

Motivating Urban Cycling Through a Blockchain-Based Financial Incentives System

by

Caroline Adair Jaffe

B.S., Yale University (2013)

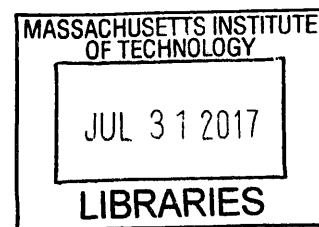
Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

© Massachusetts Institute of Technology 2017. All rights reserved.



Signature redacted

Author
..... Caroline Adair Jaffe
Program in Media Arts and Sciences
..... May 12, 2017

Signature redacted

Certified by
..... Sepandar Kamvar
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
..... Thesis Supervisor

Signature redacted

Accepted by
..... Pattie Maes
Academic Head, Program in Media Arts and Sciences

Motivating Urban Cycling Through a Blockchain-Based Financial Incentives System

by
Caroline Adair Jaffe

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on May 12, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

As cities become increasingly dense in the coming decades, they must turn to novel technologies and frameworks to address the imminent environmental, mobility, and public health issues that will arise with this population shift. The overwhelming use of single occupancy vehicles in the United States - they account for 76% of all trips - is a major contributor to pollution, traffic, and sedentary lifestyles. However, 50% of trips in the U.S. are less than 3 miles, and could likely be replaced by a more sustainable and space-efficient mode of transportation, such as bicycling, if effective policies and incentives were implemented.

This thesis presents a blockchain-based financial incentives system where cyclists can leverage their activity and location data to receive financial compensation from organizations that would like to sponsor cycling activity. For example, an insurance company may want to reward its customers with lower premiums for partaking in healthy commuting behavior. A city government may wish to encourage cycling activity to mitigate urban congestion and pollution. A local business may sponsor bicycling activity in its vicinity to increase sales. The system presented in this thesis allows these organizations to internalize the positive externalities of cycling that have not historically been recognized or rewarded.

This incentives system uses GPS data from sensors affixed to bicycles frames and powered by the cyclists themselves. The use of blockchain technology makes transactions in the marketplace secure, seamless, trustworthy, and transparent. Users are able to reveal "just enough" information about themselves to participate in the decentralized marketplace, instead of exposing their entire profile to a central entity. This market-driven system facilitates better matching between individuals and incentives, and delivers those incentives in a more timely, effective manner than current incentives programs. This thesis also envisions expanding this platform to include additional bicycle-based sensors that cyclists can leverage to collect and sell data, monetizing their commuting habits, and contributing to a scalable and stable solution for increasing the use of sustainable transportation in cities.

Thesis Supervisor: Sepandar Kamvar

Title: Associate Professor of Media Arts and Sciences

Motivating Urban Cycling Through a Blockchain-Based Financial Incentives System


by
Caroline Adair Jaffe


Signature redacted

Thesis Reader

.....

Joseph Paradiso

 Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Signature redacted

Thesis Reader

Neha Narula

Research Director, Digital Currency Initiative
MIT Media Lab

Acknowledgments

I would like to express my deep gratitude to my advisor, Sepandar Kamvar, for sharing his beautiful vision of the way we should build technologies, for trusting me unconditionally, for the opportunity to join the amazing community at the Media Lab, and for the resources and support he's provided over the past two years. To my readers, Joe Paradiso and Neha Narula: thank you for your time, expertise, and encouragement. Joe, I have enjoyed and been inspired by our discussions on the potential of urban sensor technologies, and I look forward to fruitful explorations of these ideas during my Ph.D. Neha, thank you for your willingness to explore this unconventional thesis with me, for helping me reach a more nuanced understanding of blockchain technology, and for your detailed feedback on my writing.

To the inimitable Kim Smith: thank you for being someone I could commiserate and conspire with; this has been a challenging year, but your day-to-day encouragement and friendship have kept me going, and made the time that much sweeter. Kristina, thank you for your friendship, for standing up for your students, and for your insights on the Lab's inner workings. Amanda, thank you for supporting us and making sure things ran smoothly this last semester. To the Social Computing group, thank you for welcoming me into your world and sharing your many talents. And finally, thank you to the incredible UROPs who worked on this project with me (Cristina Mata, Kathleen Zhu, and Chitti Desai): I sincerely appreciate your hard work and willingness to explore new ideas and technologies.

To my friends at the Media Lab (Ariel, Miguel, Bianca, among many others): thank you for making the Lab a warm place where I was happy to spend an inordinate amount of time. To my friends and roommates beyond the Media Lab (Shirlee, Courtney, Laura, Aviva, Jacob, Astrid, Geoffrey, Wayne, David Wang, Nathan, Starry, the amazing women of salon, and many others near and far): thank you for being tolerant of my often unreasonably busy schedule, exercising with me, exposing me to challenging new ideas, sampling the cultural and culinary offerings of Cambridge with me, and generally looking out for my sanity and health. You are an amazing support network and I am deeply indebted to your friendship.

To the MAS staff, particularly Keira, Monica, and Linda: thank you for your mentorship, organization, advocacy, and sense of levity. You make me feel that I belong here and give me confidence that there is someone responsible, fair, and kind steering the ship.

To my boyfriend, Sam: thank you for standing beside me on this journey. You inspire and challenge me, and have made me feel valued in times of success and failure. You have been there for me at every step of this process, often in less-than-glamorous capacities, like proofreading my work or bringing me take-out food at the Lab. I feel profoundly lucky to have found someone with whom I resonate in so many areas of my life and I am deeply grateful for our relationship.

To my family (Elisabeth, Howard, and Matt): the past two years have been difficult ones for our family, and I especially appreciate you holding things together at Jaffe headquarters so that I could focus on my classes, research, and building a life for myself in Cambridge. From rapid turn-around application editing, to help with my taxes, and world's best care packages, you have made me feel supported and valued. Thank you for a lifetime of unconditional love and trust, for encouraging me to dream big, and convincing me that I could do anything I wanted with hard work and positive energy.

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Contribution	8
1.3	Overview	9
2	Background & Related Work	10
2.1	Bicycling & Urban Studies	10
2.1.1	Positive Externalities of Cycling	10
2.1.2	Improving Cycling Mode Share	11
2.2	Behavioral Psychology	15
2.2.1	Nudge Behavioral Studies	15
2.2.2	Nudging with Financial Incentives	16
2.2.3	Financial Incentives & Transportation	17
2.3	Blockchain	17
2.3.1	Advantages & Typical Applications	18
2.3.2	Emergent Applications & IoT Work	19
2.3.3	Ethereum	20
3	Approach	21
3.1	Bicycles & Behavioral Psychology	21
3.2	Why Blockchain?	22
4	Hardware Procedures	24
4.1	Sensor Design	24
4.1.1	System Components	24
4.1.2	Device Power Requirements	27
4.2	Device Preparation & Assembly	29
4.2.1	Raspberry Pi Software Installations	29
4.2.2	Sensor Assembly & Deployment	29
5	Software Procedures	30
5.1	Financial Incentives Design	30
5.2	System Architecture	32
5.2.1	Blockchain Network Design	33
5.2.2	System Components	34
5.3	Blockchain Implementation	36

5.3.1	Ethereum Network Operation & Design	36
5.3.2	Node Discovery	38
5.3.3	Smart Contracts	39
5.3.4	Browser-Based User Interface	42
5.4	Embedded Software Implementation	43
5.4.1	Ethereum Node Initialization	43
5.4.2	GPS Data Collection	44
5.4.3	Blockchain Network Communication	45
6	Development, Deployment & Evaluation	46
6.1	Development	46
6.2	Deployment & Testing	47
6.2.1	Assumptions	47
6.2.2	Procedure & Testing	47
6.3	System Evaluation	50
7	Conclusion	53
7.1	Contribution	53
7.2	Future Work	54
7.2.1	Technical Improvements	54
7.2.2	User Evaluations	55
7.2.3	Future Research	55
A	Installing an Ethereum Client & Running a Private Ethereum Test-	
	net	58
A.1	Installing an Ethereum Client	58
A.1.1	Desktop Installation	59
A.1.2	Raspberry Pi Installation	59
A.2	Running a Private Ethereum Testnet	59
A.2.1	Genesis Block & Starting First Node	59
A.2.2	Starting Second Node & Adding Network Peers	62
B	Smart Contracts	64
B.1	Contract Behavior	64
B.2	Truffle Development Framework	65
B.3	Contract Implementation	66
B.3.1	Writing Contracts	66
B.3.2	Testing Contracts	68
B.3.3	Creating & Deploying Contracts	69
C	Raspberry Pi Configuration & Usage	72
C.1	Raspberry Pi Single-Board Computer	72
C.2	Installation & Preparation	73
C.2.1	Using the Raspberry Pi with the Adafruit FONA	73
C.2.2	Using the Raspberry Pi as an Ethereum Node	74
C.3	Ethereum Node Initialization Scripts	74
C.4	GPS Data Collection	77

Chapter 1

Introduction

1.1 Motivation

The United Nations predicts that urban populations will nearly double by mid-century, growing from 3.4 billion in 2009 to 6.3 billion in 2050¹. As cities become increasingly dense, they must turn to novel technologies and frameworks to address the imminent environmental, mobility, and public health issues that will arise. In particular, Single Occupancy Vehicles—that is, people driving alone—which make up 76% of trips in the U.S., are a major contributor to pollution, traffic, and sedentary lifestyles. However, there is room for change. 50% of trips in the U.S. are less than 3 miles, and could likely be replaced by a more sustainable mode of transportation, such as bicycling, if effective policies and incentives were implemented². Bicycling is a cheap, healthy, and space efficient mode of transportation that could have beneficial economic and health impacts were it to be more widely adopted³.

While transit mode choice is affected by environmental features of cities, such as weather and geography, it is also influenced by policy, infrastructure, and individual qualities such as attitudes and habits⁴. This thesis presents a system that intends to increase and encourage the choice of cycling through the use of financial incentives. In doing so, this work draws not only upon established research in transit mode choice, but also on a group of behavioral economics studies concerned with "nudging" behaviors. In nudge studies, minor changes in an individual's choice architecture are designed to push them towards a certain behavior without excessively limiting their

¹"World urbanization trends 2014: Key facts." Statistical Papers - United Nations (Ser. A), Population and Vital Statistics Report World urbanization prospects (2014): 1. Web. 12 Mar. 2017.

²Flusche, Darren. "National Household Travel Survey - Short Trips Analysis." www.bikeleague.org. The League of American Bicyclists, 22 Jan. 2010. Web. 2 Nov. 2016.

