CE 186 Final Report

aSYST-ME eB: Automated System for Managing Energy in eBikes Battery Energy Coach

Team Members:
Preet Gill, Max Morrison, Grace Vasiknanonte, and Katie Zheng

I. ABSTRACT

Despite the benefits of using an electrically-powered bike, there is a huge concern with this type of technology: an eBike with dead batteries leaves the user worse off than a regular bike. To avoid this issue, we have created a battery energy management coach. This is a prime example of a cyber-physical system, encompassing a physical hardware layer, which communicates with a software/cyber layer to provide user actuation.

The first interaction the user has with our system is a form on a server-hosted website. The user selects an origin and destination, which is fed into a Python script that uses Google Maps API and a built-in physics model to generate an energy plan. This energy plan, which contains data points in the form of a text string, is loaded onto an SD card and uploaded to an Arduino microcontroller, which is mounted on the bicycle. The Arduino uses a Hall Effect sensor attached to the rim of wheel along with a battery monitor to gather real-time data on the user's power draw, distance traveled, voltage, current and velocity. It then compares the real-time energy data with stored plan data. Based on this comparison, a colored LED feedback indicates whether the user is above, below, or roughly on track with the plan.

II. INTRODUCTION

Motivation & Background

Bikes are a cheaper more environmentally friendly means of transportation than automobiles, but they do not provide the same level of convenience and comfort that conventional vehicles do. Additionally, bikes have traditionally been targeted to only an athletic audience. Those who want increased mobility but may not be in circumstances that allow them to physically active would not be able to benefit from normal bikes.

EBikes have stepped in to bridge this gap by offering increased performance and comfort, while maintaining the maneuverability and low cost that regular bikes offer. Because of these performance characteristics, eBikes are making up a larger and larger share of the mode-split of our transportation system. This shift toward bikes and away from cars provides compounding benefits outside the scope of this course.

Due to superior performance characteristics as compared to traditional bikes and lower cost compared to cars, eBikes are becoming an increasingly popular mode of transportation. Ideally, this trend will continue, but there are several significant obstacles to the widespread adoption of electric bikes.

From the addition of an electric motor and battery pack, eBikes allow people to travel greater distances or on more strenuous terrain. Not all of the additional features provided by the e-assist hardware are beneficial. One of the main drawbacks of an eBike setup like this is the excess weight of the electronics that power it, which is not felt by the user under most circumstances. However, this excess weight means that if the rider runs out of charge, the once-useful e-assist system becomes a burden. A user with a "dead" eBike is essentially worse off than one whose bike is initially unpowered. Thus, it no longer becomes attractive to ride an eBike without a way to manage battery power.

Our solution is to optimize human energy and depletion of battery power for an eBike based on a given route. Thus, bikers will not have to worry about a depleted battery pack during their trip and can travel safely to their destination.

Relevant Literature

Moura, S. et al. Tradeoffs between battery energy capacity and stochastic optimal power management in plug-in hybrid electric vehicles. J. Power Sources, 195, 2979–2988 (2009).

Katsargyri, G.E. et al. Optimally Controlling Hybrid Electric Vehicles using Path Forecasting, American Control Conference 2009.

Martin, James C. et al. Validation of a Mathematical Model for Road Cycling Power. Journal of Applied Biomechanics, **14**, 276-291 (1998).

Cook, John D. "Computing the distance between two locations on Earth from coordinates." Singular Value Consulting. Web.

O'Keefe, M.P. & Markel, T. "Dynamic Programming Applied to Investigate Energy Management Strategies for a Plug-in HEV." National Renewable Energy Laboratory. Hybrid Electric Vehicle Symposium and Exhibition (EVS-22). 2006.

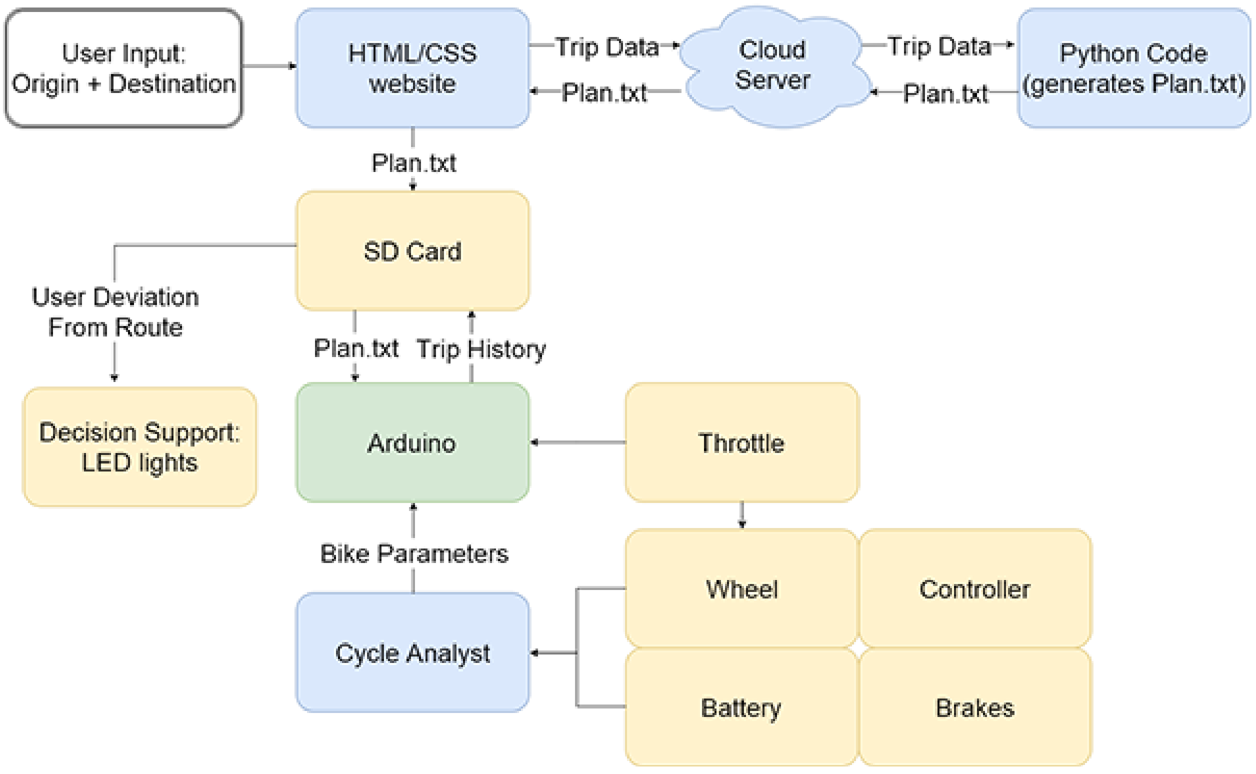
Appelkvist, O. & Gebel, P. Route Predictive Optimization of State-of-Charge Reference Signal. Department of Signals and Systems. Chalmers University of Technology. 2010.

