Converting a Normal Bicycle to an Electric Bicycle

Carlos Curbelo, Alexander Gilbert, Victor Petrillo, Christopher Rumpf

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

Abstract — The paper will show our progress in turning our bicycle into an electric bicycle while still following safety guidelines in all aspects of such as motor speed and light requirements. The paper will also present the hardware and software used in order to complete our goals of the conversion kit. The individual hardware components will be looked at as well as the software used. Then we will show what the project looks like as well as challenges that were faced and overcome during the creation of the project.

***Index terms* — Bicycles, Bluetooth, Brushless DC motors, Lithium batteries, Microcontrollers, Mobile applications.**

I. Introduction

The premise of the project is to modernize something that could easily be thrown away. By taking an old run-down bike and giving it a second chance; converting it into an electric bicycle (e-bike). Although, this idea is not new or groundbreaking, the skills and processes involved are a great way to culminate all the things learned over an Engineering bachelor's degree. Designing, integrating, and building multiple individual components that will be assembled into a useable e-bike.

Electric bicycles can be a solution for short to medium distance trips, for example, commuting to and from work, or transporting small objects, in particular: food, groceries, packages, etc. For much longer trips, with a larger cargo or even group travel, an e-bike may not be a feasible solution. In the United States, where this project is based, the average person may not be able to take advantage of the shorter trips an e-bike may be useful for. Although, as cities become denser and more self-contained, e-bikes could be used in everyday life and each trip could remove an internal combustion engine off the already crowded roads.

To achieve this, we will be using an old bicycle to convert into an e-bike. We will be creating our own custom printed circuit board (PCB) that will integrate many of the components. On the PCB will live the main microcontroller (MCU), inertial measurement unit (IMU), Bluetooth module, and various connections to peripherals. The peripherals being: motor driver, brakes, throttle, and lights. All of this will be powered by an external lithium battery. Also, a mobile application will act as a connected device that can be used to get data from the bike such as speed or battery life. For the project we set goals that we wanted to achieve such as throttle control, a functioning app for the bike, and to have controlled lighting of the bike.

II. Bicycle Safety/Standards

An electric bicycle has the following definition according to the 316.003 (23), Florida Statutes; "A bicycle or tricycle equipped with a fully operable pedals, a seat or saddle for the use of the rider, and an electric motor of less than 750 watts which meets the following requirements. A class 1 only provides assistance when pedaling up to 20 mph. A class 2 allows the rider to exclusively use the motor up to 20 mph while a class 3 is similar to the class 1, however, the speed is 28 mph instead [1]. For the bicycles in each class once the speed is reached the motor will no longer operate once the speed is reached.

In Florida e-bikes must also follow the same rules a regular bicycle must follow. Another Florida Statute 316.2065 (7), "Every bicycle in use between sunset and sunrise shall be equipped with a lamp on the front visible of at least 500 ft. … and on the rear visible from a distance of 600 ft. …" [2].

III. Project Concept

The goal of the project is to turn your average bicycle into an e-bike. We wanted to do this because we wanted to understand the process behind converting the bicycle but also the components that make up the e-bike. The parts, such as the motor, are things that we don't have a lot of experience with, and it would be interesting to have a better understanding of how this part works.

Some part of the project that we planned on doing but scrapped was regenerative braking. This was going to be a main goal, however, after speaking about it, it was decided it may be too difficult and out of the scope for what this project is supposed to be about. Other goals that we have that are not on the diagram as they are stretch goals is a form of cruise control and assist mode. The