³Mason, Jason, Lew Fulton, and Zane McDonald. "A Global High Shift Cycling Scenario: The Potential for Dramatically Increasing Bicycle and E-bike Use in Cities Around the World, with Estimated Energy, CO2, and Cost Impacts." Institute for Transportation and Development Policy. 12 Nov. 2015. Web. 4 Oct. 2016.

⁴Heinen, Eva, van Wee, Bert, & Maat, Kees (2010). Commuting by Bicycle: An Overview of the Literature. *Transport Reviews*, 30(1), 59-96. <http://doi.org/10.1080/01441640903187001>.

options⁵. The protocol presented in this thesis nudges individuals towards cycling by paying them to use a bicycle for transportation.

In this thesis work, the incentives for cyclists come from organizations—such as city governments, health insurance companies, or local businesses—that would like to encourage sustainable transportation behavior because increased cycling will result in a tangible benefit for the organization itself. For example, city governments might want to sponsor cycling because it mitigates traffic and urban pollution. Health insurance companies might want to sponsor cycling because it makes their clients healthier. Local businesses might want to sponsor cycling because it improves their retail profits. While the positive economic and health benefits of cycling are relatively well-documented, they are rarely rewarded outright. Our framework allows organizations to internalize the positive externalities of urban cycling; in doing so, we seek to build a system that provides strong incentives to both cyclists and organizations. Thus, the motivation behind this thesis is not just to increase the proportion of cyclists in cities, but to do so in a way that is self-sustaining and engaging.

The technical implementation of this system draws on two technologies that have exploded in the past decade: mobile location tracking, and blockchain databases. In this system, a GPS sensor—that is indirectly powered by a bicycle—collects cyclists' location data and cycling activity. Through a decentralized application running on a blockchain database, organizations such as health insurance companies, advertisers, or local businesses can reward cyclists for their cycling activity. These payments incentivize cyclists to ride more, but also benefit the organizations distributing them. The use of a blockchain database allows cyclists and organizations to exchange anonymous payments without a trusted intermediary—a so-called "trustless" protocol—in a robust, transparent, and accessible network⁶. This market-driven system facilitates flexible, responsive incentive matching. Though it was not explicitly addressed in our work, this thesis envisions the scaffolding of a more expansive data exchange platform, where bicycles are equipped with additional sensors (e.g. air quality, road quality, traffic) that collect geo-located data that cyclists can store and eventually monetize. In this view, the incentives system and blockchain application comprise a scalable micro-entrepreneurship ecosystem that encourages the use of sustainable transportation in cities.

1.2 Contribution

The contributions of this thesis are to: (1) design and analyze the blockchain protocol that supports this incentives system; (2) build a proof-of-concept implementation of the GPS sensor and blockchain application; and (3) evaluate the protocol from a technical standpoint. This thesis builds upon our knowledge, from nudge behavior studies, of the way people respond to financial incentives, as well as real-world exam-

⁵Leonard, Thomas C. "Richard H. Thaler, Cass R. Sunstein, Nudge: Improving Decisions about Health, Wealth, and Happiness." *Constitutional Political Economy* 19.4 (2008): 356-60. Web.

⁶Tyle, Sheel, and Mohit Kaushal. "The Blockchain: What It Is and Why It Matters." Brookings. N.p., 13 Jan. 2015. Web. 03 Nov. 2016.

ples of the ways that companies and cities try to incentivize sustainable transportation choices^{7,8}.

This thesis lays the technical and design groundwork for a blockchain-based system that would manage and monetize new forms of granular urban data, and reward and incentivize sustainable transportation behavior.

1.3 Overview

Chapter 2 covers background and related work in the several diverse areas that come together to support this thesis: bicycling advocacy and data-driven urban planning; nudge behavioral studies and financial incentives; and blockchain networks, especially as used in sensor-based applications. Chapter 3 explains how this thesis synthesizes and builds upon this background work. Chapters 4 and 5 cover the technical contributions of this thesis, including the design, development, and fabrication of a bicycle-mounted sensor device and the blockchain-based software application that manages data and distributes incentives. Chapter 6 documents the project deployment and our evaluation of the technical successes and challenges of this system, and Chapter 7 presents the thesis conclusions and explores further opportunities for this line of research. Three Appendices provide detailed documentation of the technical procedures of the thesis.

⁷Patel, Mitesh, David Asch, & Kevin Volpp. "Paying Employees to Lose Weight - The New York Times." New York Times. N.p., 4 Mar. 2016. Web. 3 Nov. 2016.

⁸Mearian, Lucas. "Insurance company now offers discounts – if you let it track your Fitbit." Computer World 17 Apr. 2015: n. pag. Print.

Chapter 2

Background & Related Work

This thesis draws upon several diverse fields including bicycling advocacy and data-driven urban planning, nudge behavioral studies and financial incentives, and blockchain systems used in sensor-based applications. This chapter intends to give a general sense of the work in each area, as well as point to relevant examples that directly give rise to the content of this thesis.

2.1 Bicycling & Urban Studies

Academics, city planners, and citizens alike have been interested in cycling for several decades. Cycling is recognized as a healthy, convenient, sustainable form of urban transportation that provides many benefits: reduced emissions, increased physical activity, reduced congestion and fuel use, and flexible and affordable mobility. Many believe that cycling has the potential to offset the growing environmental and public health concerns of increasingly dense urban areas¹.

2.1.1 Positive Externalities of Cycling

Research from a variety of fields has examined the positive externalities of bicycling. From an emissions perspective, bicycles are a completely non-polluting form of transportation. In one analysis of a bike-sharing system, where the system's bikes covered 200,000 km per day, researchers estimated that a car covering the same distance would produce 37,000 kg of carbon dioxide (CO₂) emissions¹.

From a public health standpoint, the benefits of regular exercise are well-documented. The Surgeon General's report states that 30-45 minutes of bicycling (or other physical activity) "will reduce...risks of developing coronary heart disease, hypertension, colon cancer, and diabetes," which is especially important in a country where heart disease is the leading cause of death, causing roughly 25% of deaths, and over half of the

¹Shaheen, S. A., Guzman, S., & Zhang, H. (2010). Bikesharing in Europe, the Americas, and Asia: Past, Present, and Future. *Transportation Research Record*, 2143, pp. 159-167.

American population is not regularly active^{2,3}.

Finally, the bicycle provides a flexible and affordable form of mobility that is much more accessible than a private vehicle. One source estimates that bicycling instead of driving could save roughly \$1300 per year, and writes that though only 10% of the world's population can afford a car, 80% can afford a bike; thus, the bicycle provides "economic and independent travel for those who might otherwise have their travel options restricted"⁴.

Despite this research and increasing acknowledgement of the positive externalities of cycling, they remain just that: the external benefits of cycling are rarely factored into the calculus around national health and transportation. Dora notes that: "estimates of the health impacts and costs of transport strategies do not include the health effects of increased walking and cycling and the savings associated with increased walking and cycling for a population", and suggests that the "burden of transport on health is higher than expected, partly because users do not pay the full costs of the transport activities they engage in"⁵. This thesis suggests a way to move towards recognizing and rewarding these positive externalities.

2.1.2 Improving Cycling Mode Share

Despite efforts to make cycling an accessible and attractive transportation option, cycling faces low mode share and is not yet considered a viable form of daily transportation in most parts of the world. There are many factors that contribute to low cycling rates: social and cultural norms, government policies, the lack of dedicated cycling infrastructure, and difficult cycling topography or climate, among others⁶. As such, strategies to increase the number of cyclists in cities are diverse, ranging from government policy to data-driven grassroots advocacy.

Government Policy

Under certain conditions, government policy and government-led infrastructure projects have been successful in improving cycling mode share. In their well-regarded 2008 paper, Pucher and Buehler document how governments in The Netherlands, Germany, and Denmark revived flagging cycling cultures while other similarly wealthy and industrialized countries allowed the private vehicle to dominate⁷. Many of these

²U.S. Department of Health and Human Services. "Physical Activity and Health: A Report of the Surgeon General." (1996): n. pag. Web. 30 Mar. 2017.

³"Heart Disease Facts." Centers for Disease Control and Prevention. Centers for Disease Control and Prevention, 10 Aug. 2015. Web. 30 Mar. 2017.

⁴"Cycling benefits." Department of Transport and Main Roads. Queensland Government, 24 Nov. 2016. Web. 30 Mar. 2017.

⁵Dora, Carlos. "A different route to health: implications of transport policies." *Bmj* 318.7199 (1999): 1686-689. Web. 30 Mar. 2017.

⁶Heinen, Eva, van Wee, Bert, & Maat, Kees (2010). *Commuting by Bicycle: An Overview of the Literature*. *Transport Reviews*, 30(1), 59-96. <http://doi.org/10.1080/01441640903187001>.

⁷Pucher, J., & Buehler, R. (2008). *Making Cycling Irresistible: Lessons from The Netherlands, Denmark and Germany*. *Transport Reviews*, 28(4), 495-528. <http://doi.org/10.1080/01441640701806612>.

wealthy, industrialized countries enjoyed high rates of cycling up to and during the 1950's, but saw diminished cycling rates in the 1960's and 70's due to the increased prevalence of the car.

Through policies focusing on limiting private vehicle use and encouraging sustainable transportation use, The Netherlands, Germany, and Denmark succeeded in boosting cycling prevalence and making it a popular, mainstream form of transportation. In the Netherlands, for example, 27% of all trips are made by bicycle. The success of these types of policies was enhanced by those countries' cultural attitudes towards government regulation and a history of being comfortable with cycling.

Unfortunately, along with most other countries in the world, the U.S. has not been able to enact such strong government policies. In the U.S., only 1% of trips are made by bicycle overall. Though the number of cyclists is growing, the bicycle is far from being considered a viable, mainstream means of transportation. Even in Boulder, CO, the city with the highest percentage of bicycle commuters in the country, only 11% of trips are made by bike⁸. Pucher and Buehler document some of the bicycle-focused infrastructure improvements that have been made in the U.S. (such as bike lanes and paths in Chicago and New York), but also discuss the lack of political will towards restricting car use. U.S. policy-makers are extremely resistant to anti-car policies, which is reflected in the abundance of free parking lots, our heavily subsidized gas price, and the aggressive automobile lobbying industry⁹.

Though U.S. employers can offer cycling and public transit subsidies to their employees under the federal government's Commuter Tax Benefits program, there has been serious criticism directed towards the program, which also allows employers to offer their employees subsidized parking. Dutzik and Inglis write:

Ultimately, the effect of the tax benefit for commuter parking is to subsidize traffic congestion by putting roughly 820,000 more cars on America's most congested roads in its most congested cities at the most congested times of day. It delivers the greatest benefits to those who need them least, typically upper-income Americans, and costs \$7.3 billion in reduced tax revenue that must be made up through cuts in government programs, a higher deficit, or increases in taxes on other Americans¹⁰.

