University of Pittsburgh at Johnstown

Department of Electrical and Computer Engineering

EE 1195 / COE 1195 Engineering Practice and Professional Development

Final Report

Cycle Statistic Tool

Date: April 24th, 2021

Jacob Hoffman

Cody Klingler

Advisors: Maddumage Karunaratne, PhD

Christopher L. Gabany

Table of Contents:

Problem Identification:

Currently, there are several wearable devices available for cyclists that offer general data collection about a cycling session; however, these do not offer specific statistic tracking for the user, only general data about the location, duration, and exercise of the cycling trip. Specialized meters designed for cyclists do not offer a wide range of data, but only specific measurements for the activity. Cyclists require a better, more encompassing solution for setting goals and tracking improvement. The Cycle Statistic Tool is an opportunity to provide an all-in-one solution for cyclists through the use of advancements in available technology, such as low cost microcontrollers and electrical components that can be attached to a bicycle.

Specified Engineering Solution:

The Cycle Statistic Tool is a data recording device which will be attached to the frame of a bicycle. In this position, several electronic components may be used to record statistics about a user's cycling trip. The Cycle Statistic Tool will transmit data to a companion application that is installed on a user's smartphone. This application will utilize the collected information to provide records of a user's cycling sessions in the form of graphs and charts that enable the improvement of their cycling skills. *Figure 1* depicts a general layout of the Cycle Statistic Tool's specified final design. An appropriate Arduino microcontroller will be used as the primary board to connect all sensors and for operation of the device. A secondary Arduino microcontroller will be attached to the pedal shaft to wirelessly relay data from this position to the primary board. The block diagram in *Figure 2* represents the system of hardware components used in the Cycle Statistic Tool to collect live metrics. Data will be displayed to the user by a display panel connected to the primary board. The entire device will require a weatherproof casement, which will be modeled and 3D-printed. This casement should be capable of universally mounting to the frame or handlebar of a bicycle.

Project Changes:

Hardware:

- The secondary microcontroller did not use the original 3.7V lithium ion battery specified, but instead three 1.5V AA alkaline batteries in series. This was not a final design specification, but a temporary solution to get the Cycle Statistic Tool prototype operating. The 3.7V lithium ion battery originally ordered by the project team did not supplement the 5V required to operate the load cell amplifier. The project team attempted installing a spare 5V USB portable battery, but it would not work due to its proprietary circuit design, which would automatically turn off when connected to a device with low power draw. Ideally, this would be rectified using the original 3.7V lithium ion battery and a voltage regulator, but the project team did not have adequate time to order this additional component.
- The anemometer and wind angle fin were not installed before the final presentation and subsequent report. Although the project team still planned on adding these components and had begun development, they were excluded due to time constraints.
- The photoresistor and white LED were not installed. These components were not considered high priority, and to further ensure the stable operation of the prototype hardware, the project team avoided installing these simple components to cut down on material cost, power consumption, and used space. A revised version of the Cycle Statistic Tool may include a display light and a headlight installed into the main device casement.
- The universal, weatherproof casement was never designed or installed. The project team decided to save on time by using a pre-fabricated solution for the primary hardware device's casement. In hindsight, this went against not only the specified engineering solution, but also disabled the project team's ability to test the weatherproof engineering specification. Additionally, the project team could have put more thought into incorporating the design and construction of the casement into the project's task division to better reflect and enforce the specified design.
- The specified design included a calibration button, but the project team decided to install a power switch for the device instead, which ended up providing a recalibration for the

primary microcontroller when rebooted. The secondary microcontroller had no persistent reset or calibration installed, but the project team was able to unplug the power source from the board to reboot and subsequently recalibrate the components.

- A load cell amplifier had to be included into the final design because the strain gauges' fluctuations in resistance were too small to be measured by an arduino.
- The final strain gauge circuit design was changed to incorporate a full bridge—implementing four strain gauges instead of two—so that if a strain gauge failed, the design could be converted back to a half bridge, thus eliminating the need to remount strain gauges. This would have cost the project team a large amount of time due to the complexity of mounting the sensors on the crank arm.
- The specified design originally included an infrared (IR) sensor mounted on the crank arm or the frame of a bicycle to measure the angular velocity of the pedal. This was replaced by the secondary microcontroller's integrated inertial measurement unit (IMU) due to more accurate instantaneous values. This was considered important for the accuracy of the power measurement, which depends on both the instantaneous angular velocity and the torque on the crank arm.

Software:

- The mobile application did not originally specify several NuGet packages that were used in the final application. These packages include .NET assemblies and their required files in the target project. The project team could have potentially avoided the use of these community packages by selecting a different development platform for the mobile application. Using the NuGet Package Manager UI in Visual Studio, the following packages were installed into the Xamarin project:
    - Plugin.BLE: The Xamarin mobile application platform does not have a native Bluetooth Low Energy implementation; however, the developer community has created Plugin.BLE, an open-source plugin that allows access to Bluetooth Low Energy functionality on Android and iOS.
    - Microcharts.Forms: This package is a simple, cross-platform chart library. The Cycle Statistic Tool's Companion mobile application utilizes the Microcharts package to easily draw tables of the user's performance results after recording a

trip. Although the package is installed, this feature was not fully implemented in the mobile application.

- SQLite: To enable the mobile application to store local data in a table format, the project team used the SQLite NuGet package. Unlike a traditional database management system, SQLite is not a client-server database engine, and instead it embeds an SQL database engine into the local storage of the end device. Due to the nature of the data being recorded for a trip, the project team believed a database would futureproof the design with respect to additional features, such as plotting and exporting data.

Final Results:

The development of the Cycle Statistic Tool remained as close to specification as possible, apart from the lack of wind angle fin apparatus, anemometer, display LED, and weatherproof casement. The project team successfully developed an easy to mount data recording device for a bicycle. The primary microcontroller can be seen mounted on the handlebar of a bicycle in *Figure 3.* Presented in *Figure 4* and *Figure 5* are the dimensions and internals of the primary microcontroller. The primary microcontroller is powered by a 3.7V lithium ion battery. The secondary microcontroller installed on the crank arm of a bicycle is shown in *Figure 6*. The secondary microcontroller is currently powered by three 1.5V AA alkaline batteries in series. From these positions on a bicycle, the Cycle Statistic Tool can measure a myriad of live metrics pertaining to the vehicle. Using Bluetooth Low Energy (BLE), the collected live data can be transmitted to a paired Android or iOS smartphone through the Cycle Statistic Tool Companion mobile application. The mobile application will dynamically display the data anytime the hardware notifies it's observer of an updated value. A generic attribute (GATT) profile table representing the services and corresponding characteristics transmitted across BLE is shown in *Table 1*. Each characteristic represents an output data value measured by the Cycle Statistic Tool hardware.

