Bike Dash

Vincent Altavlla, Jose Davila

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

***Abstract* — The objective of this project is to design and implement a fitness tracker for outdoor bicycle riding. Bike Dash utilizes external sensors for data acquisition and displays them on an Android device through a Bluetooth connection. Riders are able to see both their real-time statistics, as well as performance on previous rides. With the added ability to "share" a ride with friends, it is our hope that Bike Dash will encourage and challenge riders to exercise more frequently. All data is received and relayed through an onboard microcontroller and all components (with the exception of the Android device) are powered by a dynamo that utilizes momentum to provide power**

***Index Terms* — Bluetooth, Global Positioning System, Power Generation, Microcontrollers, Data Acquisition**

I. Introduction

Bike Dash is a product that enables its users to monitor and record progress made while riding their bicycles. The system utilizes external sensors paired with an MSP430 microcontroller to acquire real-time data. During the ride the user will sync their data via Bluetooth to their Android enabled device. This will make the data viewable in a rich GUI layout and will also store the data on the phone. Storing the data on the phone will allow the user to retrieve previous data and compare to current data for route analysis. The application will take in data such as GPS coordinates, two-dimensional acceleration, and ambient temperature and translate it into user-friendly data such as speed, heading, calories burned, outside temperature, etc. To further entice users to utilize Bike Dash, it also offers a map view as a key component. While in the map view, users are able to see their progress on a map enabling them to gain a visual of their traveled path. Though the map view requires users to use a large portion of their screen, it still offers the ability to see real-time ride data.

In an effort to push forward with green energy, Bike Dash was designed to generate its own power using a battery only as a way to provide steady power. To power our system, a front-wheel hub dynamo is used to convert the bike's momentum into useable electrical energy. Producing roughly 6V (3W), the dynamo charges a LiPo/LiIon battery that provides a steady 3.3V source to power the main Bike Dash unit.

II. Data Acquisition

The Bike Dash system utilizes three key pieces of data to function and perform as designed. The three necessary pieces of data are GPS coordinates, two-dimensional acceleration, and ambient temperature. A schematic of the main module, which houses all of our data acquisition devices, can be seen below in Figure 1.

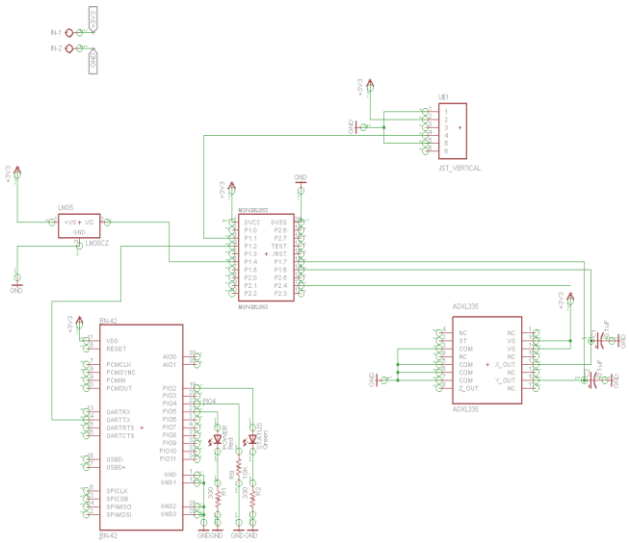


Figure 1 - Main Module Schematic

A. GPS

The GPS unit used in this project is USGlobalSat's EM-506, a 48 channel GPS module that includes a built-in antenna. In the Bike Dash project, the GPS unit is used to acquire GPS coordinates and relay GPRMC sentences to our microcontroller. To receive the proper NMEA sentences, the TX pin of the GPS unit is connected to the RX pin of the MSP430, creating a UART connection and allowing data to flow asynchronously. Once data is received by the microcontroller, it is relayed out to the Android device through a Bluetooth transmitter for further parsing. The sentences passed to the Android device are used to calculate the users speed, heading, and current location. GPS coordinates are also used to build the map view of the application. Features of the EM-506 GPS unit are outline below in Table 1.