They also note that the tax benefits for public transit users and cyclists "only weakly counteract the negative impact of the parking tax benefit", echoing other analysis that indicates that subsidies encouraging sustainable behavior are less effective in changing behavior when they're offered—as they often are—alongside steep subsidies for parking and driving¹¹.

⁸Where We Ride: Analysis of bicycle commuting in American cities. Rep. League Of American Bicyclists, 2014. Web. 23 Mar. 2017.

⁹Delucchi, Mark. "Do Motor-vehicle Users in the US Pay Their Way?" Do Motor-vehicle Users in the US Pay Their Way? N.p., Dec. 2007. Web. 3 Nov. 2016.

¹⁰Dutzik, Tony, and Jeff Inglis. "Subsidizing Congestion: The Multibillion-Dollar Tax Subsidy That's Making Your Commute Worse." (2014): n. pag. TRID. Web. 28 Mar. 2017.

¹¹"Expanded Transit Tax Incentive Doesn't Relieve Impact of Subsidized Parking." TransitCenter. N.p., 11 Feb. 2016. Web. 3 Nov. 2016.

Grassroots Cycling Advocacy Efforts

On the other end of the spectrum, alongside the democratization of data and software tools and the rise of social media, there have emerged a variety of data-driven grassroots advocacy efforts and online bicycling communities. These phenomena take many forms, but often intend to use light-touch sociotechnical tools to improve the experience of riding, or to analyze and disseminate better information on cycling habits.

Tactical urbanism is a term that has been used to describe "short-term, community-based projects...[that] have become a powerful and adaptable new tool of urban activists, planners, and policy-makers seeking to drive lasting improvements in their cities"¹². In the cycling world, this tactic has often manifested as pop-up bicycling infrastructure, with separated bike lanes, delineated by toilet plungers or flower pots (as in Figure 2-1), appearing overnight^{13,14}. These efforts are intended to demonstrate the potential effects of infrastructure changes, and are sometimes adapted into formal city policy.



Figure 2-1: Tactical Urbanism Example: Flower Pot Bikes Lanes (www.fastcompany.com)

Alongside these actions in the physical world, there is a growing body of data- and analytics-based work that has appeared online. Many city bike sharing systems publish anonymized ridership data. Hobbyists, bicycling advocates, and data scientists alike have used this data for a huge variety of visualizations meant to show system usage, identify popular routes and stations, and point out gaps in the system. For example, Oliver O'Brien at University College London maintains a dynamic map of global bike systems showing bicycle availability at each individual station, an example of which can be seen in Figure 2-2¹⁵.

¹²Lydon, Mike, Anthony Garcia, and Andrés Duany. *Tactical urbanism: short-term action for long-term change*. Washington (D.C.): Island Press, 2015. Print.

¹³Metcalf, John. "A New Twist in Guerrilla Bike Lanes: Toilet Plungers." *CityLab*. The Atlantic, 28 Feb. 2017. Web. 27 Mar. 2017.

¹⁴Peters, Adele. "A Guerrilla Bike Lane Made With Flower Pots Forces A City's Hand." *Fast Company*. N.p., 17 Sept. 2015. Web. 27 Mar. 2017.

¹⁵O'Brien, Oliver. "Bike Share Map." *Bike Share Map*. N.p., n.d. Web. 27 Mar. 2017.

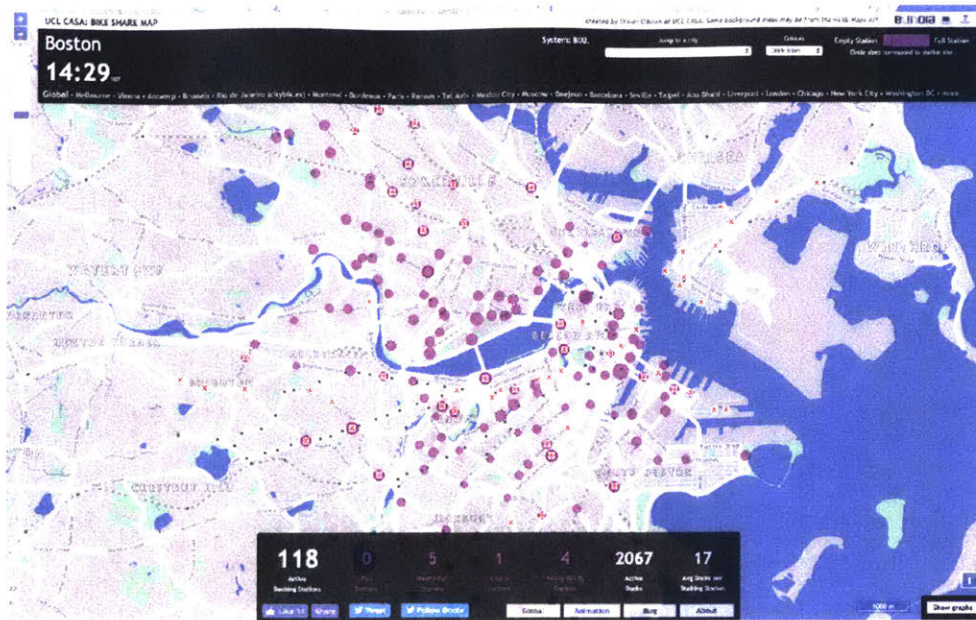


Figure 2-2: Bike Data Visualization Example: Map Showing Availability of Boston-Area Bike Share Bicycles (Bike Share Map, <http://bikes.oobrien.com/>)

Finally, the availability of increasingly powerful and ubiquitous mobile devices has led to the rise of social tracking applications, such as Strava¹⁶. Strava is used by recreational cyclists, professionals, and commuters. The application allows users to track and visualize their rides, discover routes, and compete with other users. The application has taken on a life of its own in certain settings, such as the case of a man who spelled out a marriage proposal using Strava traces from his bike ride¹⁷. Strava, and similar applications, allow cyclists to publicize and socialize their cycling activity. The application and associated forums form the basis of a bottom-up, distributed virtual community that allows cyclists to organize, connect, and engage in offline advocacy activities.

Self-organized, grassroots cycling organizations have appeared in recent years, often enabled by modern mobile technologies and the democratization of data analysis tools. Though these communities and efforts take many forms, they are generally united by their bottom-up, distributed quality, and their use of light-touch sociotechnical systems to improve the experience of cycling and the bolster cycling communities.

¹⁶"Run and Cycling Tracking on the Social Network for Athletes." Strava. N.p., n.d. Web. 27 Mar. 2017.

¹⁷Xie, Jenny. "A San Francisco Man Just Proposed With His Bike Route." CityLab. The Atlantic, 14 Jan. 2014. Web. 27 Mar. 2017.

2.2 Behavioral Psychology

Observable transportation behavior is often a product of the myriad daily microdecisions that individuals encounter. However, traditional transportation policy and engineering have not considered the psychological aspects of transportation, instead choosing to treat individuals as perfectly rational actors who execute a precise cost-benefit analysis each time they step outside their homes. While this approach is coming under increased scrutiny, and there is a new focus on examining the habits and cultural preferences of individuals, considering the psychology of an individual's decision-making is still not a common element of transportation initiatives¹⁸.

In order to design effective systems that take into account the habits and preferences of individuals, this thesis draws from "nudge" behavior studies and research on using financial subsidies to motivate behavior; both these phenomena have been explored thoroughly outside of transportation.

2.2.1 Nudge Behavioral Studies

This thesis is particularly concerned with *nudging*: situations where small changes are introduced in an individual's choice set that are designed to push individuals towards certain behavior without excessively limiting their options¹⁹. Hansen defines nudging:

Any attempt at influencing people's judgment, choice or behavior in a predictable way...that is motivated because of cognitive boundaries, biases, routines, and habits...and which works by making use of those boundaries, biases, routines, and habits as integral parts of such attempts²⁰.

Sunstein and Thaler offer an illustrative example of using a nudge to encourage healthier eating: "Putting fruit at eye level counts as a nudge. Banning junk food does not."¹⁹. Nudge design often takes advantage of certain human cognitive biases and irrational tendencies, including:

- Anchoring Bias: people tend to be heavily biased by the initial data points presented to them
- Availability Bias: people tend to overestimate the likelihood of things that are easy to remember
- Representativeness Bias: people tend to see patterns when there are none
- Overestimation: almost all individuals think they are better than average

¹⁸Abidi, Abdelhamid. "Transport and environmental regulation - common attitudes and social change." Urban Environment Alliance for Global Sustainability Bookseries (2011): 3-13. Web. 27 Mar. 2017.

¹⁹Leonard, Thomas C. "Richard H. Thaler, Cass R. Sunstein, Nudge: Improving Decisions about Health, Wealth, and Happiness." Constitutional Political Economy 19.4 (2008): 356-60. Web.

²⁰Hansen, Pelle Guldberg. "The Definition of Nudge and Libertarian Paternalism: Does the Hand Fit the Glove?" European Journal of Risk Regulation 7.01 (2016): 155-74. Web. 28 Mar. 2017.

- Loss Aversion: people tend to prefer avoiding losses over acquiring equivalent gains
- Status Quo Bias: behavioral inertia must be overcome when changing a habit
- Framing Effect: people react to a particular choice in different ways depending on how it is presented
- Priming Effect: people can be influenced to make a certain choice based on what they see or experience directly before making their choice
- Hyperbolic Discounting: people prefer rewards that happen sooner rather than later, even if the rewards have the same actual value

When properly designed to take advantage of these human tendencies and irrationalities, nudges can have a powerful effect on human behavior, leading people to change deeply ingrained habits.

2.2.2 Nudging with Financial Incentives

This thesis is particularly concerned with using financial incentives to nudge people towards certain transit behavior. In general, when designed carefully, financial incentives have proven quite effective in changing behavior and helping people overcome detrimental habits. For example, in one study conducted at General Electric, employees who were part of a treatment group were given \$250 to quit smoking for 6 months, and \$400 to quit smoking for 12 months. The treatment group had 3 times the success rate of the control group, an effect that persisted even after the study had ended²¹.

In a different study on weight loss, participants were entered into a lottery to win money if they lost weight. If they lost weight and won the lottery, they earned money. If they didn't lose weight and won the lottery, they were notified of the money they would have earned. In the four-month study, participants in the treatment group lost 3 times more weight than those in the control group²². This study takes advantage of humans' natural loss aversion to further encourage weight loss. In addition to loss aversion, well-designed financial nudge studies often employ the following two strategies: taking advantage of framing effects by paying participants explicitly (e.g. giving them a separate check instead of adding to their payroll) and considering hyperbolic discounting by rewarding positive actions quickly (e.g. people are more responsive to instant gratification)²³.

