honed a lot of my skills that will be vital in my future success as a software engineer and working with Ben on the app developed my peer programming skills exponentially.