The Arduino 33 IoT ABX00027 device used for the primary and secondary microcontrollers had been selected due to the built-in IMU, which can measure 3-axis acceleration and 3-axis angular velocity. The IMU associated with the primary microcontroller is mounted on the frame of a bicycle and measures a ratio of accelerations to calculate the angle of incline. The IMU associated with the secondary microcontroller is mounted on the crank arm of a bicycle and measures the angular velocity of the pedal. This measurement is subsequently used to calculate cadence (pedals per minute), power, and calories burned. Additionally, the selected device includes a BLE module that allows for bluetooth low energy data transmission, which is utilized by the primary and secondary microcontrollers. The Cycle Statistic Tool's hardware had been programmed using C++ and the Arduino IDE. The collection of metrics acquired by the Cycle Statistic Tool hardware become presented to the user on the display, which is shown in *Figure 7.* The display and display breakout are connected to the primary microcontroller. The display breakout is attached directly to the display and acts as a data buffer and controller. The IR sensor mounted on the bicycle frame is connected to the primary microcontroller. In *Figure 8,* the IR sensor is shown facing the path of an aluminum disc that has been mounted on the axle of a bicycle. As the axle spins, so does the aluminum disc, and the IR sensor is able to determine a full rotation of the tire based on detecting the aluminum disc. The IR sensor is able to accurately quantify the rotation of the tire, which is then used to calculate speed, distance, and acceleration. Embedded inside of the primary microcontroller's casement, there is a GPS module that measures latitude, longitude, and altitude. A full bridge strain gauge circuit mounted on the crank arm of a bicycle is connected to the secondary microcontroller. The full bridge strain gauge circuit is shown in *Figure 6*, and an enlarged image of a single strain gauge is provided in *Figure 9*. The strain gauges are variable resistors that change value based on the deflection of the crank arm. This deflection is caused by the torque applied to the pedal by the cyclist. The strain gauges are assembled in a wheatstone bridge. Both the input and output voltages are measured by a load cell amplifier that outputs a digital signal corresponding to the strain on the crank arm. The strain gauges are used to calculate power, calories burned, and torque. Finally, the finished main pages of the Cycle Statistic Tool's Companion mobile application are presented in *Figure 10*. The mobile application was based on the software requirements specification document, which outlines the design of the Cycle Statistic Tool Companion mobile application, and remained as close to specification as possible. The mobile application was implemented using the Model-View-ViewModel (MVVM) architecture pattern. The Cycle Statistic Tool Companion mobile application was programmed in C# using the .NET platform, Xamarin framework, and Visual Studio for Mac.

The project team's final engineering targets results can be seen in *Table 2*. The *Size* target refers to the total volume of the Cycle Statistic Tool's hardware. The actual result was verified by measuring the dimensions of the assembled components and calculating the total volume. Due to the exclusion of the wind angle fin apparatus and anemometer, the actual size was dramatically under the target. The *Overall Accuracy of Sensors* target refers to the overall accuracy of all measurements not directly specified in the engineering targets. Unfortunately, the project team did not determine the actual result for this target due to time constraints and missing components. The *Accuracy of Power Metric* target tests how close the calculated power value is to the actual power output on a measured bicycle. The accuracy of power metric was determined by calculating the accuracy of angular velocity metric multiplied by the accuracy of torque metric. The torque metric's accuracy is equivalent to the accuracy of the load in kilograms that is measured by the strain gauges. By placing known mass values on the crank arm and comparing the measured value to the actual value, the torque metric's worst-case accuracy was determined to be roughly 80%. The *Accuracy of Speed Metric* target measures how close the calculated speed value is to the actual speed of a measured bicycle. By measuring the accuracy of distance, the project team tested the Cycle Statistic Tool's accuracy of speed. The speed calculation is determined using change in distance and elapsed time, which is based on the ~99.99% accurate arduino clock. The project team rode a bicycle being measured by the Cycle Statistic Tool down a road several times and then compared the measured distance to the actual distance calculated using Google Maps. By using this testing process, the project team's actual result was 90.8%. The *Battery Life* target was evaluated based on the amount of time that the Cycle Statistic Tool was used. The longest duration of time where the Cycle Statistic Tool hardware remained powered on was four hours, which was above the project team's threshold of three hours. The *Lifespan* target refers to the expected duration of time where the Cycle Statistic Tool hardware maintains operability. Unfortunately, the project team could not determine the actual result of this target due to variances in usage conditions from independent users and lack of time. Based on the fragility of the strain gauges in their current, unprotected state, the lifespan of the Cycle Statistic Tool was estimated to be around two weeks. The *Time to View Metrics in App* target was tested by recording the elapsed time while launching the mobile application, connecting to the device, and displaying live metrics. The actual result was five seconds, which is below the project team's threshold of six seconds. The *Setup Steps Before Use* target's actual result was four total steps, which is less than the threshold of six steps:

1. Power on the primary microcontroller using the power switch on the left side.
2. Turn the left crank arm until it is parallel with the ground.
3. Power on the secondary microcontroller by plugging in the power source.
4. If desired, the user may now connect the Cycle Statistic Tool Companion mobile application.

The *Steps to Attach to Bike* target refers to the amount of steps required to set up the Cycle Statistic Tool hardware. Assuming that the user is provided a prefabricated crank arm that includes the secondary microcontroller and corresponding components, and the user has already removed the original crank arm, the actual result was six steps, which is less than the target of ten steps:

1. Set the Cycle Statistic Tool crank arm (secondary microcontroller) on the bicycle.
2. Fasten the crank arm bolt.
3. Attach the aluminum disc to the rear tire axle with adhesive.
4. Secure the Cycle Statistic Tool user display (primary microcontroller) to the bicycle's handlebars with zip-ties.
5. Secure the IR sensor to the bicycle frame with solid core wire.
6. Secure the loose wires between device components to the frame of the bicycle using zip-ties.

Finally, the *Cost of Materials* target's actual result was $460.79, which was below the target of $500. The bill of materials determined by the project team on 04/18/2021 can be viewed in *Table 3.* Even though the project team incurred unnecessary additional costs from ordering replacement parts, the target goal was still achieved.

Over the course of the Spring semester, the project team completed about 80% of the Cycle Statistic Tool's required tasks for completion. In *Table 4*, the current state of the Task Division Table displays an overlook of the projected time required to complete each project task. The Cycle Statistic Tool was estimated to require 440 hours to complete. Documentation of the project team's labor hours can be viewed in the project timesheet under *Table 5.* The final time documented by the project team was 583.7 hours, as shown in *Table 6,* which displays the cost analysis of the Cycle Statistic Tool determined by the project team on 04/18/2021. The Cycle Statistic Tool will be completed after debugging, testing overall accuracy of hardware components, and implementing the anemometer and wind angle fin apparatus have been finished. Additionally, some incomplete mobile application features allow plotting of data that could benefit the project team during testing. To complete the Cycle Statistic Tool, the project team has estimated an additional 100 hours of labor will be required.