²¹Volpp, K. G., Troxel, A. B., Pauly, M. V., Glick, H. A., Puig, A., Asch, D. A., ... Audrain-McGovern, J. (2009). A Randomized, Controlled Trial of Financial Incentives for Smoking Cessation. *New England Journal of Medicine*, 360(7), 699-709. <http://doi.org/10.1056/NEJMsa0806819>.

²²Volpp, Kevin G., Leslie K. John, Andrea B. Troxel, Laurie Norton, Jennifer Fassbender, & George Loewenstein. "Financial Incentive-Based Approaches for Weight Loss." *Jama* 300.22 (2008): 2631. Web. 3 Nov. 2016.

²³Patel, Mitesh, David Asch, & Kevin Volpp. "Paying Employees to Lose Weight - The New York Times." *New York Times*. N.p., 4 Mar. 2016. Web. 3 Nov. 2016.

2.2.3 Financial Incentives & Transportation

Using financial incentives to impact transportation behavior is not a novel idea. As discussed in Section 2.1.2, parts of the U.S. tax code provide subsidies for people who commute by bicycle or public transportation. However, these benefits are offered alongside a massive subsidy for employee parking. Dutzik and Inglis estimate that for a commuter, the value of the parking subsidy can be up to \$250 per month, while the transit benefit and cycling benefit only carry values of \$130 and \$20 per month, respectively¹⁰. Given this lopsided benefit structure, it is understandable that this program would be unlikely to have a huge impact on the use of sustainable transportation, and much less be able to lure a driver away from a car habit.

In other countries, there have been several programs that—similar to this thesis—experimented with paying cyclists directly. In trial programs in France in 2014, and in Milan in 2016, governments paid cyclists about 25 cents per kilometer cycled to and from work. Though the results were not overwhelmingly successful, possibly due to the small payout amount, France did decide to adopt the incentives program as a permanent initiative. In their trial period, they found that 19% of participants switched to cycling from driving, and a third of new cyclists reported increased use of cycling for other purposes²⁴.

2.3 Blockchain

A blockchain is a distributed database technology that enables network nodes to make peer-to-peer transactions without requiring the authorization of a central node. Though a few precursors to this technology had been discussed in the late 1990's and early 2000's, blockchain technology—alongside the Bitcoin cryptocurrency—was officially introduced in Satoshi Nakamoto's seminal 2008 paper, and implemented in code the following year²⁵.

A blockchain is essentially a large public ledger, or list, that keeps track of all the transactions in a network, with the idea that making everything public makes it hard to create false or fraudulent transactions. Groups of transactions are stored in a larger data structure called a "block", and blocks are linked to each other in a long list, or "chain". In order to ensure the accuracy of information in the ledger, the blockchain protocol makes it difficult to add new blocks to the chain, so that it is expensive to add incorrect information to the system's history. The difficulty of adding new transactions is often introduced by requiring nodes to complete computationally challenging or memory-intensive operations, which are called "proof-of-work". While "proof-of-work" is the most common mechanism for managing which transactions are added to a blockchain, other blockchains implement different mechanisms, such as "proof-of-stake", which allows nodes to add new blocks based on their value stake in the network.

²⁴Ng, Angie. "It's official: French will get paid for cycling to work." We Love Cycling. N.p., 16 Oct. 2015. Web. 28 Mar. 2017.

²⁵Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." Bitcoin. N.p., Nov. 2008. Web. 3 Nov. 2016.

Different nodes in the network compete for the privilege of adding a new block to the chain through a process called "mining", which involves demonstrating "proof-of-work" or "proof-of-stake". When a node mines a new block and is the first to propagate their block to the rest of the network, they are rewarded with a payment, and the new block is added to the chain. The mining reward incentivizes nodes to mine blocks and maintain the network. The network reaches consensus around which new chain to accept as the true version of system history by accepting the chain that can demonstrate the most proof-of-work.

2.3.1 Advantages & Typical Applications

Blockchain technology has many potential applications and has, in the past decade, generated considerable excitement for its potential to change operating paradigms in a variety of fields. In general, blockchain is "suited for applications that automate direct interaction between peers or facilitate coordinated group action across a network"²⁶. Blockchains are appropriate for these types of applications because they are:

- **Distributed and trustless:** data can be shared across "boundaries of trust", without requiring a central administrator. As Greenspan states: "blockchain transactions contain their own proof of validity and their own proof of authorization, instead of requiring some centralized application logic to enforce those constraints. Transactions can therefore be verified and processed independently by multiple 'nodes', with the blockchain acting as a consensus mechanism to ensure those nodes stay in sync"²⁷.
- **Robust and secure:** because data storage and transaction processing happen on each node in the network, the system is less vulnerable to malicious attacks or hardware failure points. For example, in a blockchain network with ten nodes, even if several of the nodes failed or became disconnected, the information in the blockchain database would still be available on the remaining nodes.
- **Automated:** transactions handled on the blockchain are executed and approved based on a pre-determined protocol, and do not need to be approved or reviewed by a human administrator. As a result, transactions on the blockchain are fast and cheap compared to the way value is exchanged in many of our current institutions. For example, financial transactions on the blockchain are only limited by the amount of time it takes to mine a block, which is usually on the order of 10 minutes.

It is important to note that blockchains are not always the best solution for technical systems. One disadvantage of blockchains is that they are not as efficient as a traditional centralized database, because each database transaction must be processed

²⁶"Ethereum Homestead Documentation." Ethereum Homestead 0.1 documentation. Ethereum, n.d. Web. 29 Mar. 2017.

²⁷Greenspan, Gideon. "Blockchains vs centralized databases." Web log post. MultiChain. Coin Sciences Ltd., 17 Mar. 2016. Web. 29 Mar. 2017.

on each node in the network, and in "proof-of-work" systems, energy and computing cycles must be spent on reaching network consensus²⁷. Nonetheless, in certain fields such as finance and supply chain management, the benefit of a robust, decentralized database technology outweighs these downsides, and there have been a number of attempts to integrate blockchain technology into their ongoing operations.

Bitcoin

Blockchain technology is most well known as the technology that underpins Bitcoin, the cryptocurrency that was also introduced in Nakamoto's 2008 paper. The Bitcoin system relies on a blockchain network and "proof-of-work" consensus mechanism, as described in Section 2.3, to create, manage, and exchange the cryptocurrency.

Bitcoin is a pseudonymous digital currency that is not tied to any government or existing banking system, which makes it easy to use internationally, and occasionally for illicit purposes. Bitcoin can be procured through mining (e.g. solving hard computational problems to help maintain the blockchain database, and getting rewarded in return) or purchased on an exchange (an online platform that lets people buy and sell different currencies). For example, on the popular currency exchange Coinbase, people can exchange U.S. dollars for Bitcoin²⁸. Once in possession of some Bitcoin—which will often be stored in a virtual "wallet" on a mobile device or piece of computer hardware—people can use Bitcoin to pay for goods or services, online or in person. As of 2015, over 100,000 merchants accepted Bitcoin in exchange for goods, which demonstrates the system's increasing traction²⁹.

2.3.2 Emergent Applications & IoT Work

Though blockchain technology is most familiar as the technology that supports the Bitcoin cryptocurrency, it is increasingly being used in other applications, from academic credentialing systems to music rights management^{30,31}. In these cases, the use of a blockchain helps makes systems transparent, accurate, and difficult to manipulate with malicious intent.

There has been significant interest—though few actual implementations—in using blockchain technology for Internet of Things (IoT) and distributed sensor network applications. Blockchain databases offer a solution to some of the security and data-management challenges of IoT applications, because they facilitate secure peer-to-peer communication between devices, instead of requiring those devices to talk to a central server³². One of the earliest examples of using a blockchain for the IoT is Samsung and IBM's proof-of-concept ADEPT framework, which demonstrates how appliances

²⁸"Buy and Sell Bitcoin and Ethereum." Coinbase. N.p., n.d. Web. 29 Mar. 2017.

²⁹Cuthbertson, Anthony. "Bitcoin now accepted by 100,000 merchants worldwide." International Business Times. N.p., 4 Feb. 2015. Web. 29 Mar. 2017.

³⁰"Blockchain Certificates." Blockcerts. Web. 3 Nov. 2016.

³¹Heap, Imogen, and Don Tapscott. "Blockchain Could Be Music's Next Disruptor." Fortune. 22 Sept. 2016. Web. 3 Nov. 2016.

³²"Device Democracy: Saving the Future of the Internet of Things." IBM. N.p., Aug. 2014. Web. 3 Nov. 2016.

in the home (such as a "smart washer") can autonomously request maintenance and supplies through the blockchain³³. Another example is Sensor21, a new system that allows users to install standardized environmental sensors in or near their households, and get paid for sensor data in Bitcoin³⁴. Though there is substantial interest in using the blockchain to support IoT systems, there have been few actual implementations; many questions remain about what information will be stored on the blockchain, how nodes will communicate with each other, what type of "mining" protocol will be used to secure the network, and how to reconcile the power- and space-intensive process of maintaining a blockchain with the constraints of the small, low-power devices that comprise the IoT.

2.3.3 Ethereum

Ethereum is an open-source, blockchain-based distributed computing platform that is run by the non-profit Ethereum Foundation³⁵. The Ethereum system manages, stores, and creates a digital cryptocurrency called "Ether", similar to the way that Bitcoin is managed by the Bitcoin Blockchain. Ether is less well known than Bitcoin, but has grown in popularity and visibility since Ethereum was established in 2015.

In addition to managing and creating a cryptocurrency, Ethereum provides the ability to program, store, and execute "smart contracts" on its blockchain. Smart contracts are:

...pieces of software, not contracts in the legal sense, that extend blockchains' utility from simply keeping a record of financial transaction entries to automatically implementing terms of multiparty agreements. Smart contracts are executed by a computer network that uses consensus protocols to agree upon the sequence of actions resulting from the contract's code. The result is a method by which parties can agree upon terms and trust that they will be executed automatically, with reduced risk of error or manipulation³⁶.

With the inclusion of smart contracts, Ethereum greatly expands the capabilities and possible range of applications for blockchain technology. Ethereum also provides a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), which can execute these smart contract scripts on a distributed network of nodes.

³³"Empowering the Edge: Use Case Abstract for the ADEPT Proof-of-concept." www.ibm.com. IBM Global Business Services, Jan. 2015. Web. 3 Nov. 2016.

³⁴"Sensor21: Earn bitcoin by collecting environmental data." 21.co. Web. 8 Dec. 2016.