EM-506	
Supply Voltage	3.3V
Channels	48
Time to First Fix (TTFF)	< 35s (Cold); < 1s (Hot)
Update Rate	1 Hz
Position Accuracy	< 2.5m
Velocity Accuracy	< 0.01 m/s (Speed); < 0.01 degrees(Heading)

Table 1 – EM-506 Characteristics

B. Accelerometer

The accelerometer used in Bike Dash is Analog Device's ADXL335. This device provide a three dimensional measurement of acceleration, though we will only use two for our application, and operates at the necessary 3.3V. The ADXL335 is read by connecting two ADC channels on the MSP430 (one for the X-axis and one for the Y-axis). The accelerometer provides results in terms of a voltage measured on the channel in mV and we use the voltage differences to calculate the force and angle. These measurements are used to determine if the user has been involved in a collision and will ultimately alert the appropriate parties of such an incident. The features of the ADXL335 are outlined in the table below (Table 2).

ADXL335	
Supply Voltage	3.3V
X-axis 0g Voltage	~1.5V
Y-axis 0g Voltage	~1.5V
Sensitivity	300mV/g

Table 2 - ADXL335 Characteristics

C. Temperature

The temperature sensor we are using in the Bike Dash system is Texas Instrument's LM35. The LM35 is a precision centigrade temperature sensor and has an output that is linearly proportional to the Centigrade temperature, as seen in Figure 1. The sensor also offers low power consumption capabilities and operates at an absolute maximum voltage of 35 V. To measure the ambient temperature, we use an ADC channel of the MSP430 to capture the voltage at the channel after applying a ~3.3V input voltage. This voltage reading is then transmitted to the Android device via a Bluetooth link where it is converted into degrees Celsius and Fahrenheit.

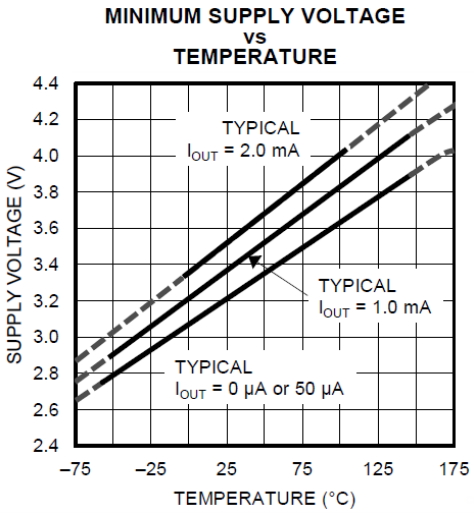


Figure 2 - LM35 Voltage vs. Temperature

D. Microcontroller

The microcontroller used in the Bike Dash project is the MSP430G2553. This microcontroller is used to collect data from all of our sensors and relay the information to the Android device for both processing and display. The MCU will be responsible for providing data both accurately and efficiently and must do so under stressful environmental conditions as this device will be subject to moderate environmental conditions. The MSP430 reads in data using a combination of ADC (Analog to Digital Conversion) channels and its RX and TX UART channels. Data from the GPS unit is collected at the receive buffer and forwarded through to the transmit buffer. As data is received at the transmit buffer, it is passed to the Bluetooth module for transmission to the Android device. The characteristics of the MSP430 in our application can be seen below in Table 3.

MSP430G2553	
Operating Voltage	3.3V
Speed	1 MHz
RAM	512 bytes
Memory	16 KB
Pins	20

Table 3 - MSP430G2553 Characteristics

III. Power System

The Bike Dash project was intended to be self-sustainable upon original design and thus empowered the Bike Dash team to utilize a hub dynamo to power the microcontroller and sensors. The hub dynamo is affixed to the front wheel of the bicycle and attaches to the charging circuit. At the charging circuit, the 6V (3W) output from the dynamo is utilized by charging a Lithium Ion battery at 4.2V. This 4.2V output is then regulated to 3.3V, a suitable voltage for the main unit. The characteristics for the power system can be seen below in Table 4.