Conclusions and Recommendations:

The Cycle Statistic Tool reinforced principles of the engineering design process by requiring the proposal, design, and development of an engineering solution for a selected problem. The project team believed teamwork and organization to be the main components for success for the Cycle Statistic Tool. To accommodate the remote work environment that the project team had to collaborate within, the majority of the required tasks for completion were divided between an emphasis on hardware and software. The project team believed this would allow for asynchronous development of the hardware and mobile application; However, during the last two weeks of the allotted time for implementing the Cycle Statistic Tool, the project team collaborated on the project in person. Testing the mobile application—specifically BLE connectivity between the smartphone and the Cycle Statistic Tool hardware—became tremendously easier whenever the hardware was physically there for all of the employees to access. The project team believes troubleshooting the Cycle Statistic Tool would have been more efficient without operating remotely. Overall, the project team found the lessons and experience gained from proposing, designing, and developing the Cycle Statistic Tool to be invaluable and fundamentally necessary for becoming a professional engineer.

Appendix:

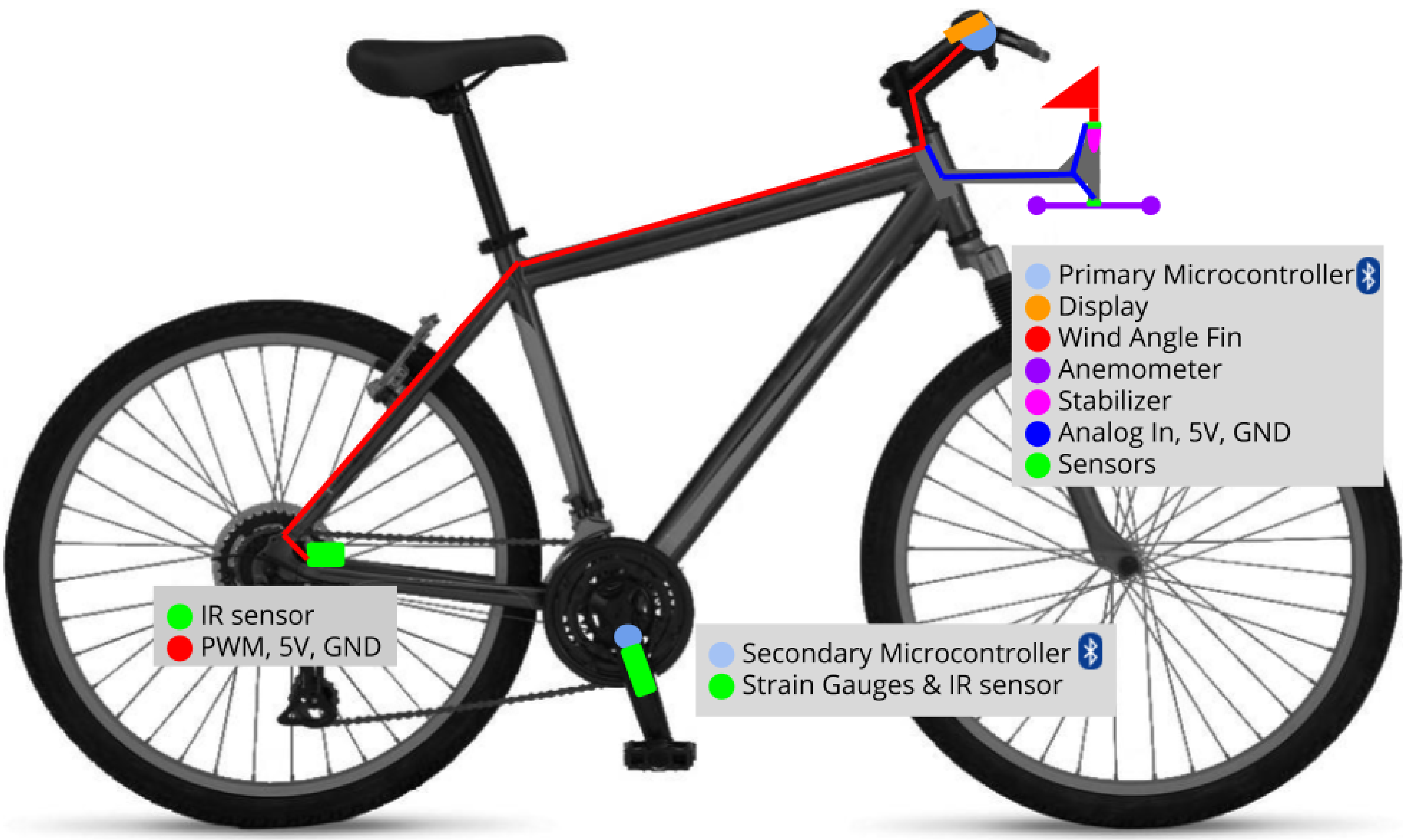

Figure 1: A general layout of the Cycle Statistic Tool's specified final design.