³⁵"Ethereum Project." Ethereum Project. N.p., n.d. Web. 29 Mar. 2017.

³⁶Ream, John, Yang Chu, and David Schatsky. "Upgrading blockchain: Smart contract use cases in industry." University Press. Deloitte, 8 June 2016. Web. 29 Mar. 2017.

Chapter 3

Approach

This thesis presents a financial incentives system for cyclists that provides a framework for organizations to sponsor and encourage sustainable transportation behavior. The technical implementation of the system consists of a network of bicycle-mounted GPS sensors that share a blockchain database. The subsequent section discusses the broad philosophy behind the project's formulation and technology selection.

3.1 Bicycles & Behavioral Psychology

In Section 2.1, this thesis discusses the numerous—yet largely unrewarded—positive externalities of bicycling, as well as several of the top-down and bottom-up attempts that have been made to improve the experience and prevalence of cycling, particularly in cities. Section 2.2 covers basics of behavioral psychology, including the idea of *nudging*, wherein small changes in an individual's choice architecture can take advantage of their innate cognitive biases and push them towards certain choices. This thesis is particularly concerned with how effective financial incentive structures can be designed to change transportation behavior.

The motivation behind this thesis is to unite the ideas of Sections 2.1 and 2.2 by designing and building a lightweight sociotechnical system that recognizes and rewards the positive externalities of cycling. The solution presented in this thesis attempts to fuse bottom-up and top-down approaches, by empowering the individual cyclist to monetize their cycling behavior, while also giving companies, governments, and organizations a secure method of incentivizing sustainable transportation behavior.

This thesis presents a long-term, self-sustaining solution that matches the incentives of participants on both sides. For instance, city governments are incentivized to encourage cycling because it will reduce urban congestion and traffic-related emissions. Local business owners might be incentivized to encourage cycling because it improves business growth in cities¹. Health insurance companies would probably like to reward cycling, much the same way they are starting to reward physical activity

¹Maher, Amanda. "In Promoting Biking, Cities Report Economic Benefits." ICIC. N.p., 21 May 2015. Web. 10 Apr. 2017.

tracked by a Fitbit, because it makes their insurees healthier². In order to develop a successful, self-sustaining solution, this system takes advantage of several of the human cognitive biases discussed in Section 2.2, such as loss aversion and framing effects. By incorporating these design principles into the rewards structure of the incentives system, this thesis aims to present a structure where the motivations of both the cyclists and the sponsoring organizations are met.

3.2 Why Blockchain?

A blockchain architecture was selected for this financial incentives system because we would like the system to be decentralized, robust, accessible, and reliable, which, as discussed in Section 2.3.1, are attributes that are particularly well-supported by the use of a blockchain. The Ethereum documentation explains that blockchain technology is "suited for applications that automate direct interaction between peers or facilitate coordinated group action across a network", which is exactly what our system aims to do³.

The *decentralized* nature of blockchain is appropriate for the system presented in this thesis because there is no central arbiter or manager who logs cyclists' data or coordinates payments from sponsoring organizations. This architecture makes the incentives framework more accessible and transparent, allows network entities to more fully control their interaction with the system, and allows the network to scale quickly and organically. The system this thesis envisions *is* decentralized by nature, which parallels the distributed nature of a blockchain database.

A blockchain database is considered *robust* and *secure* because it distributes the vulnerabilities of the network. Instead of exposing a single point of attack, like a centralized server does, blockchain systems store data on every node in the network, so that if one node is compromised, the operation and contents of the overall network remain intact. This risk distribution would be a desirable feature in many cases, but is particularly appropriate for a system like the one presented in this thesis, where personal data and private financial transactions are being recorded on the network.

Finally, blockchain databases support *automated* transactions that are much *faster* and *cheaper* than traditional financial organizations are able to provide. Sending money through a traditional financial database architecture usually takes several days and may incur a transaction cost; with a blockchain network, value can be transferred within minutes at very low cost. The design of this thesis' incentives system requires that cyclists are able to receive frequent financial rewards close to the time that they perform the actions that earn those rewards, so that they associate their actions with the financial rewards and stay engaged with the system. For this reason, the speed and low cost of transactions on the blockchain make it appropriate for this thesis.

²Mearian, Lucas. "Insurance company now offers discounts – if you let it track your Fitbit." ComputerWorld. IDG, 17 Apr. 2015. Web. 10 Apr. 2017.

³"Ethereum Homestead Documentation." Ethereum Homestead 0.1 documentation. Ethereum, n.d. Web. 29 Mar. 2017.

The qualities of blockchain databases that make them useful for the implementation of this thesis also make them appropriate for a host of other emergent applications. The Ethereum documentation argues that "beyond financial applications, any environments where trust, security, and permanence are important—for instance, asset-registries, voting, governance, and the internet of things—could be massively impacted by the Ethereum platform"³. In a similar vein, some hypothesize that the blockchain will become the "shared 'back end' to a decentralized, secure internet—Web 3.0...where core services like DNS and digital identity are decentralized, and where individuals can engage in economic interactions with each other"⁴.

There has been great interest in the potential applications of the blockchain. One need look no further than the "State of the Dapps" website, a curated listing of Ethereum-based decentralized applications, pictured in Figure 3-1, to see the breadth of blockchain-related innovation⁵. Academics, industry groups, and hobbyists alike are experimenting with blockchain-based applications, hoping to explore and understand the potential of this emerging technology⁶. This thesis intends to position itself alongside these exploratory efforts. While the system envisioned in the thesis could be accomplished with a more traditional database technology, the blockchain presents several compelling design advantages, and allows us to contribute to the growing literature around blockchain applications.

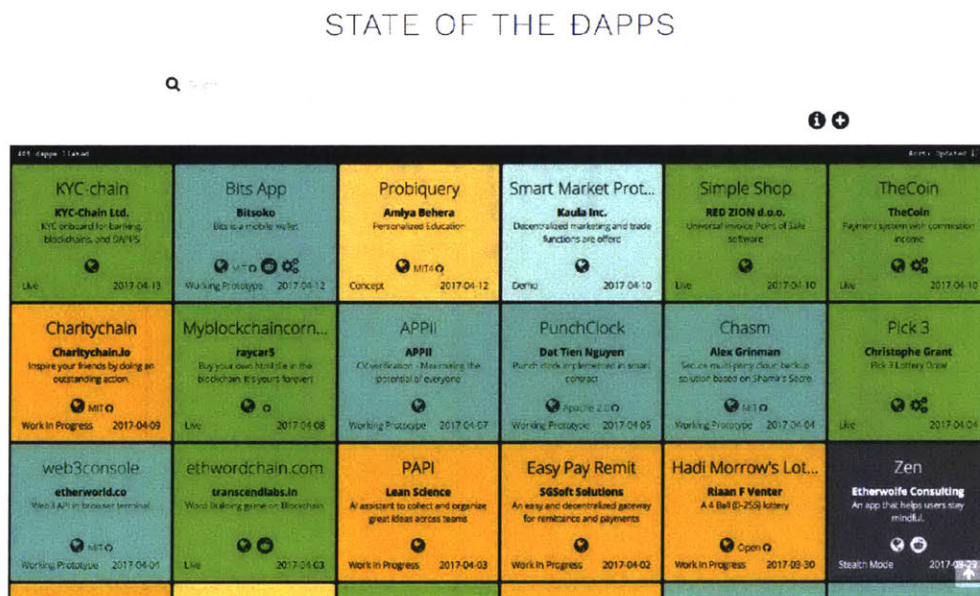


Figure 3-1: Example of Ethereum-based decentralized applications (State of the Dapps, <http://dapps.ethercasts.com>)

⁴"Web 3: A platform for decentralized apps." Ethereum Homestead Documentation. Ethereum Foundation, n.d. Web. 17 Apr. 2017.

⁵State of the Dapps. Ethercasts, n.d. Web. 17 Apr. 2017.

⁶Tapscott, Don, and Alex Tapscott. Blockchain revolution: how the technology behind bitcoin is changing money, business and the world. London: Portfolio Penguin, 2016. Print.

Chapter 4

Hardware Procedures

The main hardware contribution of this thesis is the design, development, and deployment of an Internet-connected GPS sensor that can connect to a blockchain network and be affixed to—and powered by—a bicycle. This device was largely assembled from off-the-shelf components, including a Raspberry Pi single-board computer and the Adafruit FONA GPS/ GSM module, with the idea that this strategy would allow us to focus on the high-level design constraints of the device. This section describes the design of the sensor, the component selection process, and the preparation, assembly, and deployment of the final hardware system. Descriptions of the software running on the hardware system can be found in Section 4.2.1, Chapter 5, and Appendices B and C.

4.1 Sensor Design

Central to this thesis’ hardware design is a single-board computer that runs an Ethereum node and interfaces with a GPS/ GSM module to collect GPS data and connect to the rest of the blockchain network. In this implementation, we use a Raspberry Pi as the single-board computer; for GPS and GSM capabilities, we use the Adafruit FONA board, a breakout board for the SIM808 GPS/ GSM module that provides some additional features. These devices are supported by several power supply and distribution modules, and mounted on a bicycle. The basic architecture of the system is pictured in Figure 4-1.

4.1.1 System Components

This section describes, in further detail, each of the system components in Figure 4-1 and their relationships to and interactions with each other. This section also discusses why each component was selected, and some of the tradeoffs that were considered in that selection.