Focus of Study

The project goal is to create and communicate to the user an energy usage plan for a trip between two locations. The plan will ideally balance electric and leg power to maintain rider comfort while ensuring that the destination is reached successfully. Using a real-time feedback system, this project will encourage usage of a human-electric hybrid mode of transportation.

III. TECHNICAL DESCRIPTION

CPS Architecture Schematic



This diagram demonstrates the cyber (blue) and physical (gold) layers of our CPS. The system starts with the cyber component: user input → website → cloud server → Python code → plan.txt → SD card

The information in the plan.txt file in the SD card is then incorporated into hardware components. The Arduino attached to the system compares wheel and battery data from the data monitor Cycle Analyst (as trip history) with the data in plan.txt. Based on the user deviation from the route, the LED lights will blink with the corresponding actuation.

Bill of Materials

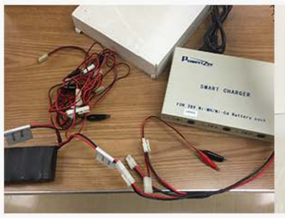
Bike components:

1. EBike kit
    - b. Controller
    - c. Throttle
    - d. Electronic Brake
    - e. Ni-MH batteries
    - f. Battery Charger
    - g. Associated Cables
    - h. Inline Fuse
    - i. Wire Connectors

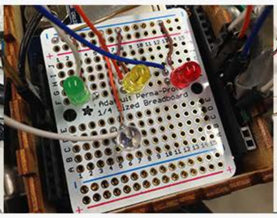
2. Bike rack
3. Laser-cut components for battery support
4. Cycle Analyst (data monitor)
5. Zip ties
6. Two laser-cut boxes

Arduino:
1. Protoboard
    a. SD card
    b. LEDs (white, red, yellow, green)

HARDWARE


Battery Controller

Battery Components


LED Actuation

Bike Components

BIKE

Software
1. Arduino Code
    a. CycleDataLogger.ino
    b. Actuation.ino
2. Python Code

   a. gMapsAPI.py
   b. nanoserver.py
3. HTML Code
   a. index.html
4. JS Code
   a. bootstrap.js
   b. bootstrap.min.js
   c. custom.js
   d. jquery.easing.min.js
   e. jquery.min.js
   f. jquery.scrollTo.js
   g. jquery-1.10.2.js
   h. wow.min.js
5. CSS Code
   a. animate.css
   b. bootstrap.css
   c. bootstrap.min.css
   d. style.css
   e. default.css

<u>CycleDataLogger1115</u>
This is the code that we used to collect and log data from the Cycle Analyst:

- Setup
    - Initializes Serial, SD
    - Performs Various Diagnostic Tests
    - Creates "data.txt" file on SD card
    - Timestamps beginning of run
- Loop
    - Uses Serial event to collect strings from CycleAnalyst as they come in
    - Timestamps each incoming string
    - Writes String to SD Card

<u>Actuation</u>
This is the code that takes the real-time data from the trip, which is compared against the plan.txt file:

- LED pins initialized
- Global Variables which must persist across loops are initialized
- planVals array is initialized
- Setup
    - LED pins as OUTPUT
    - Begin Serial, SD
    - Open "plan.txt"
    - Call function parsePlanToArray()
    - Flash all LEDs for style points
- Loop
    - Grab string from CycleAnalyst with serialEvent()

    - Read AmpHr and distance from string with readStr()
    - Actuate on results of readStr with actuate()
        - Red = low on battery = pedal more
        - Yellow = matching plan = continue
        - Green = excess battery = pedal less
        - White = use throttle

<u>gMapsAPI1207</u>
This is the most recent version of our Python code, which generates and sends to the server an energy plan. Following is an outline of the program.