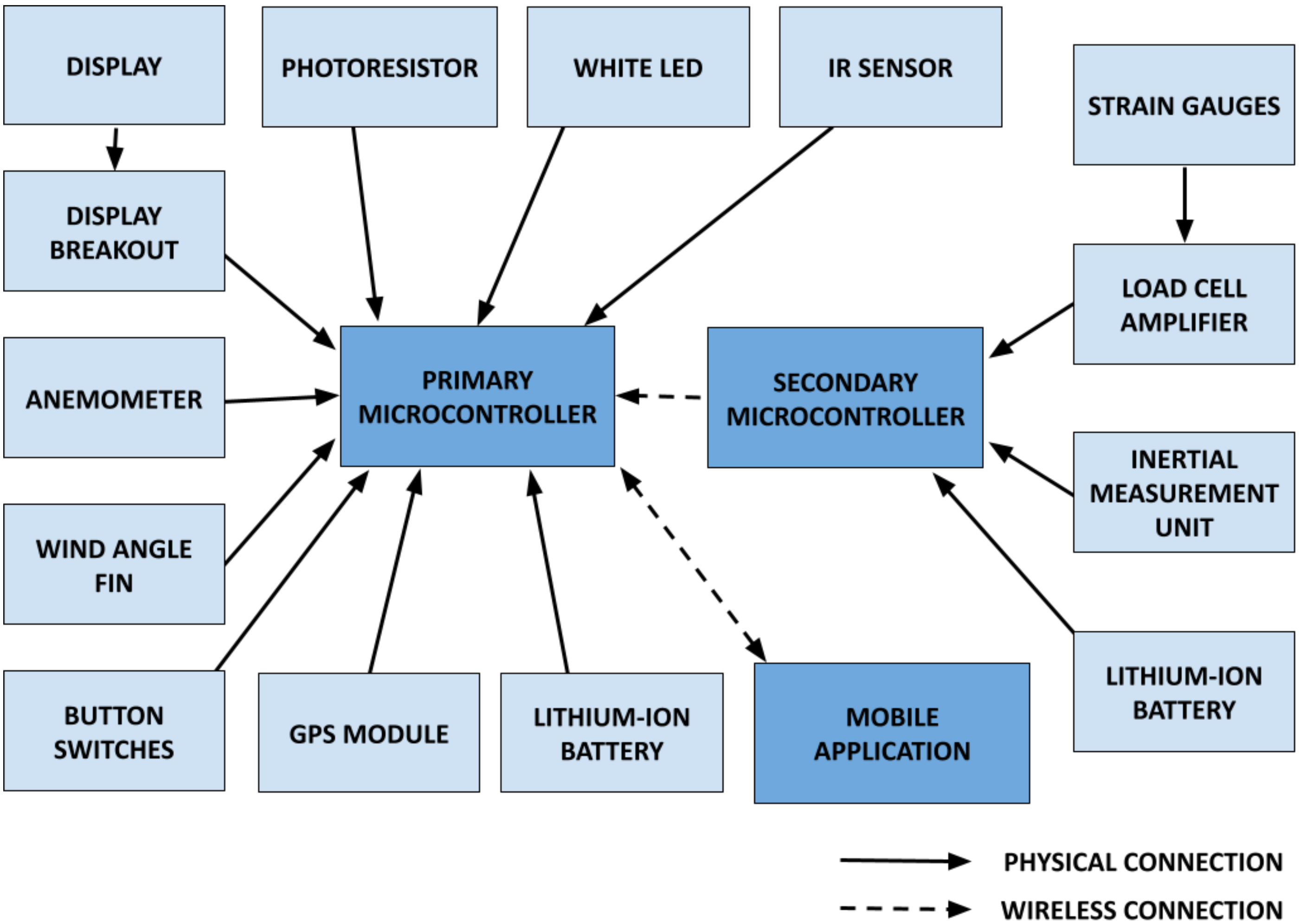

Figure 2: The Block Diagram represents the system of hardware components used in the Cycle Statistic Tool to collect live metrics.

Figure 3: The primary microcontroller mounted on the handlebar of a bicycle.

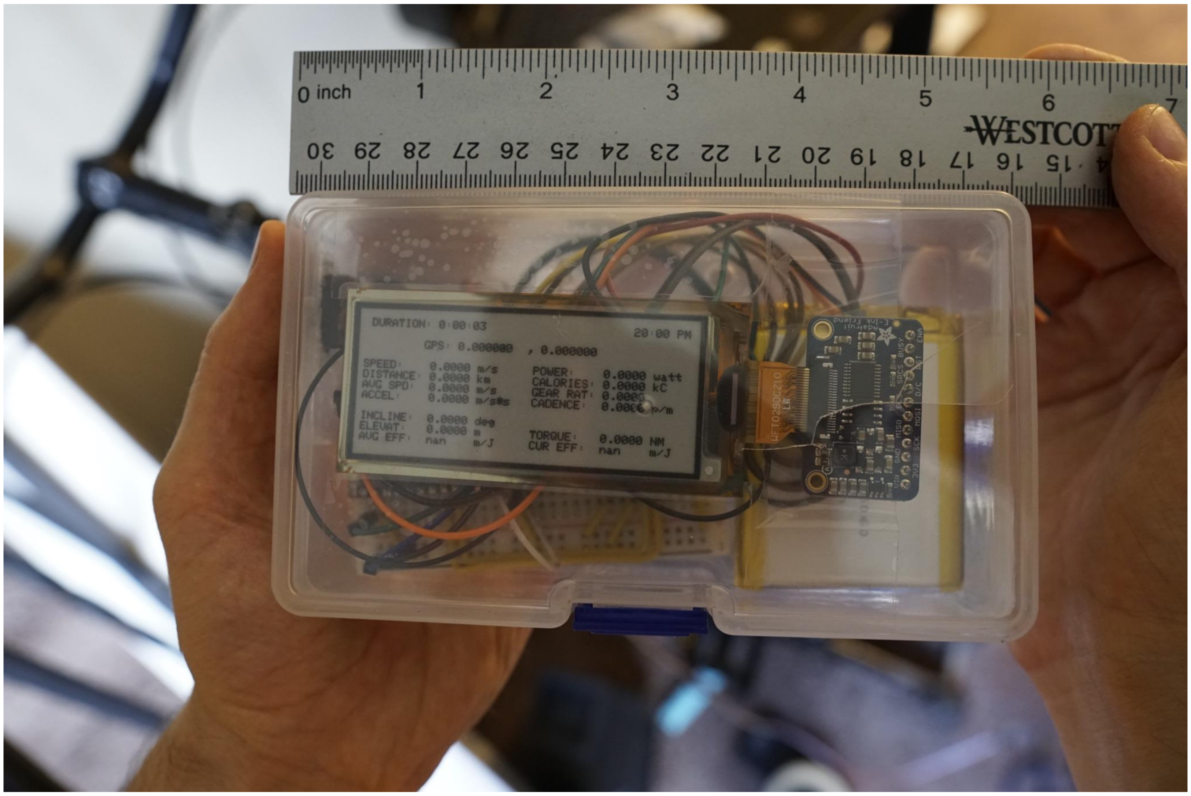

Figure 4: The general dimensions of the enclosed primary microcontroller.

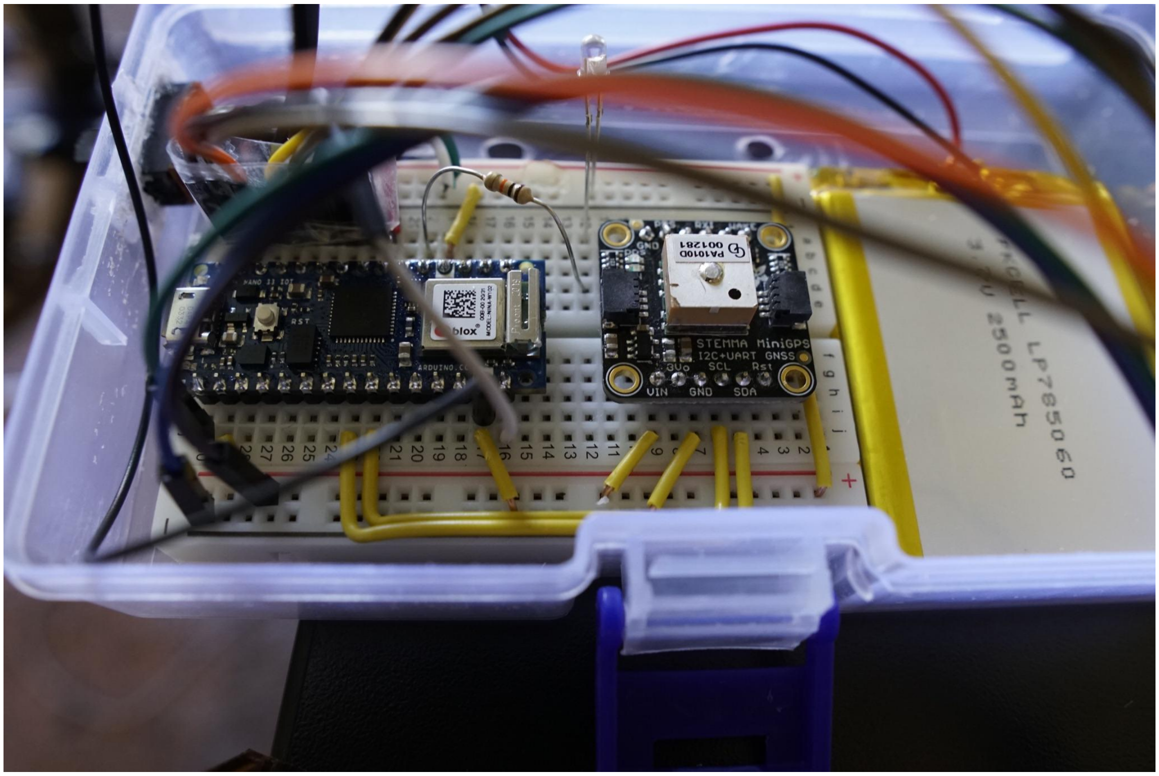

Figure 5: The internal circuitry of the primary microcontroller. The primary microcontroller (left), the GPS module (center), and the 3.7V lithium ion battery (right) are visible.