- **Raspberry Pi:** the Raspberry Pi single-board computer is the brains of the sensor device. It was chosen because it strikes a balance between its portability and its capability to run complex software. We had originally planned to use an

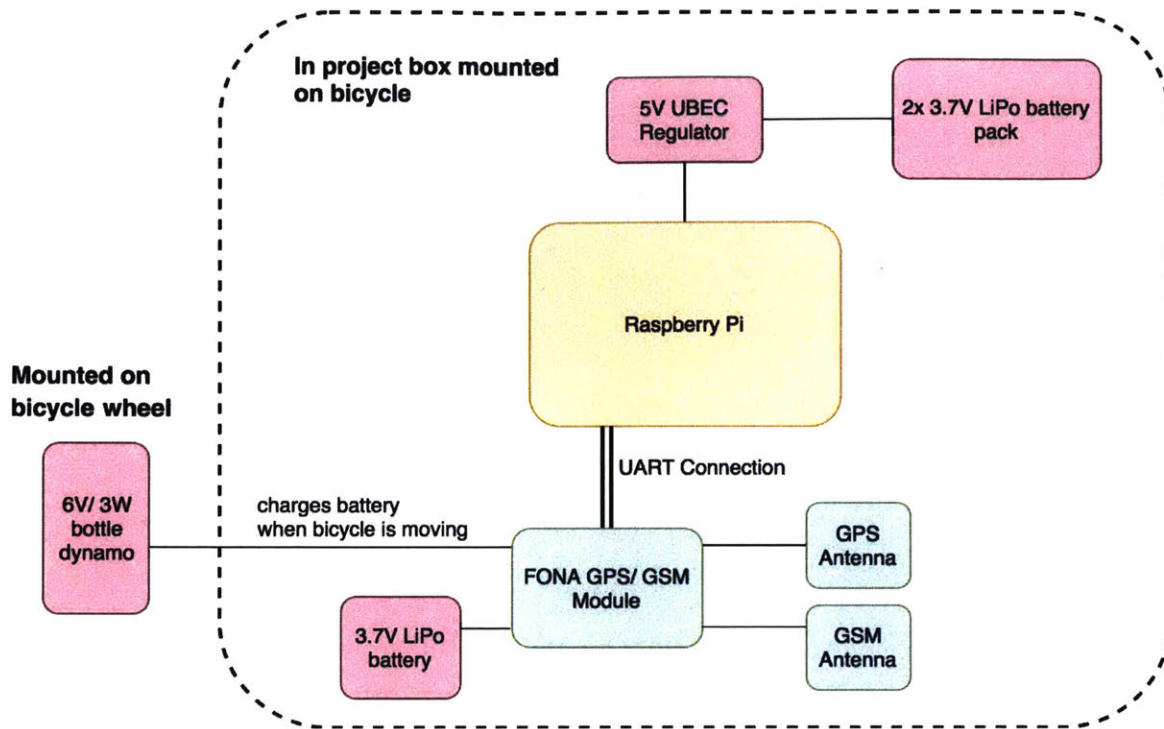


Figure 4-1: Hardware Architecture Diagram

Arduino microcontroller, but changed the design to use a Raspberry Pi because the Arduino platform was not capable of running an Ethereum client. We used the Raspberry Pi 2 Model B V1.1 for this thesis project. This model features four USB ports, a quad-core ARMv7 processor, 1G of RAM, and runs at 5V with a recommend current supply of 2A¹. The Raspberry Pi's chief responsibilities are: first, to run an Ethereum node that synchronizes with and sends cyclist GPS data to the rest of the blockchain network; and second, to request and store GPS from the FONA GPS/ GSM module. To manage this functionality, the Raspberry Pi, which runs an embedded version of Linux called Raspian, runs several scheduled bash scripts. These scripts are discussed in detail in Section 5.4 and Appendix C.

- **Adafruit FONA GPS/ GSM Module:** the Adafruit FONA, picture in Figure 4-2a, is a breakout board for the SIM808 GPS/ GSM module. The board breaks out the pins of the SIM808, and also provides a LiPo battery charging circuit, LED status indicators, and uFL antenna connections, among other features. The module must be used with a GSM antenna, a passive GPS antenna, a 2G SIM card, and a LiPo battery². We used a Ting Mobile data plan with the 2G SIM card that cost \$3 for 100Mb of data, plus \$6 per month for each

¹"Raspberry Pi 2 Model B." Raspberry Pi. N.p., n.d. Web. 11 May 2017.

²"Adafruit FONA 808 Cellular GPS Breakout." Overview | Adafruit FONA 808 Cellular GPS Breakout | Adafruit Learning System. Adafruit, n.d. Web. 04 May 2017.

device³. The Raspberry Pi communicates with the FONA module using the AT Command set, which is sent over a UART connection on the Raspberry Pi's GPIO pins⁴. The FONA is powered by a 500mAh, 3.7V LiPo battery, which is charged, through the MicroUSB interface to the on-board charging circuit, by the output of the bottle dynamo affixed to the bicycle wheel.



Figure 4-2: System Hardware Components

- **HobbyWing UBEC regulator:** the HobbyWing UBEC, pictured in Figure 4-2b, is used to provide a stable, constant, and portable voltage source for the Raspberry Pi. The UBEC is a switching regulator that efficiently converts its input voltage into a stable 5V output voltage, at up to 3A of continuous current. The input voltage can range between 5.5 – 26V. In our system, the input voltage to the UBEC is provided by two 3400mAh 3.7V rechargeable LiPo batteries.
- **AXA Trio Bottle Dyanamo:** the AXA Trio bottle dynamo, pictured in Figure 4-2c, is a small generator that affixes to the frame of a bicycle and generates electrical power from the mechanical motion of the bicycle. The dynamo uses a rectifier to supply DC power instead of AC power; this model of dynamo is able to supply 6V and 3W of power. While it cannot be directly used to power the hardware components of this system, it supplies input voltage to the FONA's recharging circuit, thereby indirectly powering the circuit and obviating the need to manually recharge the FONA's battery. The bottle dynamo also provides an input to the Raspberry Pi that allows the Raspberry Pi to detect when the bicycle is in motion, so that it knows when to actually collect GPS data. For example, there is no need for the system to collect data overnight or when the bike is not in motion. This process is described in further detail in Section 5.4. As pictured in Figure 4-4, the dynamo is mounted on the bicycle

³"A smarter way to do mobile." Ting. N.p., n.d. Web. 15 May 2017.

⁴AT Commands Reference Guide. N.p.: Telit Wireless Systems, 2006. Pdf.

using a mounting bracket that affixes to one of the rear seat stays, and braces the dynamo's roller against the bicycle's moving wheel. Possible improvements to this power supply solution are discussed in Section 7.2.1.

- **Casing and mounting hardware:** the hardware components discussed in this section, with the exception of the bottle dynamo, are arranged and secured within a plastic electrical box, which is secured to the frame of the bicycle using zip ties. This arrangement is pictured in Figures 4-3 and 4-4.

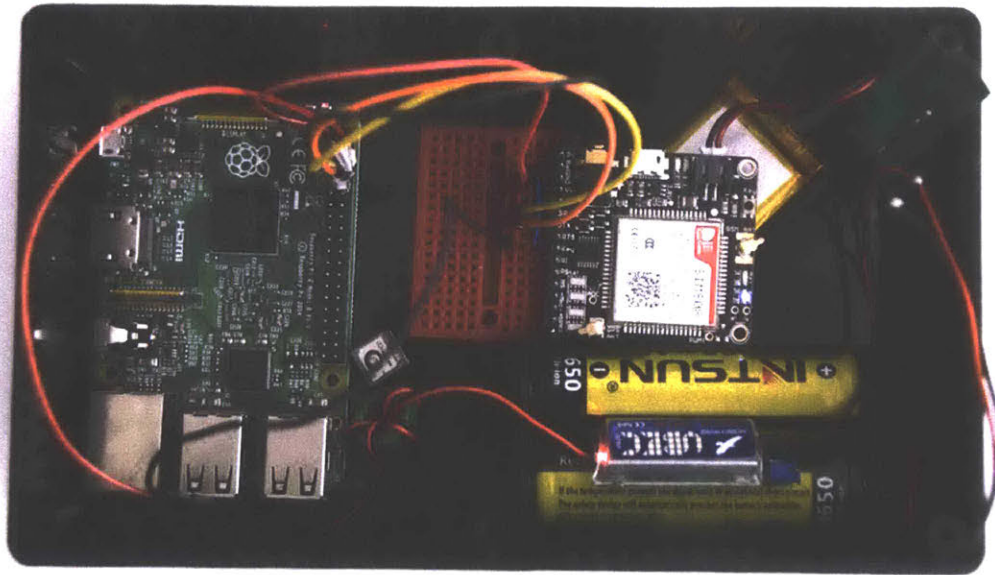


Figure 4-3: Sensor Device in Electronics Box

4.1.2 Device Power Requirements

One of the key challenges in making this hardware system portable was developing a suitable power supply system. In lieu of powering the sensor device directly from a bicycle generator, as had been originally proposed by this thesis, power is supplied to the device through two different LiPo battery packs. The FONA module runs off a single 500mAh 3.7V LiPo battery, while the Raspberry Pi runs off two 3400mAh 3.7V LiPo batteries that are regulated by a 5V switching regulator. The FONA's single LiPo battery is recharged by power from the bicycle generator through the FONA's on-board charging circuit. Due to development time constraints, in the current version of the hardware system, the Raspberry Pi's batteries must be manually recharged; however, given an appropriate recharging circuit, they could also use the bicycle generator to recharge.

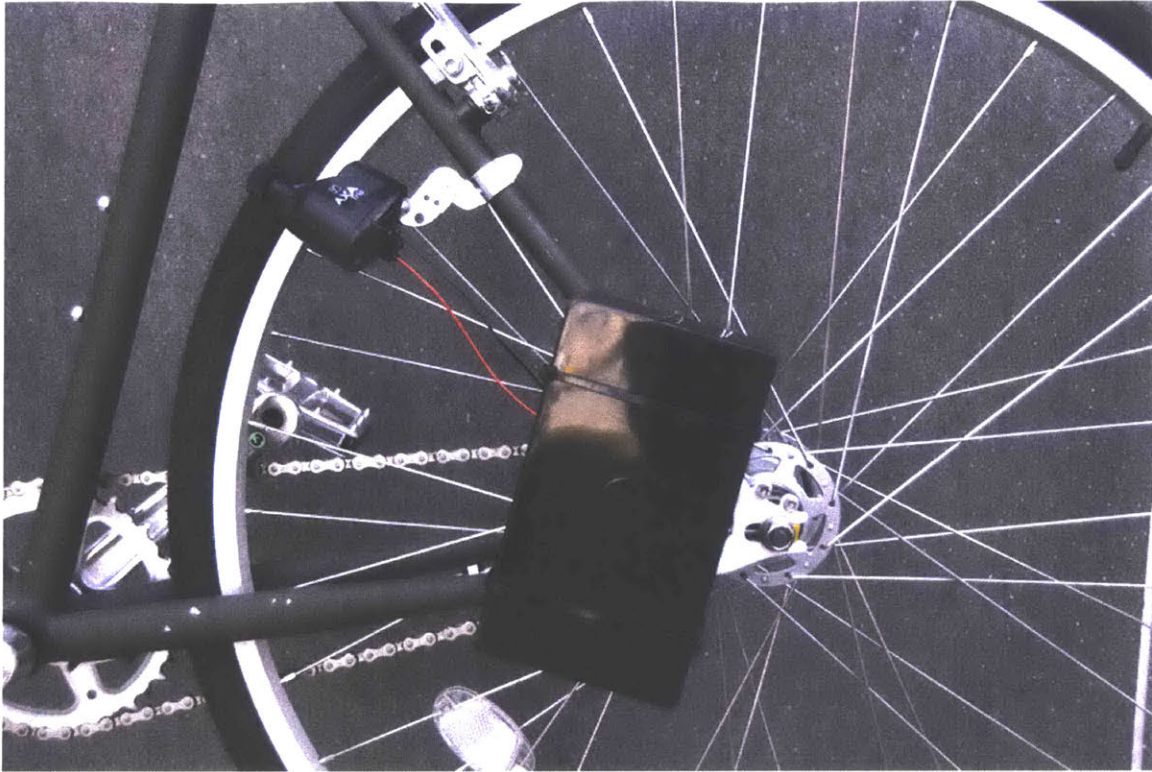


Figure 4-4: Full Hardware System Mounted on Bicycle

According to device documentation, the FONA draws 20-25mA of current on standby, and up to 200mA+ of current while sending or receiving data². Assuming the higher current draw, because the sensor will be using the cellular network data connection so often, we estimate that the FONA can run for about 2.5 hours off the fully charged 500mAh battery. Though this cycle duration would not be sufficient if the battery were the only power source, the FONA battery is constantly recharging as the cyclist rides, which serves to extend how long the FONA can be used.