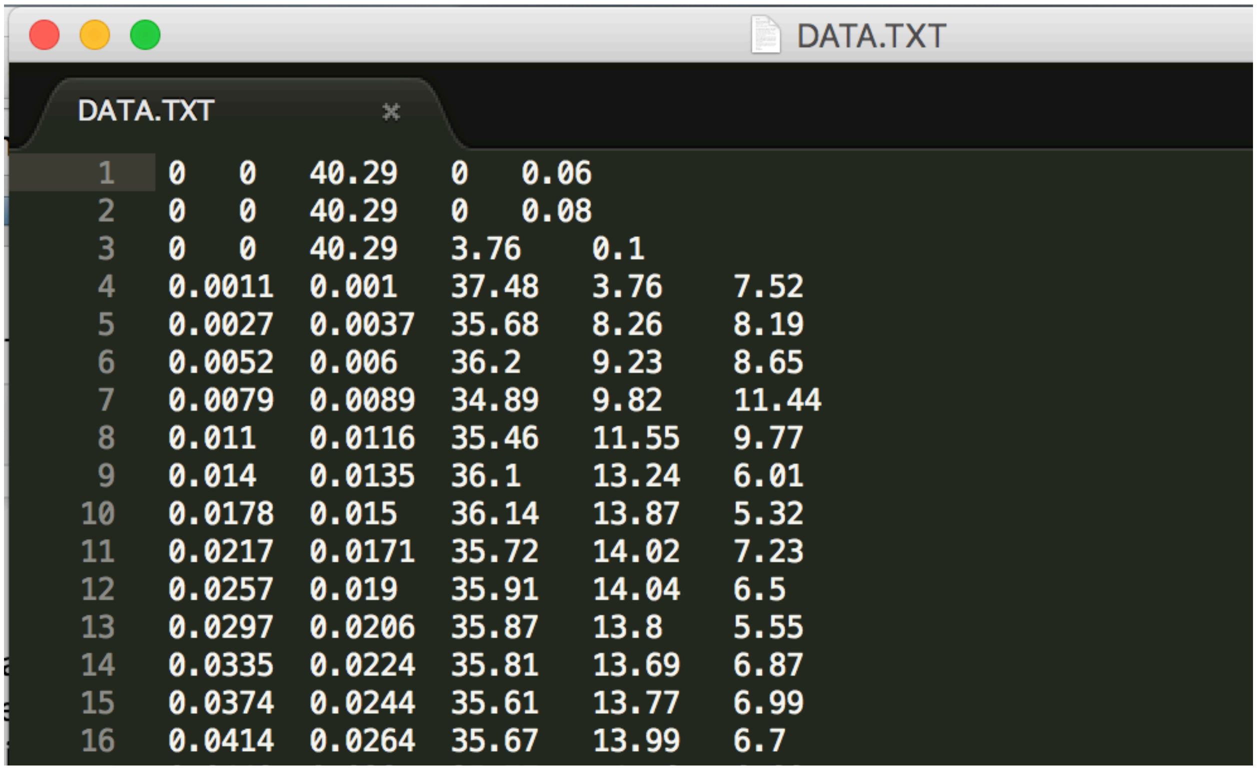
- listenFromServer waits for the origin and destination to be sent from the javascript AJAX request, and then sets them to two variables, imHere and goHere
- An HTTP request is then made to the google maps API using the two points, returns a dictionary containing the various legs of the trip.
- An array containing polylines corresponding to each leg of the trip, as well as one with the distance of each of these legs is created.
- Another HTTP request to the google maps API is made using each of the polylines and sufficient samples to result in a sample roughly every 50m, returns a dictionary containing sampled elevation data.
- An arc length calculation is performed for each elevation point to calculate the distance between consecutive points, these will be used as the x values for elevation points.
- Physics model is then applied to a linearly spaced subset of the distance,elevation pairs. This results in a monotonically increasing array of energy vs. distance.
- Several things are plotted:
    - Elevation v. Distance
    - Slope v. Distance
    - Energy v. Distance
- Energy Plan ('plan.txt') is written to a text file on the desktop.

<u>index</u>
This is main HTML file that serves as the foundation of our website. The first part of the code sets up the layout of the website as one page that allows the user scroll through the different sections. index.html calls several other CSS and JS files for added functionality and design.

The second part of the script is the JavaScript code that communicates with the server

- When the user clicks the submit button, the form data is posted to the server as a JSON string
- The code listens for the network data, which is sent as different streams, and parses through them and converts them to integers
- Stream icons are loaded with their corresponding data streams
- Using Highcharts, the data for elevation and energy are mapped as separate plots on the page and labeled

Data Collection
Below is an example of cycle data logged in the SD shield:



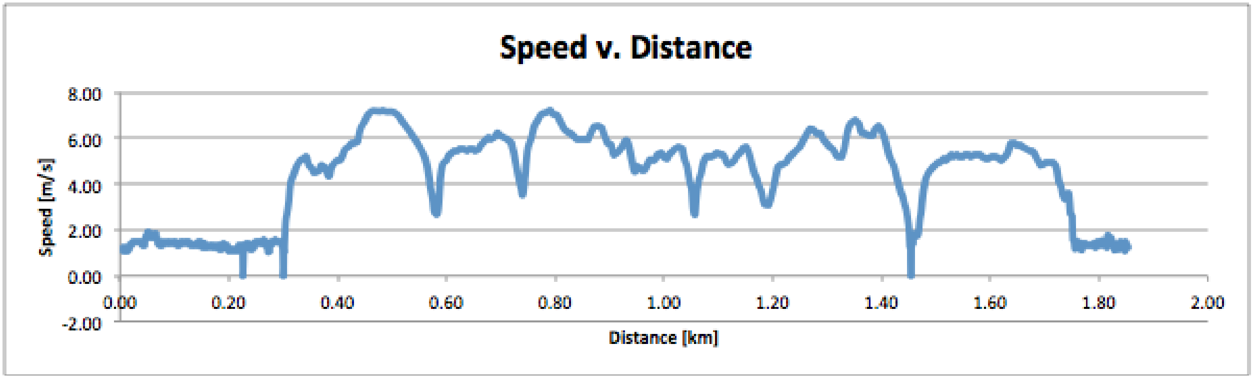
```
0       0       40.29   0       0.06
0       0       40.29   0       0.08
0       0       40.29   3.76    0.1
0.0011  0.001   37.48   3.76    7.52
0.0027  0.0037  35.68   8.26    8.19
0.0052  0.006   36.2    9.23    8.65
0.0079  0.0089  34.89   9.82    11.44
0.011   0.0116  35.46   11.55   9.77
0.014   0.0135  36.1    13.24   6.01
0.0178  0.015   36.14   13.87   5.32
0.0217  0.0171  35.72   14.02   7.23
0.0257  0.019   35.91   14.04   6.5
0.0297  0.0206  35.87   13.8    5.55
0.0335  0.0224  35.81   13.69   6.87
0.0374  0.0244  35.61   13.77   6.99
0.0414  0.0264  35.67   13.99   6.7
```