Figure 6: The secondary microcontroller mounted on the crank arm of a bicycle. The secondary microcontroller (back of crank arm), the full bridge strain gauge circuit (top of crank arm), and the three 1.5V AA alkaline batteries in series (front of crank arm) are visible.

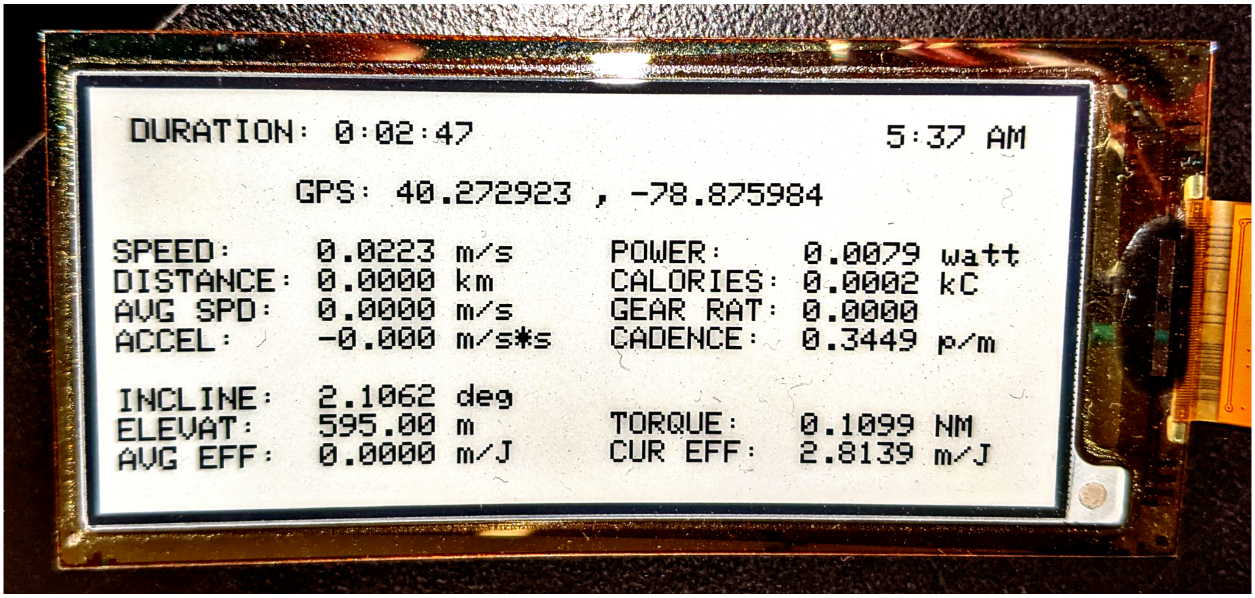

Figure 7: The display of the Cycle Statistic Tool presenting the collection of metrics acquired by the hardware.

Figure 8: The IR sensor mounted on the frame and aluminum disc mounted on the rear axle of a bicycle.

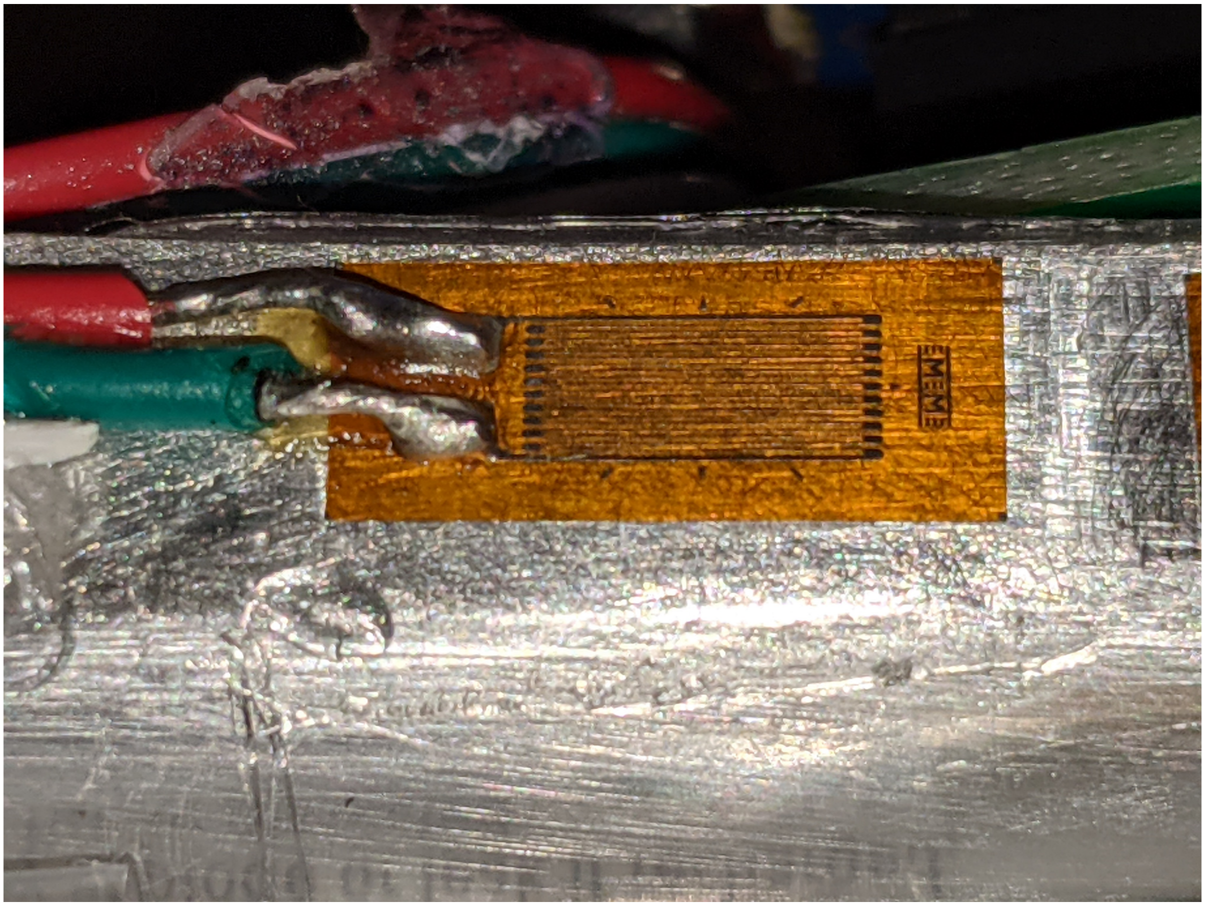

Figure 9: An enlarged photo of a strain gauge mounted on the crank arm of a bicycle.