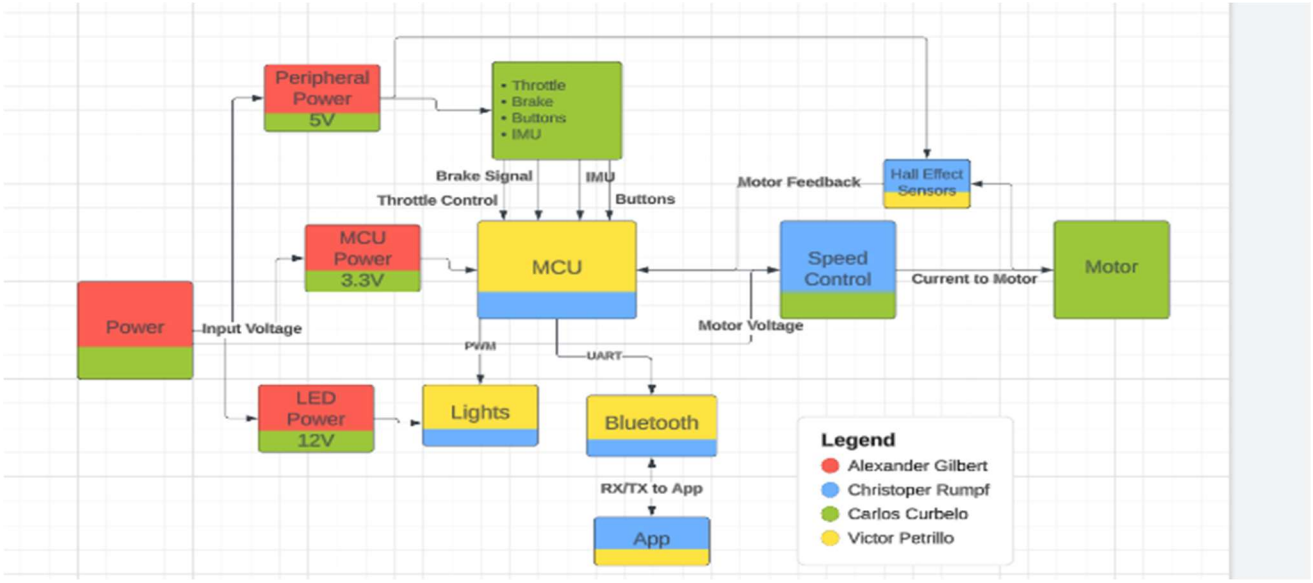

Figure 1: Block Diagram for the Project

cruise control would allow you to set the speed as you would on a car. The assist mode would be the same as what is stated in the safety standards where the motor would only help such as on an incline.

Some ideas we had that did not get past the initial phase of what our three main, advanced, and stretch goals would be tire pressure sensor, location logging, and a few others.

In the end our three main goals are an app, throttle control, and lighting. The advanced goals are cruise control and assist mode. The stretch goal is regenerative braking, however, that goal is too difficult for the scope of the project.

IV. System Components

In this section we will discuss our individual components and how they all fit together. We will cover why each component was selected, use in the project, and how it integrated into the overall project. Also, any changes or issues that happened along the way.

A. Motor

We made the decision to use a geared rear hub motor after conducting extensive research on the subject. The Bafang G310 Standard Wind (8.5 rpm/V) Geared Rear Hub Motor is the one we chose. The Bafang G310 motor has a nominal 250–350-watt power rating and weighs 2.5 kilograms. It functions effectively for hidden systems that do not require much help with incline or weight.

The motor is a brushless DC motor (BLDC) which uses a DC voltage input and a controller (discussed later) to turn the motor. The motor provides hall-effect sensor outputs to determine when the controller should switch its output to the next set of windings. This motor features an integrated speedometer which is another output that pulses 6 times per revolution. It employs a side cable exit and a free shaft to allow different kinds of bicycle cassettes. Internal to the motor is a 11:1 reduction ratio designed with spiral gears for the first stage and second stage planetary gear for low noise and higher torque for a small motor. This hub is suitable for 26” and 700c wheels; the bike we are converting has 26” wheels. The motor was purchased from an e-bike component website focusing on conversion kits. When purchasing the motor we can have the business assemble the motor to the bike tire rim for a fee.