This data required significant cleaning for a variety of reasons. For example, any point at which the bike was not moving was not useful for analysis. As we later found, points at which the bike was moving downhill are also not useful and the physics model did not apply. Due to the fact gravity is doing a significant amount of work on the bike-rider system, there is no power required of the motor during this time.

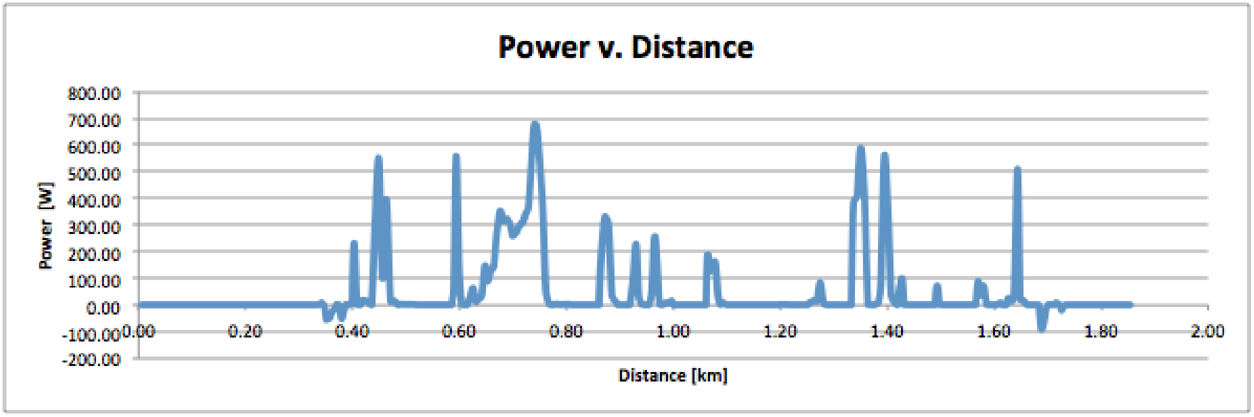
It turns out that Cycle Analyst was choking the power output from the battery, so there was not enough power to run the SD shield at the same time as the real time clock embedded in the shield. This forced us to plot all of our parameters against distance instead of time. It is unclear how this would have changed our analysis, but for one thing it would have been easier to see where any stops occurred during the run. Additionally it would have allowed us to more accurately recall landmarks in the run, which would have aided data analysis.

Following are examples of graphs used to visualize the parameters collected for each run. This data comes from the route from Jacobs Hall to Clark Kerr Campus.

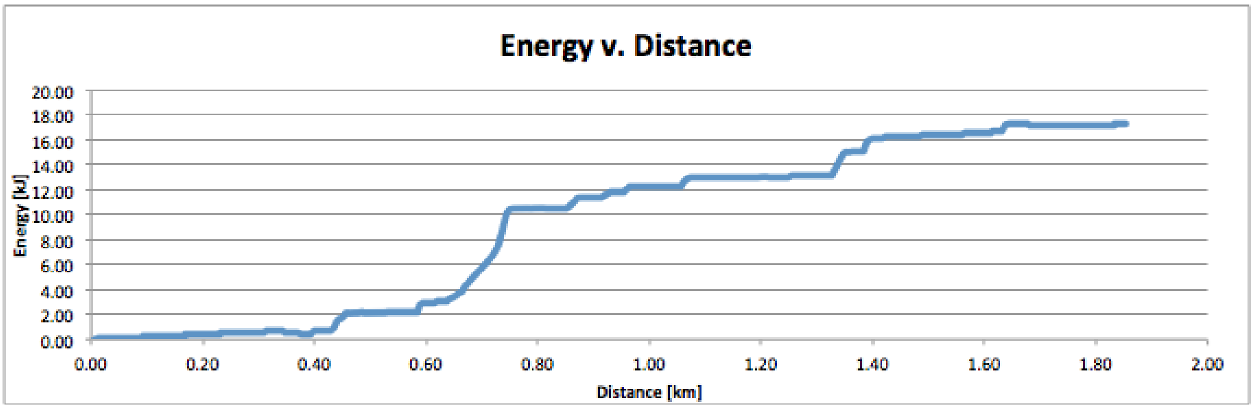
Speed is converted from rpm by the Cycle Analyst.



Power is calculated as the product of instantaneous current and voltage.



Energy is calculated by taking a running sum of the instantaneous power values. Data is collected at a rate of 1Hz, so summing power values each second results in energy consumption. We attempted to fit our model data to this estimated energy use.