Power System Characteristics	
Dynamo Out	~6.0V (3W)
Charge Output	~0V – 4.2V
Battery Discharge	~3.7V
Regulated Output	~3.28V

Table 4 - Power System Characteristics

A. Dynamo

Bike Dash utilizes a Shimano DH-3N80 hub type dynamo utilizing a rider's momentum as energy and converting it into a useable voltage. This dynamo is rated at 6V (3W) and is front wheel hub type dynamo to be used with rim brakes only. The dynamo is lightweight, coming in at just under 18 ounces, and is rated to be weather resistant. The dynamo is connected by two leads and a voltage ~6V is passed to our charging circuit. The voltage produced by the dynamo is a DC voltage and therefore does not need to undergo any AC/DC conversions. Because the dynamo cannot always output a continuous 6V, the addition of a battery to supplement power was implemented.

B. Charging Circuit

The charging circuit design in this application is crucial to the efficiency of the system. With that in mind, we designed a system that would offer characteristics that most closely matched our needs. The first step in the charging circuit is a charge management controller, the MCP73831. This controller takes in a maximum voltage and outputs a selected voltage. For our application, we have opted to output 4.2V as this is the optimum charge voltage for our battery. While in use, the battery will provide anywhere from 3.6V to 4.2V, which is above the minimum voltage of 1.8V required by the voltage regulator, and above our desired output voltage of 3.3V. The battery discharges a voltage in the range mentioned above and is regulated to continuous output voltage of 3.3V, the required input voltage of our microprocessor. A schematic of the charging circuit can be seen below in Figure 3.

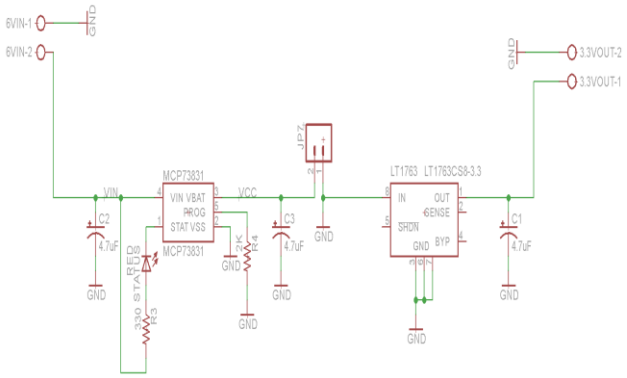


Figure 3 - Charging System Schematic

C. Battery

The battery choice plays an important role in the design of our project. It will control all the power created by the generator and supply it to our temperature sensor, Accelerometer, GPS, and Microcontroller. Because the battery is going to be mounted on the bike, which will be ridden outdoors, the weight, size and durability are important restrictions to the project. To account for this, a battery with a small form factor was selected and used in this project. The battery being used is a 3.7V lithium-ion battery that holds a charge of 1000mAh. To make efficient use of the battery, the battery is connected to the output of the charge controller as well as the input to the voltage regulator. This will ensure that we are always receiving a voltage source and that the batter will be under a constant charge. Because of the nature of the circuit,

and the decay of batteries under constant charge, we have made sure to setup our charging circuit to switch into a trickle charge mode when the battery is providing enough voltage. Ideally, the battery should remain near full charge during use of the device. This is intended as the battery acts primarily as a means to keep the system live during a crash or during stoppage periods during the ride.

IV. Android Application

The android application was created utilizing the Eclipse IDE along with the Android ADT/SDK plug-in. The application is limited to devices that support Android API 13 (Honeycomb) through the newest Android API 19 (KitKat) due to the fact that those APIs support version 2 of Google Maps. The android devices used for testing were the LG L9 and the Samsung Galaxy S4 which are equipped with Android API 4.1 (Jellybean) and the Note 3 which is equipped with Android API 4.4 (KitKat).