B. Brakes

The original bike used rim brakes which we chose to keep for simplicity. The other option would be to use disk brakes, which can be better for braking, but can require complex mechanical changes and was outside the scope of this project. The pads were replaced and function in normal bike operation. The brake levers were replaced with Ebrake levers so when the brakes are pulled a signal will be sent that will tell the motor to shut off while also applying the normal rim brakes to slow the bike.

C. Microcontroller

There are many choices for MCUs on the market and there were many factors to consider in which MCU we picked, such as: availability, due to the current climate of global chip shortages, versatility, support, and new

environment to learn. In the end we chose one of STMicroelectronics line of ARM MCU in the STM32 family. This allowed us to explore a new type of chip that is outside of what the University taught and is a common MCU used in the industry. The STM32F405RG6 is the MCU we chose. It will be our main processor and will be communicating with all the other components on the bike.

The STM32F4 series is an ARM Cortex-M4 processor with fast speed and a wide range of peripherals that can be used for our project, particularly a sophisticated timer function that can output multiple pulse-width modulation (PWM) signals for motor control. It also has a fast speed and multiple common serial interfaces: universal asynchronous receiver-transmitter (UART) which is built on top of the RS-232 protocol, serial peripheral interface (SPI), and inter-integrated circuit (I2C). The MCUs physical chip size was the smallest pin count with 64 pins and was provided in a low-profile quad flat package (LQFP).

Software and programming of the MCU will be discussed in the software section.

D. Bluetooth

To communicate from our bike to a mobile application we will use a Bluetooth module. In looking for the most up to date and lowest power Bluetooth version we were going to use Silicon Labs BGM220PC22HNA2, which used Bluetooth v5.2. However, the usage of the module required additional programming and setup which proved to be difficult. This module is very flexible allowing multiple communication protocols and even allowing inputs/outputs (I/O) from the device itself. For simplicity we pivoted to using the DSD TECH, HM-10 Bluetooth module. This module uses a lower Bluetooth version v4.0 but is still using Bluetooth low energy (BLE). This module only uses UART communication but does not require any setup.

E. IMU

IMUs are widely available on the market being used in mobile phones and small devices. The goal of the IMU would be to gather positional data from the bike that could be used to detect impacts or if the bike were to fall over. A 9 degree of freedom (DOF) IMU with 3 axes accelerometer, gyroscope, and magnetometer is very common and affordable. Converting the IMU data into usable information is another factor. The steps to do this are not covered by this project. Fortunately, some IMUs come equipped with on-board microcontrollers that can perform the processing and offload it to the IMU rather than the primary MCU. The primary MCU can then retrieve this data from the IMU. The BNO085 IMU was selected since it uses a digital motion processor (DMP). As convenient as the DMP is advertised it does not allow for direct polling of data over the standard protocols. Instead, it uses a Sensor Hub Transport Protocol (SHTP) which requires every communication to be sent via packets and internal channels. Though libraries exist which abstract SHTP we were unable to find a library or create our own for the STM32 over the course of this project.

In the end we switched to the AltIMU-10 v4 which is a 10 DOF IMU by Pololu. The extra DOF is altitude. The IMU communicates over the I2C protocol and provides raw data which does not allow us to offload calculations from the MCU. Since this is not a main requirement, we will instead look for spikes in acceleration that could be indicative of an impact.

F. Motor controller

The motor controller is a Flipsky electric speed controller, FSESC4.20. On the module is a Texas Instruments motor driver for 3-phase brushless motors the DRV8302DCA. The motor driver has hall sensor inputs for proper switching of the motor. By using a PWM signal from the MCU we can drive the motor.

The PWM signal required to drive the motor has specific characteristics that are needed. It uses a signal with a time period of 20 ms, and the size of the duty cycle determines how fast the motor should go. The positive side of the signal must range from 0% to 10% and will be the range of the motor. This means that the PWM signal used to drive the motor will have a positive time ranging from 0 s to 2 ms. With these characteristics we can map the throttle to this range for controlling the speed of the motor.