The Raspberry Pi has somewhat higher power demands. Its documentation indicates that the bare board (without any USB peripherals) draws about 330mA⁵. Given the two 3400mAh batteries, we estimate the Raspberry Pi can run for roughly 10 hours off a single charge, which, in most cases, is sufficient for a single day's use.

While our power supply strategy is adequate for our proof-of-concept system, it is far from optimal. We selected electronic devices for their convenience and ease-of-use, not necessarily for their energy efficiency. We made this trade-off so that we could focus on building an entire system instead of optimizing components at a low level. Suggestions for improving the power efficiency of this system are discussed in Section 7.2.1.

⁵"Raspberry Pi FAQs - Frequently Asked Questions." Raspberry Pi. N.p., n.d. Web. 04 May 2017.

4.2 Device Preparation & Assembly

4.2.1 Raspberry Pi Software Installations

The Raspberry Pi runs the Raspian operating system, an embedded version of Debian that is optimized for the Raspberry Pi's hardware. In order to use the Raspberry Pi for the system proposed by this thesis, we had to install several pieces of important software, which is briefly described in this section.

First, in order to use the FONA module with the Raspberry Pi, we had to disable the Raspberry Pi kernel's use of the board's hardware serial connection, so that the Raspberry Pi could instead communicate with the FONA on that channel. To enable this communication, we included the line `"enable_uart=1"` in the Raspberry Pi's `/boot/config.txt` file. We also configured a persistent Point-to-Point Protocol that allows the Raspberry Pi to use the FONA's GPRS service as its default internet connection⁶.

Next, we had to install the *geth* Ethereum client. Due to the power and storage demands of running a blockchain node, running an Ethereum client on an embedded device or a single-board computer is not common practice. Fortunately, *geth*'s Ethereum client implementation supports the ARM chip architecture, and also offers a beta version of the Ethereum light node, which relieves some of the space and computing constraints of running a full node. The *geth* website provides precompiled *geth* binaries for different hardware architectures; to install, we were able to simply copy the binary to the Raspberry Pi, unzip it, and run the program⁷. The choice of the *geth* Ethereum client is discussed further in Section 5.3 and installation details are covered in Appendix A.

Finally, in order to run the Raspberry Pi Python code that communicates with the FONA module, we had to install a number of Python packages, including *geopy*, a geocoding library, and *pytz*, which provides timezone support, using the Python package manager, *pip*. Further details on our usage of the Raspberry Pi are covered in Appendix C.

4.2.2 Sensor Assembly & Deployment

For purposes of our proof-of-concept deployment, sensor assembly and deployment were largely manual processes. To connect the different electronic components of the system, most of which expose header pins, we used jumper cables and a small breadboard. All electronics were mounted in a plastic electronics box using double-sided tape; the box was in turn mounted on the bicycle frame using zip ties. The bottle dynamo was affixed to the rear seat stay of the bicycle using a specialty mounting bracket. The dynamo exposes power and ground electrodes which we connected to the rest of the system using electrical wires. To power the electronics on and off, we simply connected or disconnected the battery pack from the Raspberry Pi.

⁶"FONA Tethering to Raspberry Pi or BeagleBone Black." Setup | FONA Tethering to Raspberry Pi or BeagleBone Black | Adafruit Learning System. Adafruit, n.d. Web. 04 May 2017.

⁷"Downloads." Go Ethereum. N.p., n.d. Web. 04 May 2017.

Chapter 5

Software Procedures

The software written for this thesis utilizes a variety of libraries and languages, and was written to run across several devices. This section describes the design of the financial incentives program and the overall software architecture of the system, particularly the blockchain network and the Ethereum smart contracts that run on top of it. This section also includes in-depth explanations of the techniques and technologies used to build the system, our rationale for choosing those technologies, and details on the development process of the project. Further information on the topics covered in this Chapter can be found in Appendices A, B, and C.

5.1 Financial Incentives Design

The contribution of this thesis is to build a technical system that rewards the positive externalities of bicycling by connecting cyclists with organizations who are incentivized to encourage more cycling. For example, city governments might like to encourage cycling because it reduces urban congestion and traffic-related emissions; local business owners, because cycling improves business growth in cities; insurance companies, because it makes their insurees healthier. In lieu of interfacing our technical system with actual organizations, which was out of the scope of this thesis, we researched and designed a model for how these organizations might reward cyclists.

After considering the positive externalities of cycling, as well as the financial limitations of various organizations, the reward system implemented in this thesis is as follows: cyclists receive \$1 each day that they bike, plus an additional \$0.25 per mile biked. For example, someone who cycled 4 miles in one day would earn \$2: \$1 as a baseline payment, and \$1 for their observed mileage. Because payments in our blockchain network were delivered in the Ether cryptocurrency, our system translates these rewards based on the current valuation of the Ether relative to the dollar.

This scheme was chosen by observing similar rewards programs, and identifying the positive externalities of cycling as well as the social costs of other transportation methods. For example, the insurance company Oscar rewards its insurees \$1 per day (up to \$240 annually) for hitting a daily step count goal¹. A study found that

¹Oscar | Smart, simple health insurance. N.p., n.d. Web. 26 Apr. 2017.

for a company with 1,000 employees, obesity costs \$285,000 per year in health care costs and lost productivity². However, cycling can offset those costs: another study demonstrated that if national cycling participation increased enough to reduce obesity by about 3%, national medical expenditures could be reduced by \$6 billion³.

Cycling has also been shown to improve business outcomes and profits for local business owners. In one study, when San Francisco made its Valencia Street less conducive to automobile travel and better for bicyclists and pedestrians, nearly 40% of merchants reported increased sales and 60% reported more area residents shopping locally due to reduced travel time and convenience. Two-thirds of merchants said the increased levels of bicycling and walking improved business⁴. Similar effects have been observed in New York: after the construction of a protected bike lane on 9th Avenue, local businesses saw a 49% increase in retail sales; for businesses on other streets in the borough, the average increase was only 3%⁵. There is ample evidence to indicate the public health, environmental, and economic arguments for investing in cycling, which we believe would motivate companies, governments, and non-profits alike to invest in the financial rewards system.

While organizations are incentivized to invest in this system because of positive health or economic outcomes, cyclists are incentivized to use the system because of the financial rewards they could earn. Additionally, the incentives strategy was designed to incorporate principles of behavioral psychology in order to make the system more effective in changing behavior. In addition to the basic premise of providing financial incentives in exchange for demonstrated use of bicycling, we chose to use loss aversion, framing effects, and hyperbolic discounting to encourage cyclists to stay engaged with the system:

- **Loss Aversion:** People tend to prefer avoiding losses over acquiring equivalent gains. Cyclists will receive periodic notifications indicating how much money they could earn that day (or week) if they commuted or ran errands by bike, based on their past behavior and rewards.
- **Framing Effects:** People react to a particular choice in different ways depending on how it is presented. Instead of bundling biking rewards into another payment (e.g. including the rewards in a monthly paycheck) or deducting them from an expense (e.g. reducing health insurance by the reward amount), people will get an outright payment that directly corresponds to their recent earnings.
- **Hyperbolic discounting:** People prefer rewards that happen sooner rather than later, even if the rewards have the same actual value. Instead of offering a

²Finkelstein, Eric, Ian C. Fiebelkorn, and Guijing Wang. "The Costs of Obesity Among Full-time Employees." *American Journal of Health Promotion* 20.1 (2005): 45-51. Web. 24 Apr. 2017.

³Rashad, Inas. "Cycling: An Increasingly Untouched Source of Physical and Mental Health." NBER Working Papers 12929 (2007): n. pag. National Bureau of Economic Research, Inc. Web. 24 Apr. 2017.

⁴Drennen, Emily. *Economic Effects of Traffic Calming on Urban Small Businesses*. Rep. Department of Public Administration, San Francisco State University, 2003. Web. 24 Apr. 2017.

⁵Measuring the Street: New Metrics for 21st Century Streets. New York City Department of Transportation, Oct. 2012. Web. 26 Apr. 2017.

monthly or yearly reward, this system makes incremental payments close to the time people cycle. Cyclists are able to receive financial rewards close to the time that they perform the actions that earn those rewards, so that they associate their actions with the financial rewards and stay engaged with the system.

The current implementation of this system features a static rewards scheme for cycling—\$1 per day, plus \$0.25 per mile biked. However, the nature of the system is such that organizations could easily implement flexible and customized rewards that are responsive to the behavior and activities of the cyclists.

5.2 System Architecture

At the heart of this thesis is a distributed software system that uses a blockchain database. The software system links embedded devices (e.g. the bicycle sensors described in Chapter 4) that are owned by cyclists as well as traditional desktop computers which are owned and operated by the sponsoring organizations. These devices share and communicate over a blockchain network. Each device runs an Ethereum node; the embedded devices run a light node that stores a subset of the blockchain’s data, while the organization’s computers run a full node that stores the entire history of the blockchain, as well as the smart contracts which dictate the interactions of entities in the system. The high-level architecture of the system is pictured in Figure 5-1.