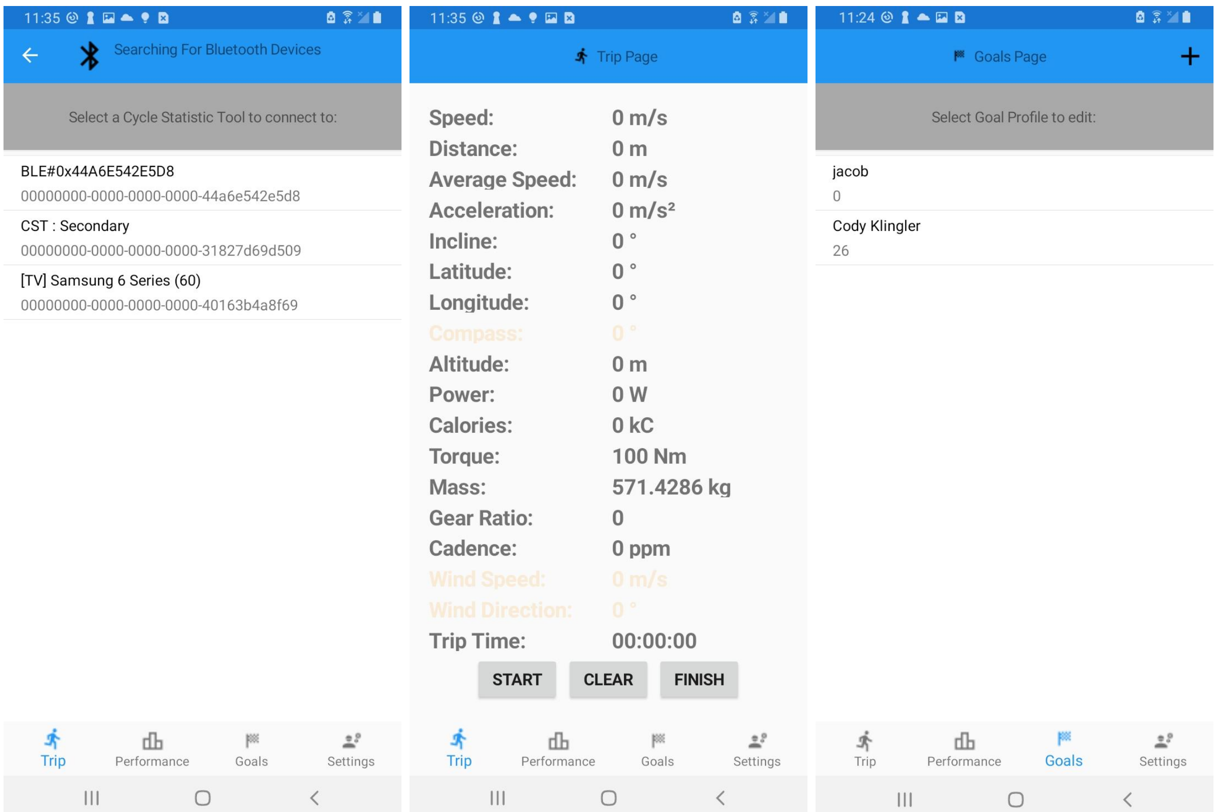

Figure 10: The pages of the Cycle Statistic Tool Companion mobile application. The device list page (left) allows the user to select a locally discovered device. Once the Cycle Statistic Tool hardware has paired, the trip page (center) automatically displays dynamic metrics from the hardware. The goals page (right) allows the user to create/edit custom goal profiles, which locally saves persistent information about a user, such as height, weight, and tire diameter.

Table 1: The GATT Table represents the services and corresponding characteristics that are transmitted across Bluetooth Low Energy. Live data from the pedal shaft is transmitted from the secondary microcontroller to the primary microcontroller, and all of the live metrics are transmitted from the primary microcontroller to the connected smartphone through the Cycle Statistic Tool Companion mobile application.

Name	GUID	Units	Data Type	Permissions
CST Service	f5806ca8-6c2f-4f9d-a232-b43b200c9479			
Speed	08cdf8af-d030-43ed-85ea-24d020556e21	m/s	Float	Read | Notify
Distance	e4e8f3d1-38f6-48ac-b3a2-a9b8987d6947	km	Float	Read | Notify
AverageSpeed	e76d7d76-5a7a-408f-9844-2dd6eabbc8e6	m/s	Float	Read | Notify
Acceleration	3e29dd5a-c12a-4ccb-b34c-f9d7d6bfbc47	m/s*s	Float	Read | Notify
Incline	fe3a2d74-a83b-40d9-81da-49986812c7d2	degrees	Float	Read | Notify
GPS Service	b6eb07bd-1079-4a61-ab37-f18b73af22d2			
Latitude	5af0c5fa-43f6-42f1-a749-a4a497dde66e	degrees	Float	Read | Notify
Longitude	81d6828c-ddf2-48bc-a3c8-7382336926ef	degrees	Float	Read | Notify
Altitude	100e4627-3e40-437d-b9ca-c63d89b6348f	m	Float	Read | Notify
Pedal Service	732ca8ca-b303-4e56-974a-bb642d11bb3b			
Power	a1a882a8-dd89-4c86-99d3-b36045fa280d	watt	Float	Read | Notify
Calories	9c90a5cb-02d1-4f56-bf86-7a65f197074a	kC	Float	Read | Notify
GearRatio	de75328d-63fb-4307-b993-c7e240a01b5a		Float	Read | Notify
Cadence	6e072e2b-87d7-4df9-b3e0-02c5f9b4d8a9	p/m	Float	Read | Notify
Torque	e8c8136c-80d3-4450-9b4d-619647eb6314	n*m	Float	Read | Notify
Currently Not Broadcasting				
WindSpeed	c3092ab4-3b63-4090-a6c7-d6ddc4afd8f0	m/s	Float	Read | Notify
WindDirection	3d55cdfc-493f-484a-aa64-0ad499b99c0a	degrees	Float	Read | Notify
Compass	199d3869-f865-48b8-8368-32057568e4cb		Float	Read | Notify

Table 2: The Engineering Specifications Table outlines the project team's target goals.

Engineering Specification	Unit	Direction	Target	Threshold	Actual
Size *Ease of use*	m^3	▼	.2	.5	0.00023
Overall Accuracy of Sensors *Accuracy of measurements* *Price* *Battery life*	%	▲	90	80	-
Accuracy of Power Metric *Accuracy of measurements* *Price* *Battery life*	%	▲	90	75	~ 76
Accuracy of Speed Metric *Accuracy of measurements* *Price* *Battery life*	%	▲	95	85	90.8
Cost of Materials *Accuracy of Measurements* *Price* *Battery life* *Durability*	$	▼	500	700	460.79
Weatherproof *Price* *Weatherproof* *Durability*	IP	▲	23	54	-
Battery Life *Price* *Battery life*	hr	▲	12	3	> 4
Lifespan *Weatherproof* *Durability*	mo	▲	24	12	-
Time to View Metrics in App *Statistic utilities in app*	s	▼	2	6	5
Setup Steps Before Use *Attaches universally* *Ease of use* *Ease of setup*		▼	3	6	4
Steps to Attach to Bike *Attaches universally* *Ease of use* *Ease of setup*		▼	10	20	10

Table 3: The Bill of Materials determined by the project team on 04/18/2021.