G. Converters

There is a requirement for numerous power supply regulators for this project. The most typical voltages are 3.3V, 12V, and 5V. Our power source is a battery that has a nominal 36V DC. On the one hand, switching regulators are more complex but have higher efficiency than linear regulators, which are very basic but have low efficiency. For each of our main power voltages, we will select a switching regulator. Although there are numerous free tools available, such as WEBENCH Power Designer from TI, these power circuits can also be created manually. With the use of these tools, parameters can be utilized to select solutions with a balance of high efficiency, cheap cost, and compact footprints. After identifying each component's power requirements, we

can utilize a tool to design the circuitry for the main control board.

H. LEDs

The RGB LED Weatherproof Flexi-strip, purchased from Adafruit is one of several LED solutions available. We have the freedom to cut and reuse segments of LEDs at 10 segments per meter, and we have plenty of room in case any other components are harmed or break. This choice of lighting aids us in coming up with something fresh and unique for the project. With this light, it's possible that the app will allow us to modify the color of the strips. With this form of LED, we can also have a variety of design options because it can stick to any surface. Also, with the waterproof capabilities make this an overall better option for the design of the project. They will be PWM controlled by the MCU.

I. Switch

The switch has four buttons, an up down, a mode, and a power button. These buttons will provide a backup in case the app stops working for whatever reason and will cycle through different modes that we will predefine. The first mode will be throttle mode where the bicycle will use the motor to move and normal mode where the motor will not be engaged.

J. Throttle

The throttle of the bike is a knob that is placed on the right handlebar. Turning the knob toward the rider will be the signal that is sent to the motor driver to tell the motor how fast to go. The output of the throttle comes from an internal linear hall effect device that outputs a 1 – 4 V signal, when powered with 5 V. The MCU will read this analog signal and covert its digital value into a PWM signal for driving the motor, discussed previously.

K. Battery

Given that the motor's recommended maximum power rating is 700 Watts, but that nominal is likely to be between 250 and 350 Watts, a 36 V battery with a 9.5 A restriction would generate 342 Watts in an effort to keep below the maximum nominal power. We would maintain the amperage at roughly 14 A for a wattage of 336, which would be less than the 350 Watts if the battery outputs at 24 V. By limiting the amps, we can prevent the e-bike from exceeding the speed at which it would become a class three vehicle. The *36V 20Ah Unit Pack Power e-bike battery* is the option we have. If we limit the amperage to the 9.5 A that were calculated before, this would provide us with the voltage needed to power the motor and allow it to run constantly for about two hours. The battery will provide the motor with the energy it needs to run for two hours and, in addition, will enable the motor to move faster while climbing a slight slope.

V. Hardware

There are multiple objectives needed to accomplish the project of turning a bike into an e-bike for senior design. The two most important parts of the conversion being the battery and the motor. The motor is a Bafang G310 motor with a nominal 250 – 350 Watt power rating which keeps us within our electric bicycle classification. The recommended max power is 700 W which is still below the 750 W for e-bike classification. The battery is lithium ion with nominal 36 V with 20 Ah of electric charge. We expect approximately 2 hours of charge or around 30 miles on one charge.

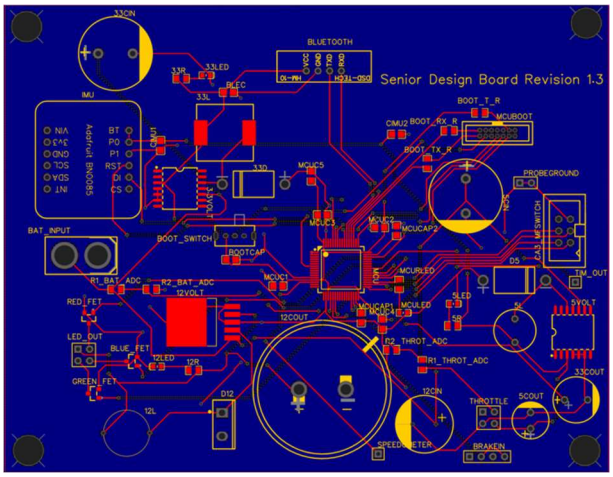

Figure 2: Newest Iteration of Project PCB

Above is the last iteration fo the PCB that we will have and we will use the reflow oven in order to put the surface mount components on.