Data Analysis - Physics Model
For modeling energy use for the eBike, we chose three main forces to take into account: force due to gravity, rolling resistance, and air drag.

$$F_{grav} = 9.8 \times \sin(\arctan(slope)) \times weight$$

$$F_{roll} = 9.8 \times \cos(\arctan(slope)) \times weight \times C_{roll}$$

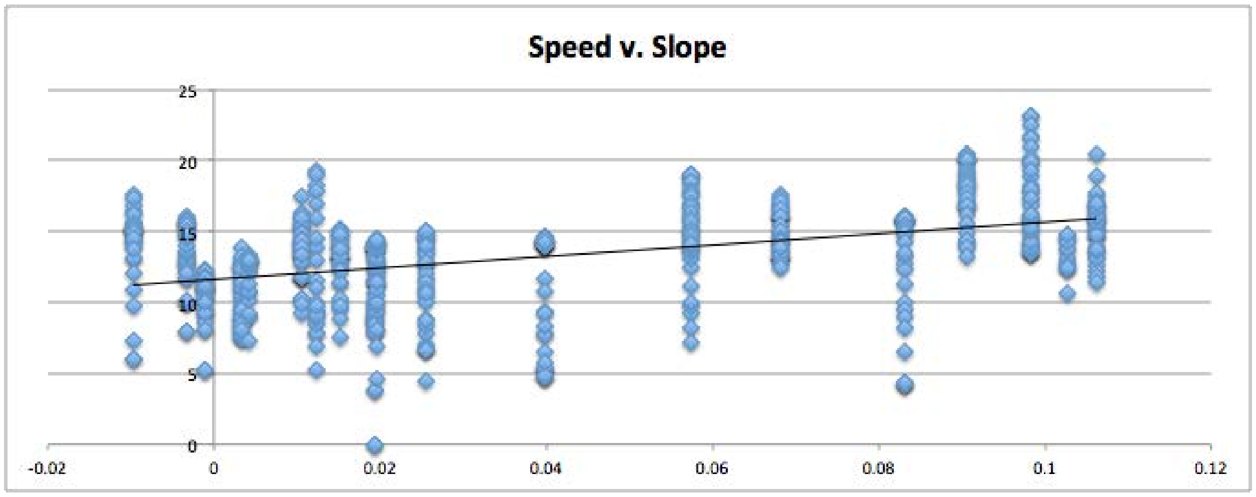
$$F_{drag} = 0.5 \times \left(C_{drag} \times A_{front}\right) \times \rho \times v^2$$

After calculating an array of slopes using Google Maps API Directions data we calculated power as follows:

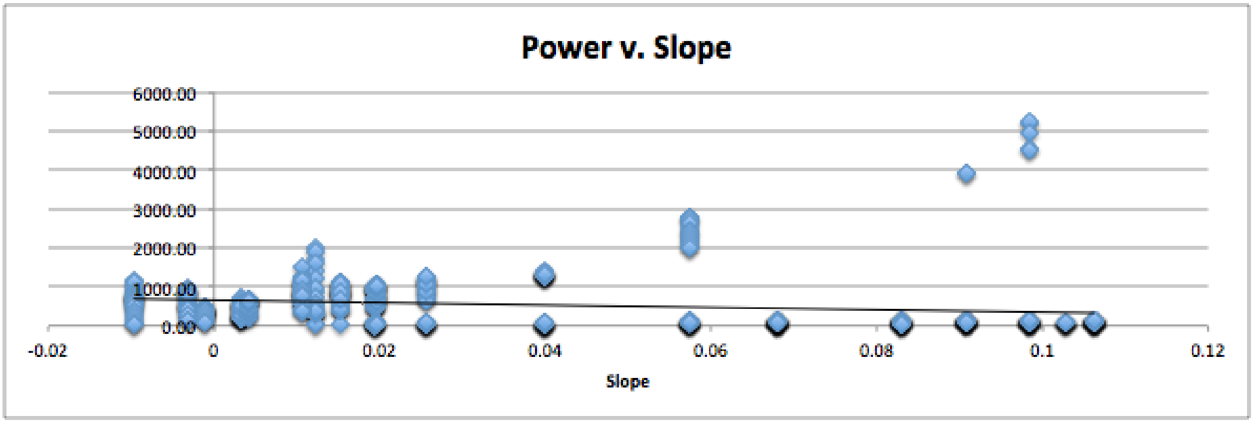
$$Power = \left(F_{grav} + F_{roll} + F_{drag}\right) \times v \times \frac{1}{\eta_{drivetrain}}$$

For the purposes of our analysis we assumed that rider output was small and constant, but adding variation to this model would be ideal to most accurately predict power output.

Using speed and slope, as well as literature values for coefficients, we estimated the energy that would be required by the bike to traverse a route and formulated a battery depletion plan. Estimating a speed parameter proved to be a challenge, so we used regression to calculate a likely speed based on the slope. Much of the data we collected was difficult to interpret, for example speed seems to increase with increasing grade in this case. For ease of analysis we have discretized slope values into several bins.



Paradoxically, our data illustrates a slightly negative relationship between power and slope. Perhaps this is just due to our selection of run.



Collected data did not fit physics model initially. With each data collection, we learned more about our parameters of interest and which terrain would provide useful data that we could use to fine tune our physics model. There were opportunities to be more rigorous in conducting experiments to isolate variables. Because each user rode the bike differently, there were significant confounding variables, with more time we could have more effectively controlled for these effects. For example riding the same route, with the same degree of reliance on the throttle might have yielded useful results. Varying the degree to which we relied on the throttle for identical runs would also have produced interesting data. As one might expect, there are innumerable ways that we could have teased meaning out of the raw data.