The android application will serve as the user interface that we provide to the user. The android application will receive data from the Bike Dash components via Bluetooth and use the data to provide the user with a rich GUI layout. In order to provide an optimal user experience we will provide the user with two ways of viewing the workout data collected by the sensors of the system. The primary option will be a map view which will provide the user with real-time visual representation of their current location, route and some important statistical data pertaining to their current workout such as time elapsed, distance, speed and ambient temperature in both Fahrenheit and Celsius representations. The map view was designed utilizing Google Maps APIv2 and required the registration of our Android Application officially through the Google Developer Console and obtaining an API key in order to receive support from the Google servers for displaying the map. Figure 4 shows the actual map view presented to the user.

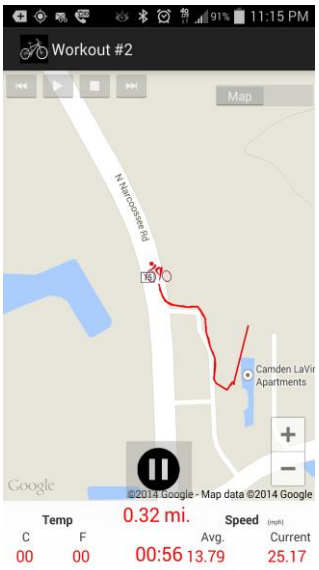


Figure 4—Android App Map View

The secondary option will provide the user with far more statistical data about their current workout. Along with some of the data present in the map view, the statistical view will also boast the calories burned, the pace in minutes per mile and the maximum and average value for each category. Figure 5 shows the statistical view that will be presented to the user.

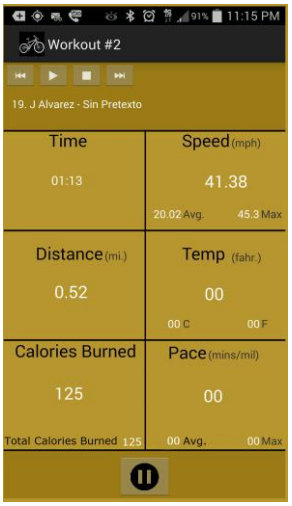


Figure 5—Android App Stats View

With the android application we aimed to provide the user with motivation to improve their workout experience and allow them to workout longer and more often. Therefore the user interface includes an essential necessity for every workout, a music player. The application searches for all media files with the extensions MP3 and WMA located on

the SD card of the Android device; it retrieves them and stores them in an array of songs. This then allows the user to control the music stored on their phone directly from the application without ever having to navigate outside of the application. Figure 4 and 5 shows the music player control buttons on the top-left corner of the screen. Furthermore we provide the user with the ability to share their workouts with their friends via text message. The application accesses the contacts that are stored in the phone (SD card, phone storage, google contacts, etc.) and turns it into a list displaying the friends name and phone number. In which the user can then choose which friend they would like to "challenge" to beat their workout time or distance.

In order to improve usability we modeled the application to mimic popular applications such as Twitter. Allowing the user to easily navigate the application and access the wanted information with a small number of clicks. Figure 6 shows the main layout of our application design.

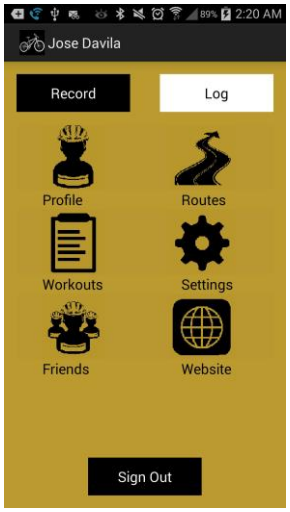


Figure 6—Android App Main Layout

The main application is made up of three buttons and six image buttons which are all clickable. When the user clicks any of the buttons the user will be taken to a different part of the application. From the main application, when the user clicks the profile image they will be able to view their profile which displays information about the user that was gathered from the sign up activity such as full name, email, height and weight. While also allowing access to the user's last workout data along with the overall total distance, workout time and calories burned. As mentioned before the data that's received via Bluetooth will be stored in the phone to later be used to access pass workout data and routes that the user may save. When the user chooses the workout image he will be brought to another activity of the application in which they will be able to view workout data and will have the ability to text that information to their friends.