The PCB we made will have converters to lower the input voltage from a nominal 36 V to 12 V, 5 V, and 3.3 V. There will be three major circuits for each voltage that was converted and a circuit for the 36 V from the battery.

The 12 V will be for the LED strips that we will have along the edge of the bicycle. The LED strips have an input voltage of 9-12 V and will use a PWM from the MCU to change the color of the LEDs.

The 5 V is for the throttle to allow throttle control of the motor which is one of the specifications to allow throttle control of the project. The IMU will also be on the 5v circuit. The IMU will connect to the MCU and provide orientation data to the MCU, so if the bicycle falls on the side, the motor will not be able to run even if the throttle is engaged.

The 3.3 V is what will power the MCU and the Bluetooth. The Bluetooth module will allow communication from the app to the board to send and receive instructions or data such as the battery percentage. The motor control will send the pulses required to control the motor.

The bicycle we are using is approximately 15 years old and is the second bike we are using. The first attempt had a frame that would be too small to fit the motor and wheel, so we needed to use a different frame. The new frame is wider and works much better. The PCB will go in a waterproof box mounted on the bike. The LEDs were cut into 4 segments. The first 3 will be on the front of the bicycle where each one is 100 mm then the remaining 700 mm will be wrapped around the back of the bicycle below the battery adhered to the wood blocks.

VI. Software

In this section we will discuss the software, code, and tools used on the e-bike. There are two main parts of software for the project, the MCU and the mobile application.

A. MCU Software

For programming the MCU there are many languages and options. We picked what we believed to be a standard default for getting started in the STM32 development environment. We will be using STMicroelectronics integrated development environment (IDE) STM32CubeIDE which allows us to write code in C, use an interface to pick pins and peripherals, and access to multiple libraries such as their hardware abstraction layer (HAL) drivers and real-time operating system (RTOS) specifically FreeRTOS. Since actions in this project need to happen in real-time using a more common MCU program structure like a super loop could cause unnecessary slowdowns. The use of FreeRTOS allows us to break up the code into tasks and give each task a priority. Then the scheduler will handle when each task is allocated to the processor.

The task priorities from highest to lowest are: Realtime, High, AboveNormal, Normal, BelowNormal, and Low. We could have multiple tasks in each priority but chose to split them up in case the scheduler starved a task from never running. The tasks we created in order from highest priority to lowest are: motor control - Realtime, Bluetooth communication - High, IMU and battery readings - AboveNormal, and LED control - Normal. To read the motor's speed we chose to use an interrupt service routine (ISR) which will happen outside of the scheduled tasks and only activate when triggered.

The motor control task will read the throttle and convert the analog value into a PWM signal to be sent to the motor controller. The Bluetooth communication task is focused on transmitting data to the mobile application. The data that is sent is sent as repeated strings that are comma delimited for example: "battery: 50, speed: 7". The MCU will calculate the speed in miles per hour (mph) and the battery voltage will be read and converted into a percentage from 0 – 100 sent as an integer. This packet can add more values to be sent to the app by adding more data to the end of the string. IMU data could be sent as pitch, roll, and yaw angles or acceleration data in the x, y, and z plane.

For receiving data from the Bluetooth module instead of polling for data from the app we were able to use another function of the STM32 MCU by use of direct memory allocation (DMA). The STM32 has an internal DMA module which can be setup to connect different peripherals together or send data from a peripheral straight to memory. This is another method we used to take less load off the processor and increase speed.