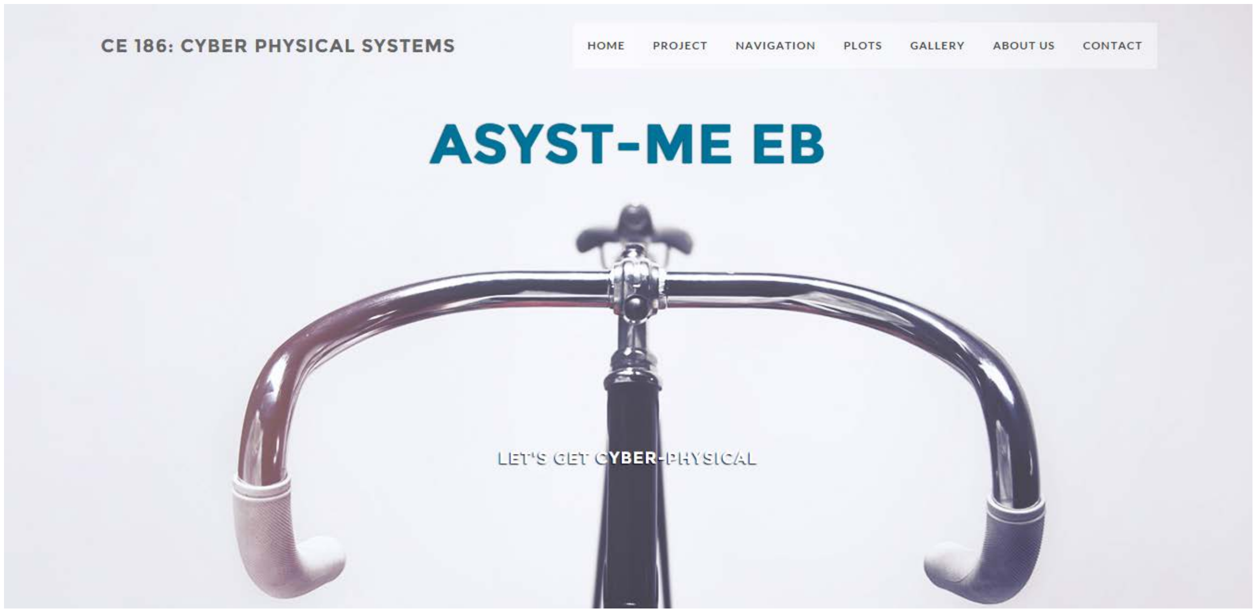
Looking forward, an interesting avenue of exploration would be to try to more accurately account for human effort in the power calculations. At around 100 watts, the contribution of leg power is not insignificant. Allocating this power most efficiently will allow us to increase rider comfort while minimizing battery output.

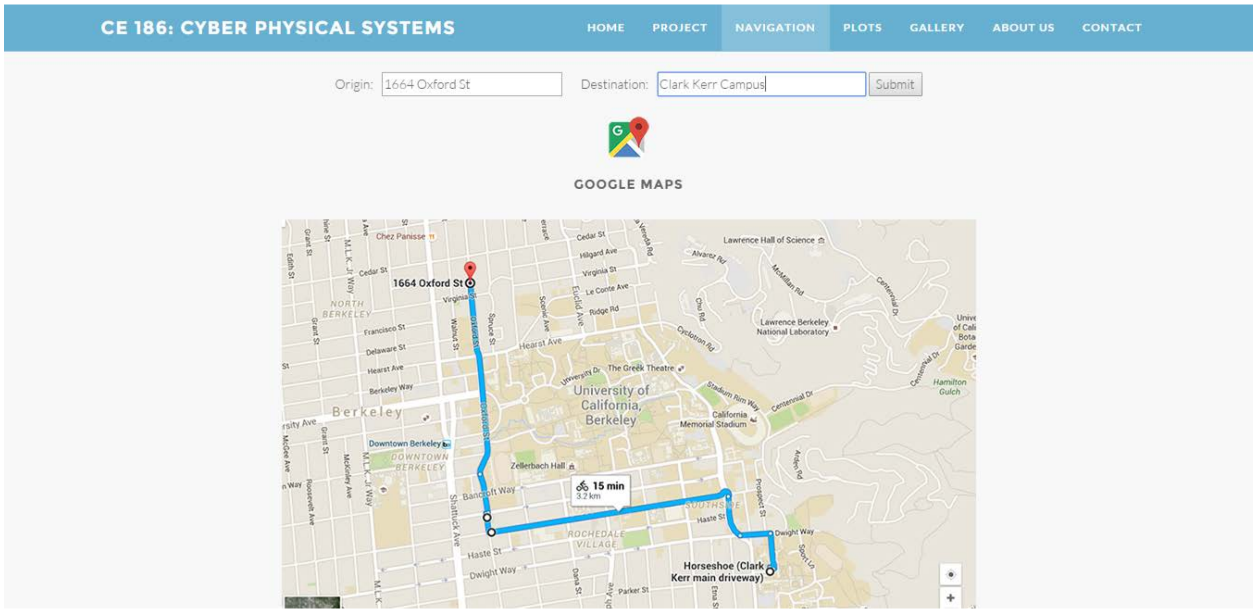
Visualization

To visualize our data, we chose to create an interactive website. The website was created from Squadfree, a free HTML template online. The dynamic functionality was included with the Bootstrap 3 framework. All the pages were customized by the team to best represent all the components of our project. The sections of our website are listed below:

- Project: A quick overview of the problem we are trying to solve and our solution
- Navigation: User form to submit Origin and Destination to the server and access the Python code. There is also an embedded Google Map that we intended to display the route. We were able to embed just a coordinate view of Google Map with no specific location, but our intention was to have a dynamic map that would show the directions based on user input.
- Plots: These plots show Energy vs. Distance as well as Elevation vs. Distance graph based on user input. If there are other parameters, additional plots would be mapped here according to the network data received.
- Gallery: Images of the project hardware.
- About Us: Images of our team members.