Part Number	Product Name	Quantity	Actual Cost	Description	Standards
ABX00027	Arduino Nano 33 IOT	2	$18.40	Microcontrollers with Bluetooth	IEEE 802.11b/g/n
Adafruit PID: 4224	Adafruit eInk Breakout Friend with 32KB SRAM	1	$8.50	Memory buffer for eInk display	
Adafruit PID: 4262	2.9" Flexible Monochrome eInk	1	$26.95	Low power eInk display	
C000160	Grove - Infrared Reflective Sensor V1.2	2	$5.90	Measures angular frequency	
CEA-06-125UT-350	Micro Measurements Strain Gages	1	$29.99	Measures sheer force	
HX711	Load Cell Amplifier / ADC	1	$9.95	Reads strain from gauges	
176SF1D-L394	Honeywell Precision Continuous 10K Potentiometer	1	$22.95	Measures wind angle	
PA1010D	Adafruit Mini GPS	1	$29.95	10Hz GPS module	GPS SPS PS
LP785060	Lithium Ion Polymer Battery - 3.7v 2500mAh	1	$14.95	Power for Microcontrollers	
	Metal Crank Arm	1	$13.63		
	Misc		$225.87	Jumpers, Headers, Solder, additional strain gages, additional microcontrollers, glue, catalyst, etc.	
	Shipping		$53.75		
		TOTAL	**$460.79**		

Table 4: The state of the Task Division Table as documented on 04/18/2021 illustrates the progress made by the project team. Additionally, the Task Division Table provided appropriate start and end dates for each task to complete the Cycle Statistic Tool within the appropriate time frame.

Critical	Task Description	Duration	Start Date	End Date
	Research Components	6 hrs	Fri 1/15/21	Sat 1/16/21
•	Design System	6 hrs	Mon 1/18/21	Wed 1/20/21
	Design Display UI	4 hrs	Fri 4/2/21	Sat 4/3/21
	Design Wind Sensing Unit	9 hrs	Sun 3/7/21	Fri 3/12/21
	Assemble Wind Sensing Unit	13 hrs	Mon 4/12/21	Mon 4/19/21
	Design Mobile Application	12 hrs	Wed 2/10/21	Wed 2/17/21
	Test Bluetooth Communication	12 hrs	Wed 4/7/21	Mon 4/12/21
	Research Bluetooth	6 hrs	Fri 2/19/21	Sat 2/20/21
	Code Microcontrollers	76.5 hrs	Fri 1/29/21	Sat 3/13/21
	Test Components	12 hrs	Mon 1/25/21	Sat 1/30/21
	Mount Hardware	12 hrs	Sun 3/14/21	Sat 3/20/21
	Test Accuracy of Calculations	14 hrs	Wed 4/7/21	Wed 4/14/21
	Weatherproof	3 hrs	Wed 4/14/21	Wed 4/14/21
	Test Durability	6 hrs	Fri 4/16/21	Sat 4/17/21
	Research Data Collection	8 hrs	Wed 3/17/21	Sat 3/20/21
	Research Data Analysis	6 hrs	Wed 3/17/21	Fri 3/19/21
•	Research Mobile Development	10 hrs	Wed 1/13/21	Wed 1/20/21
•	Code Mobile Application	78 hrs	Fri 2/19/21	Sat 4/3/21
•	Debugging	27 hrs	Mon 4/5/21	Mon 4/19/21

Task Status Key
Completed
In Progress
Not Started

Table 5: The Project Timesheet provides the project teams' individual labor entries accompanied by a description of the task involved.

Name	Component	Time (min)	Date	Description
Cody	Research	180	9/1/2020	Researched sensors for proposal
Jake	Research	120	9/1/2020	Researched Microcontrollers, sensors for proposal
Cody	Research	240	9/2/2020	Researched sensors, IO, logistics of project
Cody	Documentation	180	9/2/2020	Proposal
Jake	Documentation	180	9/2/2020	Proposal
Cody	Documentation	170	9/12/2020	Proposal
Jake	Documentation	180	9/12/2020	Proposal
Cody	Documentation	90	9/19/2020	Oral Presentation
Jake	Documentation	120	09/20/2020	Oral presentation
Cody	Documentation	50	9/24/2020	Final Proposal
Jake	Documentation	120	9/24/2020	Final Proposal
Cody	Documentation	120	9/30/2020	House of Quality
Jake	Documentation	120	09/30/2020	House of Quality
Cody	Documentation	210	10/1/2020	House of Quality
Cody	Documentation	390	10/6/2020	HoQ, Func Req.
Jake	Documentation	60	10/04/2020	Assembled digital binder

Jake	Documentation	60	10/04/2020	Assembled physical binder
Jake	Documentation	210	10/01/2020	House of Quality
Jake	Documentation	390	10/06/2020	Func Req.
Cody	Documentation	15	10/07/2020	HOQ
Cody	Documentation	300	10/09/2020	Func Req. diagram, table, hoq, flowchart
Jake	Documentation	60	10/28/2020	Func Req.
Jake	Documentation	180	11/03/2020	Block Diagram
Jake	Documentation	30	11/03/2020	Func Req.
Cody	Documentation	60	11/10/2020	Recreated Schematic
Cody	Documentation	120	11/31/2020	Time Schedule Task Division
Cody	Documentation	120	12/4/2020	Pin Out, BoM
Jake	Documentation	120	11/31/2020	Time Schedule Task Division
Jake	Documentation	210	12/04/2020	Final Design Presentation,Assembled digital binder
Jake	Research	480	01/12/2021	Researched Xamarin platform by Microsoft
Jake	Research	480	01/13/2021	Read book on Xamarin and app development
Jake	Application	240	01/14/2021	Made general application for testing Xamarin
Jake	Documentation	240	01/22/2021	Worked on Software Requirement Specifications document
Jake	Documentation	240	01/24/2021	Worked on Software Requirement Specifications document
Jake	Documentation	240	01/29/2021	Worked on Software Requirement Specifications document

Jake	Documentation	240	01/31/2021	Worked on Software Requirement Specifications document
Jake	Application	120	02/06/2021	Worked on tabbed page design for app
Jake	Application	120	02/07/2024	Worked on tabbed page design for app
Jake	Application	120	02/12/2021	Implemented goal/trip databases
Jake	Application	120	02/13/2021	Implemented goal/trip databases
Jake	Application	240	02/20/2021	Implemented goal/trip databases
Jake	Application	120	02/21/2021	Goals page implementation
Jake	Application	120	02/22/2021	Added general settings
Jake	Application	120	02/25/2021	Worked on Trip View
Jake	Application	240	02/26/2021	Worked on databases
Jake	Documentation	240	03/01/2021	Monthly presentation
Jake	Application	240	03/06/2021	Developing View Model
Jake	Application	180	03/07/2021	Developing View Model
Jake	Research	120	03/10/2021	Researching Xamarin BLE
Jake	Application	120	03/12/2021	Setting up Plugin.BLE
Jake	Application	180	03/13/2021	Setting up Plugin.BLE
Jake	Application	300	03/14/2021	Bug Fixing, set up Github repo
Jake	Application	240	03/15/2021	Setting up Plugin.BLE
Jake	Application	300	03/19/2021	Setting up a new Nuget Packet, Reactive Bluetooth
Jake	Application	360	03/20/2021	tried to set up a new bluetooth module, failed, reverted back to Plugin.BLE