The user also has an option to log their workouts into the system to allow for the application to act as a journal for their bicycle ride outside of Bike Dash. Another important design decision made for the user interface was the design of the route list. The route list was design to give the user visual feedback of recent routes, giving the user a map that plots the location points captured by the GPS sensor during that particular workout. Figure 7 shows the layout of the activity, in the figure the green represents the starting point and the red represents the ending point of the workout.

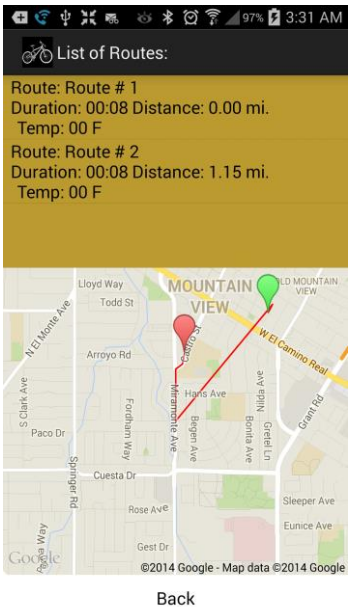


Figure 7—Android App Route List

V. COMMUNICATION

In order to establish communication between the Android device and the onboard components we decided to use the RN42 Bluetooth SMD Module from Roving Networks. This Bluetooth module is a powerful and easy to use module that boasts a small footprint. For this project the Bluetooth module will be used to replace serial cables in order to transfer data from the MSP430 to the Android device which utilizes its built-in Bluetooth module to be paired with the RN42. This Bluetooth module also contains and embedded Bluetooth stack which allowed for and easy integration into our system. The data to be passed from the microcontroller will be place in the RX

buffer of the Bluetooth module which will then put that data into the TX buffer. The data is then sent to the Android device by utilizing a simple UART interface with a data rate of 1Hz. The characteristics of the RN42 Bluetooth module can be seen below.

RN42 Bluetooth	
Operating Voltage	3.3v
UART Data Rate	1 Hz
Antenna	On-board

Table 4--Bluetooth characteristics

The RN42 Bluetooth module is in slave mode therefore the android phone will serve as the client and initiate the Bluetooth connection. In order to allow the application to run smoothly once connected the Bluetooth connection reminds active until the entire system or Android device are powered off. In order to retrieve the data being sent we created a foreground thread that listens for incoming messages continuously, when data is received the message is stored in a string builder object in order to be parsed out and to retrieve the usable data. An example of the format of the incoming messages is shown below.

$TEMP73#$AXLX456@$AXLY520%$GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62~

This string will be stored in a string builder then the end symbols (#, @, %, and ~) will be used to signify the end of that particular chunk of data. The temperature value signifies a voltage which is extracted and used to calculate As mentioned before a StringBuilder object was utilized to parse the data received from the RN42 Bluetooth module. Then data is then converted and stored on the phone to be used by the application in order to display a readable representation for the user. Figure 8 shows the part of the code to extract temperature data from the string builder.

```
endOfTemp = sb.indexOf("#");
        if( endOfTemp > 0){
        // Extract temp string
        tempData = sb.substring(0, endOfTemp);
        tempDataDisplay = tempData.substring(5,
endOfTemp);
        // Display temperature in celsius
                tempDataDisplayC =
        tempToCelsiusString(tempDataDisplay);
        temperatureDataCelsius.setText(tempDataDispla
yC);
        // Display temperature in fahr.
        tempDataDisplayF =
        tempToFahrString(tempDataDisplayC);

        temperatureDataFahr.setText(tempDataDisplayF

        //Delete
        sb.delete(0, endOfTemp +1);
}
```