- Contact Us: A simple form that would be used in a hypothetical situation to receive emails but not actually connected to anything.



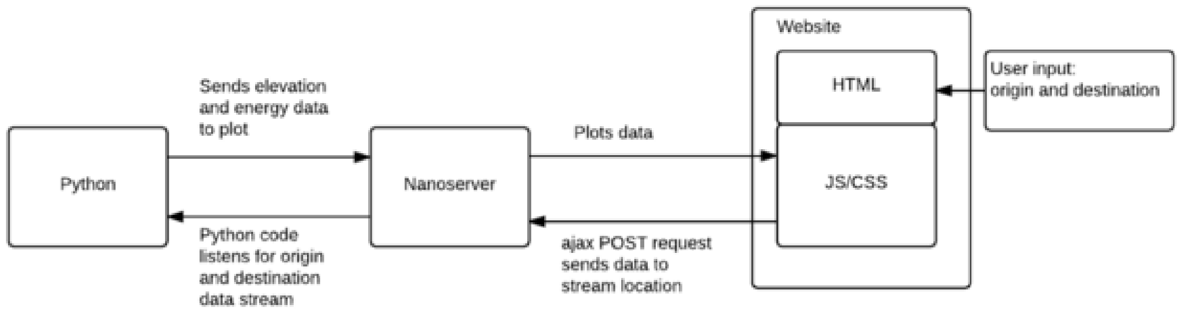

CE 186: CYBER PHYSICAL SYSTEMS
HOME
PROJECT
NAVIGATION
PLOTS
GALLERY
ABOUT US
CONTACT
Submit
GOOGLE MAPS
1664 Oxford St
15 min
3.2 km
Horseshoe (Clark Kerr main driveway)
University of California, Berkeley

Origin: 1664 Oxford St

Destination: Clark Kerr Campus

The key component is the Navigation tool, which takes the user input and communicates with our server. The server calls our Python code and makes a GoogleMaps API request based on an origin and destination provided by the user. From GoogleMaps, elevation and distance data were collected and the physics model we programmed into the code output an energy plan. The elevation and energy data streams generated from the Python code were sent separately to the server, which then posted the graphs to our website under the Plots tool.

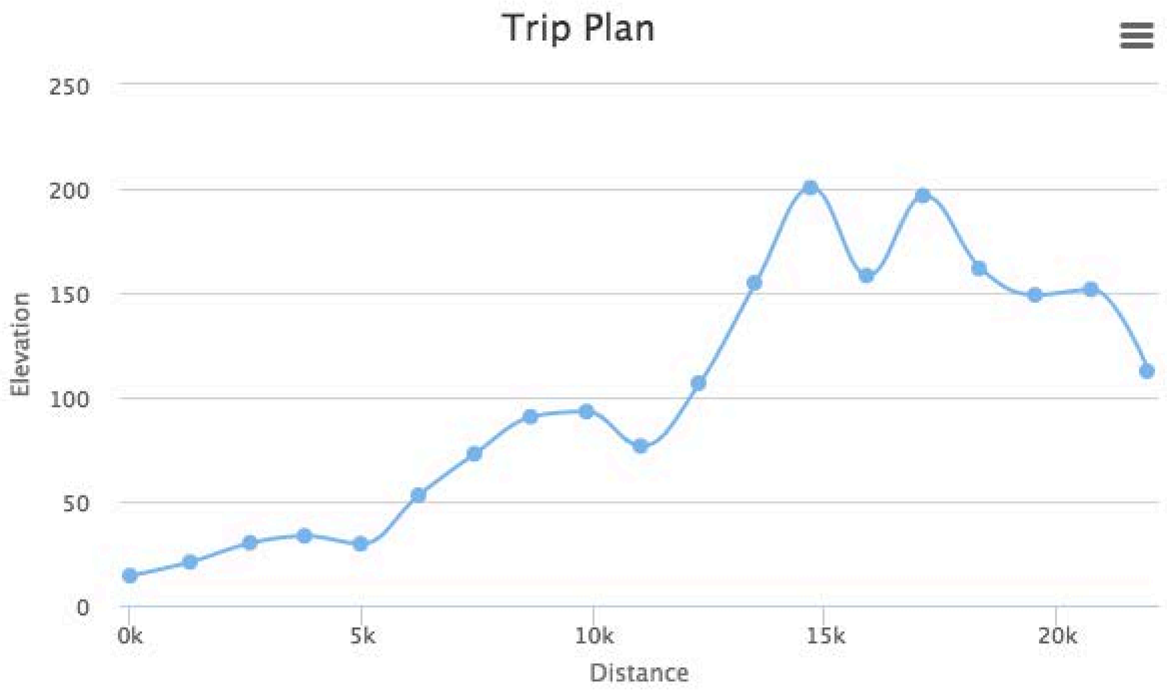
Below is a flow chart for Web Visualization