Jake	Application	180	03/21/2021	trying to set up some BLE stuff in Android native
Jake	Research	300	03/25/2021	spent time researching native Android BLE development and Android Visual Studio
Jake	Research	240	03/26/2021	researching setting up Plugin BLE in Xamarin
Jake	Application	240	03/27/2021	setting up Plugin BLE sample cope inside of Companion application (weird)
Jake	Application	360	03/28/2021	setting up Plugin BLE sample cope inside of Companion application (weird)
Jake	Application	360	03/29/2021	got BLE View Model working (old code)
Jake	Application	480	04/02/2021	setting up Trip View
Jake	Application	360	04/04/2021	setting up BLE View Model, made GATT spreadsheet
Jake	Application	360	04/05/2021	setting up BLE View Model
Jake	Application	480	04/06/2021	Getting project ready for SPACE demonstration
Jake	Application	360	04/11/2021	Data binding
Jake	Application	240	04/12/2021	Data binding
Jake	Application	720	04/13/2021	Setting up new SQLite database, tested bluetooth
Jake	Application	540	04/14/2021	Set up new bluetooth View Model client
Jake	Application	390	04/15/2021	Set up service/characteristics connectivity, dynamic data
Jake	Application	780	04/16/2021	working on saving trips
Cody	Microcontroller	540	02/01/2021	Writing test code for gps, cpu fan, ir sensor, potentiometer, display

Cody	Assembly	30	02/05/2021	Set up bicycle on jack stand to allow for indoor testing
Cody	Microcontroller	420	02/07/2021	Aimed IR sensor on bicycle tire. attached aluminum strips to the tire.
Cody	Microcontroller	360	02/09/2021	Soldered antenna, soldered cables for LION batteries, debounced IR Sensor
Cody	Microcontroller	120	02/12/2021	Fixed BLE issues, BLE coding
Cody	Microcontroller	300	02/15/2021	Sent tire RPM over BLE
Cody	Assembly	240	02/18/2021	Soldered two strain gauges.
Cody	Assembly	240	02/19/2021	Glued strain gauge to bicycle crank-arm.
Cody	Assembly	720	02/20/2021	More strain gauge work. Broke the last strain gauge.
Cody	Assembly	300	02/25/2021	Constructed two anemometers, tested them.
Cody	Microcontroller	480	02/26/2021	Started putting the main code together. rewrote code. tested BLE buffers
Cody	Assembly	270	03/03/2021	Mounted 3 strain gauges. One had a wrinkle from excess catalyzer
Cody	Misc	60	03/05/2021	Ordered additional parts
Cody	Assembly	300	03/07/2021	Soldered two more strain gauges. signal was extremely noisy
Cody	Assembly	315	03/11/2021	Modified HX711 to make it operate at 3.3V (failed)
Cody	Assembly	120	03/12/2021	Tested HX711 at 5V. Signal is still noisy.
Cody	Assembly	750	03/14/2021	Flattened crank arm for gauges. Mounted a new secondary gauge.

Cody	Assembly	390	03/15/2021	Replaced primary strain gauge. flattened crank arm again. tested measurements.
Cody	Assembly	180	03/16/2021	Primary gauge broke. Ordered additional gauges.
Cody	Assembly	150	03/20/2021	Drilled holes in the hobby box and ripped a switch from a toy car.
Cody	Assembly	315	03/21/2021	Created a permanent solution for mounting an IR sensor on the tire.
Cody	Assembly	600	03/22/2021	Applied a new strain gauge to the crank arm. Created a permanent circuit .
Cody	Assembly	240	03/23/2021	Converted strain gauge design to full bridge.
Cody	Assembly	495	03/24/2021	Added new gauges for full bridge and modified permanent circuit.
Cody	Assembly	585	03/27/2021	Fixed gauge issue by manually setting gain. Made variations of circuits.
Cody	Microcontroller	315	03/28/2021	Finalizing microcontroller code.
Cody	Documentation	270	04/02/2021	Worked on SPACE deliverables.
Cody	Microcontroller	360	04/03/2021	Continued finalizing microcontroller code. Implemented new classes
Cody	Microcontroller	465	04/05/2021	Reconfigured microcontroller code for the third monthly demonstration.
Cody	Microcontroller	300	04/07/2021	Fixed compiler errors and wrote main for the primary microcontroller.
Cody	Microcontroller	375	04/09/2021	Tested primary microcontroller code. Was bug free and needed minor adjustments.
Cody	Microcontroller	300	04/10/2021	Troubleshooting the GPS code. Pointers wouldn't work

Cody	Misc	330	04/11/2021	packed up project, drove to johnstown, set up work shop at Jake's apartment
Cody	Microcontroller	390	04/12/2021	Fixed GPS issue caused by poorly documented code.
Cody	Microcontroller	600	04/13/2021	Fixed GPS BLE crash. Array was not properly sized.
Cody	Assembly	735	04/14/2021	Broke the secondary controller and had to order two new ones.
Cody	Microcontroller	645	04/15/2021	Finalized display code, fixed another GPS issue
Cody	Microcontroller	810	04/16/2021	Fixed BLE issues with app and secondary microcontroller connected at once
Cody	Microcontroller	600	04/17/2021	Set up rest of BLE characteristics and passed measurements from secondary
Jake	Documentation	300	04/18/2021	Final presentation
Cody	Microcontroller	840	04/18/2021	Soldering final circuits/protoboards. Nothing made this day worked
Cody	Microcontroller	840	04/19/2021	Mounted hobby box with the breadboard. Tested batteries for secondary controller

Table 6: The Cost Analysis of the Cycle Statistic Tool determined by the project team on 04/18/2021.

Employe e	Total Hours	Estimated Hours to Complete	Predicted Total Hours
Cody	295.2 hrs	50.0 hrs	345.2 hrs
Jake	243.5 hrs	50.0 hrs	293.5 hrs
Total:	**538.7 hrs**	**100.0 hrs**	**638.7 hrs**

Labor Hourly Cost (based on $60,000/year salary):	$30.00 /hr
Overhead Cost at 150% of Direct Labor Costs:	$45.00 /hr
Total of Salary and Overhead Costs:	$75.00 /hr
Add 20% Profit Margin:	$93.75 /hr
Total Cost for Labor, Overhead and Anticipated Profit:	$59,875.00
Cost of Materials (from Bill of Materials):	$460.79
Total Project Cost for Amortized Engineering and Business Costs:	**$60,335.79**