VI. Board Design

To design the boards for both the charging circuit and the mainboard, we used Cadsoft's Eagle software to first create the schematics shown previously (Figures 1 and 3). We designed the boards separately to avoid any interference that may occur from the power system. It also allowed for boards with smaller surface areas allowing us more options for mounting. The boards designed are two-layer boards with the main components affixed via surface mounting. The PCBs were created using Eagle as well, which transforms a schematic into a PCB layout. The parts were laid out by hand, but the board routing was configured using Eagle's "Auto Route" feature. Once we had our board design laid out, the layout was exported as Gerber and then submitted to FreeDFM.com for verification of design for manufacturability. Upon receiving satisfactory results, the board was ordered as shown.

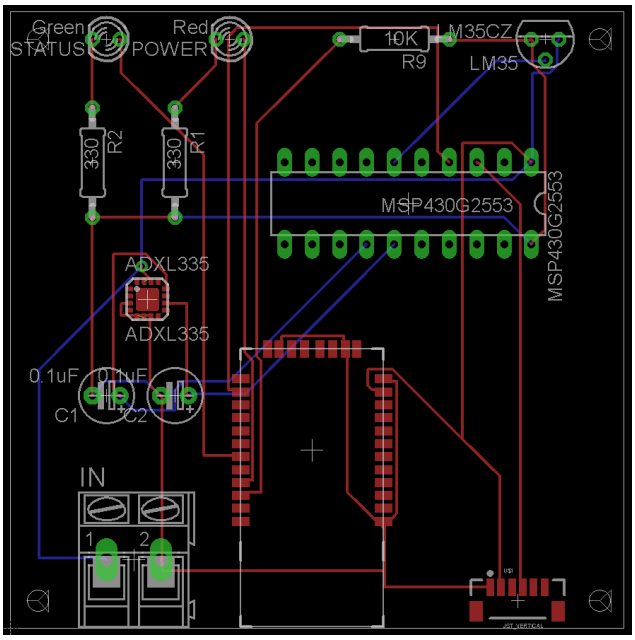


Figure 4 - Bike Dash PCB Layout

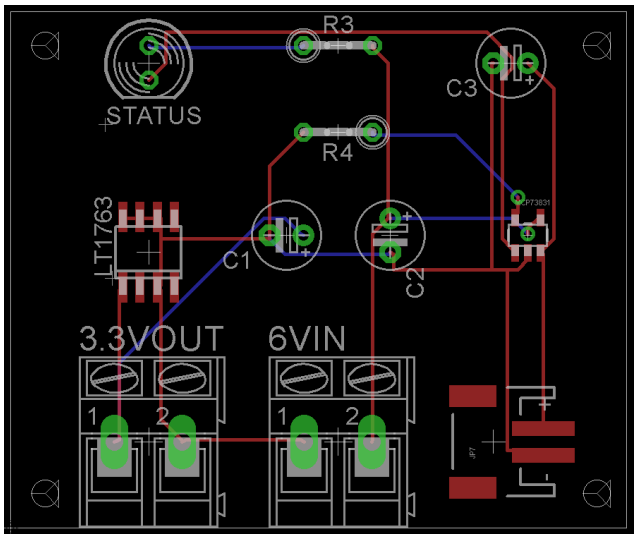


Figure 5 - Charging Circuit PCB Layout

VII. Mounting and Assembly

Mounting and assembly was done mostly by the Bike Dash team with the exception of surface mounting some components to our PCBs and the lacing of the rim as both these processes are extremely delicate.