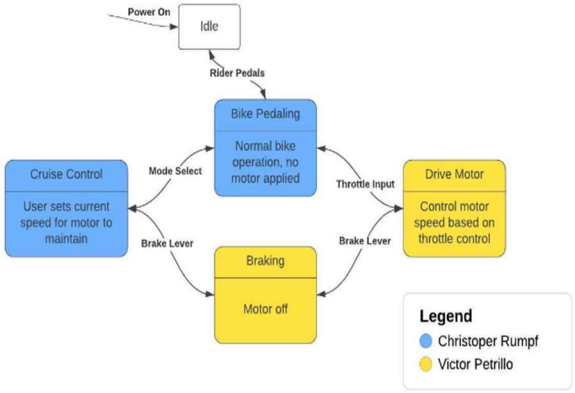

Figure 3: Block Diagram of the Controller States

Unfortunately, we had issues in receiving full strings from the Bluetooth module and to save time were only able to accept single characters. We chose to create our own encoding scheme which would allow us to transmit all relevant information from the app in a single 8-bit character. The information received from the app is broken into: mode, red, green, and blue values. The mode states we chose had 3 possibilities: idle – no motor running, throttle – using the throttle for motor control, and cruise – set a speed for the motor with no active throttle input. Due to time constraints, we were unable to implement the cruise control mode. All possible mode states can be covered in 2-bits allow us 4 total possibilities. Since we are only using 3 states, we could allow for another mode in the future that some e-bikes

provide, which is pedal assist. Red, green, and blue values are the colors for the lights to be controlled; each color value received is also allotted 2-bits. This does inhibit us from allowing the very common full RGB color scheme of 24 bits with 256 possibilities per color for a total of 16,777,216 possible colors. Instead, this allows us 6 bits with 4 values per color for a total of 64 possible colors. We believe this still gives us enough flexibility in color control and is all hinged on a constraint of receiving a single character vis Bluetooth. This flaw could be worked out in the future.

Reading data from the IMU comes via the I2C protocol. We periodically poll the IMU for what it is currently reading and store the most recent data looking for spikes in acceleration. Then the battery voltage is sent through a voltage divider circuit turning the maximum battery voltage of 42V down to a readable analog value by the MCU at 3.3V. Finally, controlling the LEDs is done by sending PWM signals to each of the 3 channels red, green, and blue.

The above tasks will all be running and handled by the FreeRTOS scheduler. At the same time, the motor outputs a speed value given by 6 pulses per revolution. We will use an ISR on this output measuring time between pulses as the current speed of the motor. We will need to monitor this speed value to ensure the bike remains under the required 20 mph. Also, the brakes on the bike are connected to the brake input of the internal MCU timer which will shut the motor output off whenever the contact is closed. When switching between throttle mode and the other modes the output will be disabled; ensuring the motor will not run outside its defined time.

B. Mobile Application Software

The development of the app is being done with Flutter, a cross-platform development software used to develop apps for devices running both iOS and Android operating systems. To write code and implement testing, Virtual Studio Code has been used for not only its base capabilities to support many different languages, but also its extensive third-party extension library for additional features.

Flutter, when installed on a machine and before development can begin, must have the appropriate technologies to support it. One technology required is Android Studio, which allows Flutter to compile and execute the application on Android devices. Android Studio also includes an emulation feature, which VSCode can recognize and utilize to execute the application on emulators. These emulators can run on the device being used for development.

While Flutter is cross platform and can compile the written code for both platforms, Apple's Xcode IDE is required to do so for iOS devices. Unfortunately, Xcode does not run natively on Windows machines. Our group does have access to multiple Apple computers running appropriate versions of macOS that can run Xcode. With these, we can compile the application for iOS devices. A fortunate feature of a macOS computer is that iOS emulators are installed on the device already and are configurable through Xcode.

Due to the group having access to both technologies required, them being Android Studio and Xcode, the application's development has gone smoothly. The application is Bluetooth-enabled and can connect to, transmit, and receive information from the Bluetooth module we are using for the e-bike. Testing the application on an emulator is great but is incapable of testing Bluetooth functionalities since it is not a physical device with unique identifiers. Thankfully, Flutter is also capable of executing the application on a connected physical device as well. With iOS devices, it must be plugged into the computer to install the application on the device. Android devices can connect to the computer via Wi-Fi and can receive the application over that connection. With the application on physical devices, Bluetooth functionalities were able to be tested.