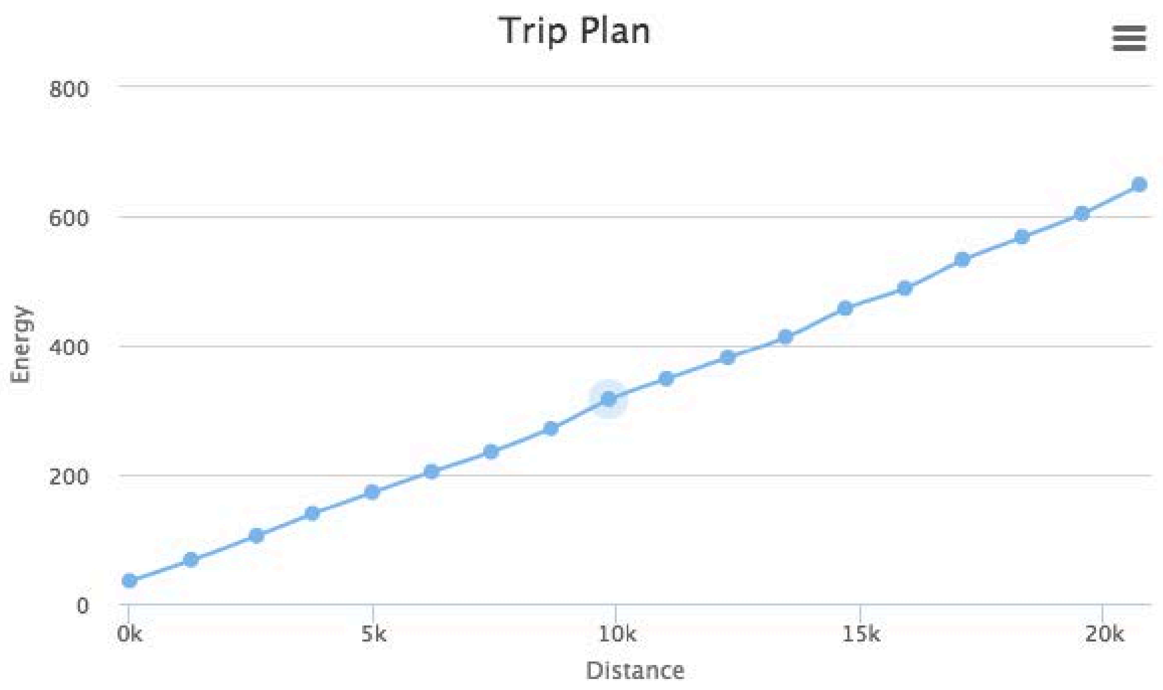

Website
HTML
User input: origin and destination
Sends elevation and energy data to plot
Plots data
Python
Nanoserver
JS/CSS
Python code listens for origin and destination data stream
ajax POST request sends data to stream location

- First off the server (nanoserver.py) is run on a python console
- Next the python code (gMapsAPI.py) is run on another python console. It begins listening for data from the server, and waits until something is found.
- Meanwhile, the user inputs addresses for the desired origin and destination into the html forms
- The Javascript code makes an ajax 'POST' request to send data to the server to the specified stream url. In addition, it prevents the website from redirection to the url.
- The origin and destination data (data type: string) are now in the server
- gMapsAPI.py now successfully finds the data it has been listening for from the server. It takes the origin and destination addresses and uses them to generate elevation vs. distance and energy vs. distance plots.
- gMapsAPI.py then sends the elevation and energy data points to the server and the Javascript file then plots the graphs onto the website.

The following graphs are the plots produced by HighCharts about the trip data with an example route.

Origin: Downtown Berkeley BART
Destination: Clark Kerr Campus


Trip Plan
Elevation
250
200
150
100
50
0
0k
5k
10k
15k
20k
Distance


Trip Plan
Energy
800
600
400
200
0
0k
5k
10k
15k
20k
Distance

IV. DISCUSSION

aSYST-ME eB solves one of the most inherent problems of battery energy storage system in transportation industry, which is limited energy capacity gives you limited miles and the sole purpose of any vehicle is to get the user to its destination. As discussed earlier, if the battery runs out of energy, the extra added weight of the hardware system on an eBike compared to a regular bike makes it a burden and a worse option.

However, aSYST-ME eB with its smart nexus of software and hardware, gives the user feedback of the eBike state of charge in relation to energy use while on the go. With the help of its user friendly web interface, it also provides user a beforehand graphical data of elevation change as well energy use expected over the route to destination. By doing so, it not only acts as an early warning system for any sudden charge depletion due to system malfunction but also makes sure user reaches the destination in the most optimal manner. An aSYST ME-eB driven eBike broadens the spectrum of customers from avid bikers to anyone who wants to adopt a green and clean means of transportation. This can include commuters who use bikes to travel long distances with least amount of human energy used to get to their destination and recreational bikers who wants to enjoy their decreased carbon footprint worry-free.

V. SUMMARY

The project addresses one of the major issues with electric hybrid modes of transportation: rate of battery depletion. 4 of the 5 pillars of CPS are fully developed in this project. Infrastructure is well integrated with sensing and actuation hardware, in the form of an eBike, Cycle Analyst, and Arduino/LEDs. Data visualization is done on an HTML/CSS/JS platform, and all these parts are connected together by a combination of HTTP requests and SD file transfers.

Looking to the future, we would streamline the user experience. Additional data collection would enhance the accuracy of our mathematical model and predict SOC based on user determined physical parameters. We would also consider adding other factors into the user input, such as “sweatiness factor” or other fitness options for a more customizable system. In the end, this eBike will be an optimized human hybrid electric system that aims to help eBike users complete their trip without worrying about running out of battery.