A. Boards

When mounting our boards, we had to consider many factors. First and foremost was the installation of the boards in a way that would not interfere with our bike's daily operation nor interfere with the rider's normal movement and range of motion. Another important consideration when mounting our device was the operation of the accelerometer. Due to the nature of the measurement techniques, it was necessary to mount our device in a way that would ensure our coordinate system was maintained.

B. Dynamo

With the dynamo being a 32 spoke dynamo and our bike having a 36 spoke rim, it was necessary to build a custom rim for our bicycle. The rim was built using a WTB 19mm SpeedDisc rim. The rim was taken to a local shop where it was fitted with custom length spokes to match the dynamo hub. Once the dynamo was mounted on the rim, the wheel was affixed to the bicycle using a quick release nut system. This will allow the user to interchange rims giving them the opportunity to ride with or without the dynamo. This is beneficial in instances where riders may desire a higher performance wheel for racing, touring, or the like

VIII. Conclusion

Our end goal with this project was to offer an attachment to a bicycle that would give the user feedback on their ride. It was intended that we would appeal to riders of all skill levels and encourage the population to become more active in their life. It would allow users to count calories better, encourage improvements, and empower users to get out and exercise. Bike Dash was designed to be light and portable so that it would not impede the riders' normal range of motion and travel. Bike Dash was also designed to generate its own power using a battery only as a way to provide steady power providing the user with a green friendly, efficient product. With the utilization of the external sensors for data acquisition we were able to design and build an elegant user-end application that will allow the user to engage in a positive workout experience that can be shared with friends or used to challenge the user to personally achieve better performance.

IX. Acknowledgements

Bike Dash would like to recognize many people who became an integral part of the project. First and foremost we would like to thank our sponsor, Duke Energy. Without them this project would not have been possible. Furthermore we would like to thank the friends and family of both Vince and Jose as their continued support has motivated them to push through to the end. Also, we would like to thank Dan Predko of Cubic Simulation Systems for surface mounting our parts and giving us a "how-to" for soldering, De'Marion Robinson for chauffeuring the Bike Dash team during testing, David's World Cycle for providing an excellent rim build, and the University of Central Florida for providing a lab environment to develop and test our system. Finally, the members of Bike Dash would like to sincerely thank Dr. Samuel Richie for giving them an exemplary Senior Design experience. We hope that he has enjoyed his time here at UCF and wish him the best of luck in all his future endeavors.

X. References

[1] Google Developers—Google Maps APIv2. https://developers.google.com/maps/documentation/android/. N.p., n.d, Web. 3 Apr. 2014

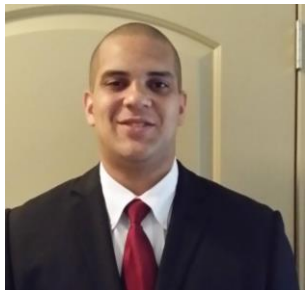
[2] "ArduDroid: A Simple 2-Way Bluetooth-based Android Controller for Arduino." http://www.techbitar.com/ardudroid-simple-bluetooth-control-for-arduino-and-android.html. N.p, n.d, Web 3 Apr. 2014

[3] Android Developers—Android 4.4 APIs. https://developer.android.com/about/versions/android-4.4.html. N.p., n.d., Web 3 Apr. 2014

X. Biography

These are the two engineers from the University of Central Florida who designed, manufactured, and built Bike Dash.

This is Vincent Altavilla. He will be graduating in May of 2014 from the University of Central Florida with a Bachelor's degree in Computer Engineering. He is currently working as an intern for Cubic Corporation and will most likely continue a full-time career with them as an engineer after graduation. His focus is primarily in embedded systems as well as simulation systems.



This is Jose Davila. He will be graduating in May of 2014 from the University of Central Florida with a Bachelor's degree in Computer Engineering. He is currently working as an Installation Technician for Control Designer, Inc. After graduation he will be seeking a full-time career as a software engineer. His focus is primarily in software and mobile applications development as well as gaming