Initially, the application itself was prototyped to be somewhat modular, with a home page that can bring the user to certain pages that hold certain features. This modularity was traded for a more linear experience to streamline development. When the user opens the application, they will be brought to a warning screen, warning them to please not use the application when riding the bike, as that can lead to distracted operation. From there, the user will be brought to the Bluetooth connection screen; the application must be connected to the e-bike to access the features. From there, the user can access the heads-up-display screen, which will receive information being sent from the e-bike. This information includes, but is not limited to, current speed, battery charge, and bike orientation.

As mentioned previously in the MCU Software section, the Bluetooth module is only capable of receiving a single character. Although this initially seemed to be an issue, the group was able to engineer a way around it; we are using a single character (1 byte/8 bits) to transmit enough information to satisfy our requirements. As also mentioned previously, time constraints prevented us from implementing a cruise control feature. However, due to the nature of the byte encoding, there exists enough overhead in the byte

encoding to allows us to implement this feature in the future, if so desired.

Below is the updated wire flow diagram for the application, depicting the experience.

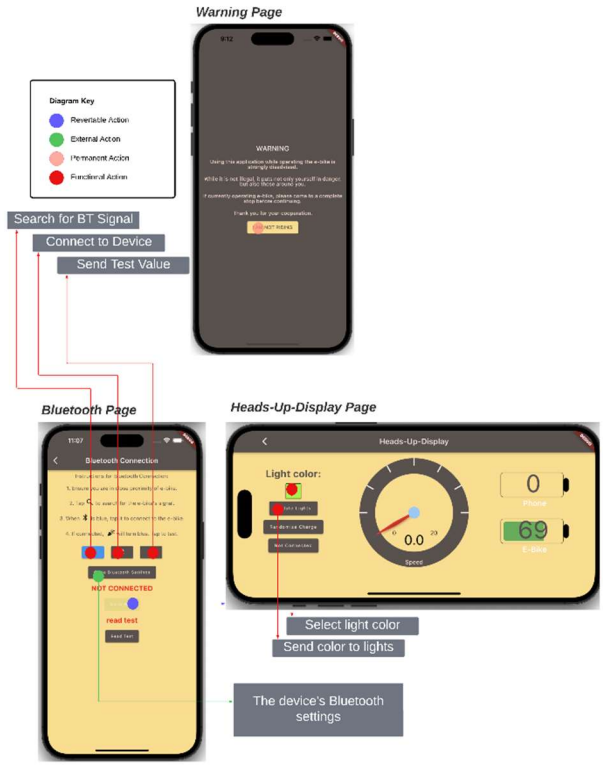

Figure 4: Application Wireflow Diagram

VII. CHALLENGES & TESTING

A. Challenges

The first challenge that was encountered was the bicycle not having enough space in the rear for the new tire with motor to fit well. The back was much too narrow that while it was possible to put the new tire on it would take more effort than normal to put the tire on. In order to get around this issue we used a different bicycle that Victor provided that has a wider back and could fit the tire much more easily.

The second challenge was with the Bluetooth library we were using in Flutter. Initially, we were unable to transmit information from the application to the Bluetooth module. This seemed to be an operating system issue at first, but it turns out the incorrect form of the device's identifiers were being used. To fix this, we used the correct identifiers. The Bluetooth connection is now a complete, two-way connection.

Figure 5: Front half of the prototype

The third challenge we encountered was the design aspect of the bike since the new frame was a bit smaller. We had to come up with a way to mount the power supply and PCB to the bike. The best solution we found for this was to use a rear bike rack that is capable of holding the weight of the components.

A fourth challenge we encountered had to do with the parts that were not in stock. Some parts such as the

Figure 6: Back half of the prototype

2n7002 MOSFETs were out of stock or has a lead time of a few months, for example, some components were not expected to come until the end of August so we had to use a different supplier or a different component all together.

B. Testing

So far we have tested that the motor and battery work and we are able to control the motor with the throttle. The ebrakes we have also work when testing. Currently the PCB is able to convert 30v (the max voltage output of the power supplies in the lab) to approximately 3.28v.

As seen in Figure 4, we can see how we were able to get the power supply onto the bike through the use of a back rear rake. It is able to withstand the weight of the battery and keep it steady so there is no movement while riding. This helped us in figuring out how we can mount he lights on the bike. In the figure there is a platform created by wood to elevate the power supply and that was able to give the LEDS something to adhere to in order to create the rear lights that we wanted. In Figure 5 we essentially are showing the newer iteration of the throttle, brakes and the switch we will be using for the bike. All these components are electric and connect to the PCB directly. Also its not seen be we were able to cut the LED and solder extensions in order to have then run along the bike. Also, in the image there is a black box seen on the handle bars which is the phone mount we will be using.

VIII. Conclusion

In essence, the goal of this project is to develop an e-bike with a design that is both practical and interesting. As previously said, whether traveling to work, school, or the grocery store, people nearly always need to commute by car. With an e-bike, the user has additional freedom. Compared to electric cars, an e-bike is a more practical and economical alternative form of transportation in urban areas. More people can ride an e-bike because it is more affordable than a car compared to its cost. Even while that might be the case, we would still have to overcome the challenge of producing the finest design at the lowest cost.

We think we can accomplish this thanks to the numerous team meetings that assisted in choosing each component and each design concept. Additionally, one of our group members had some prior experience working on projects of this nature, and one of the committee members could be regarded as a bicycle enthusiast who provided us with some excellent feedback. Many of the decisions we made regarding part selection took into account all viewpoints, but we were careful to ensure that for some parts, performance was prioritized over cost. Because we have chosen components that are renowned for their compatibility with one another, we hope that the project will offer a simple implementation overall. By doing this, we hope to increase the effectiveness of our design throughout the building process. Soldering and mounting procedures will be completed during the integration phase using both domestic resources and the UCF laboratories. We will consider our alternatives and determine what stage we are in before deciding whether to 3D print or potentially buy an enclosure for the PCB.

In terms of software, our two CpEs have gone above and beyond to ensure that we are procuring the best parts for the PCB and the creation of the app. We currently have a functional app that can display our test cases, such as altering the color and showing a random number. The program has also been opened and tested on emulated versions of Apple and Android devices as well as a real Apple device. To successfully accomplish this project, a large amount of teamwork is required. To keep a team functioning efficiently, a project management strategy must be in place. Although no project management strategy is perfect, adhering to a timeline may help to avoid future issues. We have taken all necessary steps to follow the milestone chart from Senior Design One, and we anticipate producing results that are similar in Senior Design Two. We are confident that our design will be entirely functional, suitable for the majority of riders, and an excellent final product.

Engineers Introduction

Alexander Gilbert will graduate from the University of Central Florida in August 2023 with a Bachelor's in Electrical Engineering and plans to go back to Colorado to get a masters and/or a job.

Carlos Curbelo will graduate from the University of Central Florida in the Summer of 2023 with a Bachelor's in Electrical Engineering. Has plans on obtaining an internship/job upon graduating.

Christopher Rumpf will graduate from the University of Central Florida in the Summer of 2023 with a Bachelor's in Computer Engineering and plans on beginning his career with working at Lockheed Martin.

Victor Petrillo will graduate from the University of Central Florida in the Summer of 2023 with a Bachelor's in Computer Engineering and plans on working for Disney as a Systems Engineer.

REFERENCES

[1] "The Florida Senate." *Chapter 316 Section 003 - 2022 Florida Statutes - The Florida Senate*, 2022, www.flsenate.gov/Laws/Statutes/2022/0316.003.

[2] "The Florida Senate." *Chapter 316 Section 2065 - 2022 Florida Statutes - The Florida Senate*, 2022, https://www.flsenate.gov/Laws/Statutes/2022/316.2065.