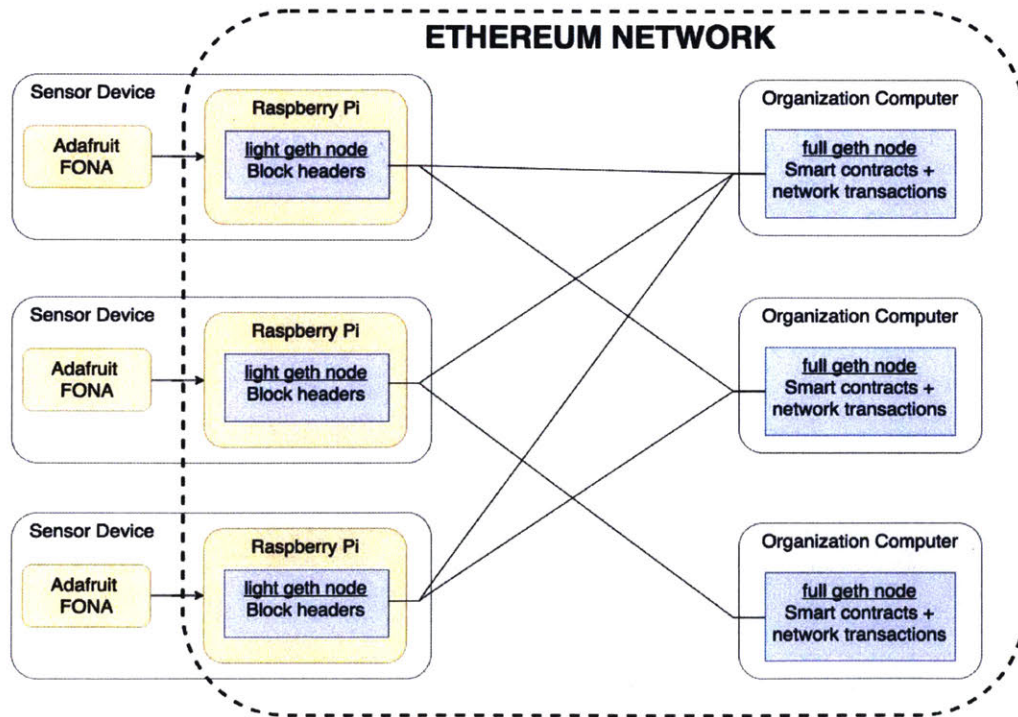


Figure 5-1: High-Level System Architecture Diagram

5.2.1 Blockchain Network Design

This thesis implements an Ethereum network with two different types of nodes: cyclist or user nodes, which are run on embedded devices, and organization nodes, which are run on traditional desktop hardware. In this network, each node runs an Ethereum Virtual Machine (EVM), which can "read and write...both executable code and data" to a blockchain⁶. Organizations run full Ethereum nodes that store every transaction on the blockchain, while cyclists' sensor devices run *light nodes*, which store a subset of the blockchain—usually block headers—to ease the data storage burden of running a full node.

Light nodes or light clients are an emerging blockchain technology that have broadened the range of devices that can participate in blockchain networks. They use a process called Simple Payment Verification (SPV) to verify the information they receive from the blockchain network. SPV is an algorithm that allows nodes to easily verify the validity of a transaction with only a subset of the blockchain ledger data; this algorithm is possible due to the structure of Merkle trees, which are the data structure used to hold the blockchain's data⁷. Light clients can use SPV to ensure they can trust their interactions with the blockchain, but cannot participate in mining or maintaining the network. A recent post on the Ethereum blog makes the following comment about light nodes:

The only meaningful solution for running a blockchain on tiny embedded devices is for them to become light clients, where they do not bear the full burden of sustaining the network, but rather only bear the burden of their own operation. Not only is this beneficial for the small devices, but it also benefits the network as a whole as it removes slow links and thus makes the core network smaller, tighter and more performant⁸.

A *user account* runs on top of each Ethereum node in the system. Accounts act as pseudonymous identities for users of the network; they provide a public address used for interacting with the network, can store the Ether cryptocurrency, and have a public-private key pair that is used to verify and secure that node's interactions with the network. Accounts can be accessed through a command line tool, but in this system, cyclists and organizations alike interact with their accounts through a decentralized browser-based application (called a "dapp"), that connects to the account running on their Ethereum node via a remote procedure call, and allows users to trigger certain actions and send data to the network.

The heart of the blockchain network, and the reason the Ethereum platform was selected for use in this project, are the *smart contracts* that define system logic, manage interactions, and live and run on the blockchain. Unlike the Bitcoin blockchain, the Ethereum blockchain, via the EVM, is able to execute code that lives on the

⁶"Ethereum Homestead Documentation." Ethereum Homestead 0.1 documentation. Ethereum, n.d. Web. 29 Mar. 2017.

⁷"Light client protocol." Ethereum/wiki. GitHub, 9 May 2016. Web. 13 May 2017.

⁸Szilágyi, Péter. "Whoa...Geth 1.5." Web log post. Ethereum Blog. Ethereum Foundation, 17 Nov. 2016. Web. 25 Apr. 2017.

blockchain in modules called "smart contracts". Smart contracts are pieces of programming, similar to object classes, that define and enforce rules around the exchange of value in the network.

Instances of contracts can be created by other contracts or through executed JavaScript commands, and are stored on the blockchain at a unique address. Contracts contain code that they execute when they are instructed to do so by the network. The system interacts with contracts through the use of *transactions*, which are similar to function calls in a programming language; the transaction indicates a method for the contract to execute, and contains data arguments that are relevant to the contract's code execution. Transactions are addressed to the contract's address on the blockchain and can be sent by user accounts or by other contracts. The mechanics of contract creation and interaction are discussed in detail in Appendix B.

In the case of our network, there are three main types of contracts. Sensor contracts represent sensor devices, and there is an instance of the contract mapped to each physical device. Likewise, User contracts represent cyclists, and Organization contracts represent organizations (e.g. city governments or health insurance companies). On behalf of its corresponding real-world entity, contracts manage interactions with other entities in the network and store pieces of important metadata. The relationship between the smart contracts in our system is pictured in Figure 5-2; contract implementation and interaction is discussed further in Sections 5.2.2 and 5.3.

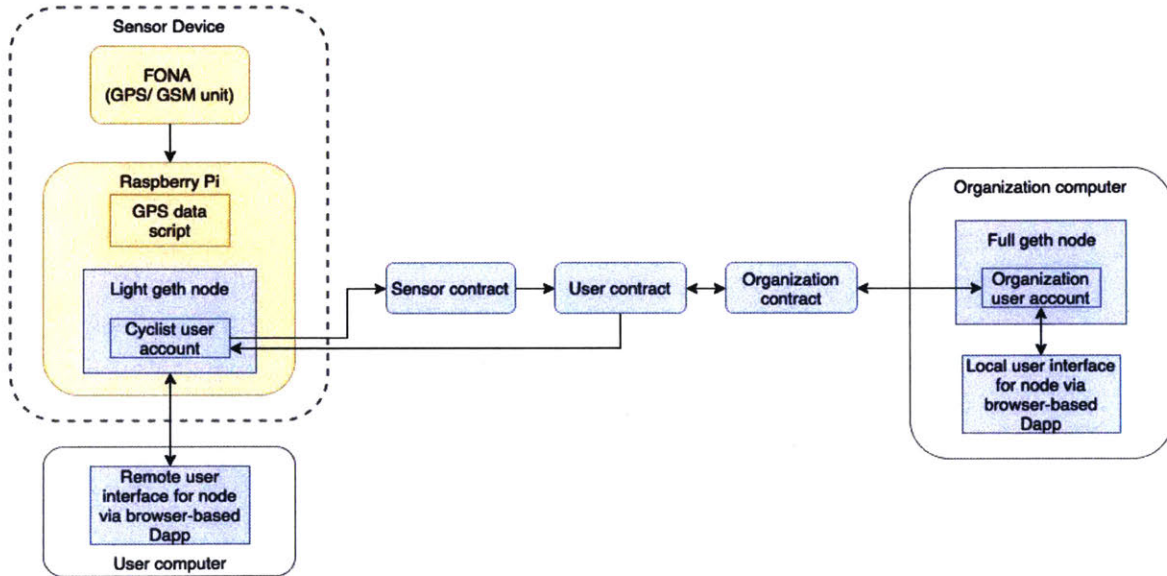


Figure 5-2: Software Architecture Diagram

5.2.2 System Components

This section describes, in further detail, each of the system components shown in Figures 5-1 and 5-2, and their relationships and interactions with each other.

- **Sensor Device:** This physical device consists of a Raspberry Pi single-board computer connected to a FONA GPS and Cellular GSM module. The device connects to the network via the General Packet Radio Service (GPRS) capabilities of the FONA module, which is connected to a 2G cellular network. The hardware and connectivity specifics of this device are discussed in further detail in Chapter 4. An embedded version of the geth Ethereum client is installed on the Raspberry Pi, which is running a light Ethereum node with the cyclist's Ethereum account as its identity. The Raspberry Pi collects GPS data using a script running periodically on the Raspberry Pi. The script executes and collects data only when it senses that the bike is moving by looking at the power input from the bicycle's bottle dynamo. After collecting GPS readings, the script uses the Vincenty's Formula to calculate distance travelled⁹. While the bike is collecting GPS data, this script will periodically make calls to the Ethereum node in order to send distance and location data to the blockchain network and request rewards.
- **Sensor contract:** There is one instance of the Sensor contract for each sensor device. This contract lives on the blockchain at a specific address and is an interface point for the Raspberry Pi device, managing location and distance information that comes from the device. The contract stores important state information but no long term data. When the contract receives a transaction with GPS and distance data from the sensor, it updates its state fields and passes the data along to the user contract in another transaction.
- **User contract:** There is one instance of the User contract for each cyclist enrolled in the network. This contract lives on the blockchain at a specific address and stores state information about the associated cyclist, such as the distance they've travelled, and lifetime earnings. The contract also stores a list of all organizations that currently participate in the rewards network. When the User contract receives a transaction from the Sensor contract, it will update its state variables, and send, as a series of transactions, requests to some or all of the organizations' contracts. These transactions will include distance metadata and solicit rewards from the organizations. Based on the data in the request and the settings in their contracts, organizations will make a reward offer to the cyclists, and send the offer information back to the User contract in another transaction. The User contract will select one of the offers, and then solicit an Ether transfer of the specified amount from the corresponding organization's contract to the User contract's balance. After the balance in the User contract reaches a certain threshold, the User contract will initiate an Ether transfer between the contract and the user's Ethereum account.
- **Organization contract:** There is one instance of the Organization contract for each organization participating in the network. This contract serves as an Ether wallet for the organization and lives on the blockchain at a specific

⁹GeoPy Documentation Release 1.11.0. N.p., n.d. Web. 2 May 2017.

address, managing how the organization spends money. This contract contains preset rules on how the organization would like to spend money, daily limits, and the conditions of any special competitions or promotions that they would like to offer to cyclists. The organization contract offers rewards based on the data sent in transactions from the User contracts, and also dispenses Ether rewards to approved user Ethereum accounts. The settings of each Organization contract are managed by the organization's respective Ethereum account via the browser-based user interface.

- **User interface (cyclists):** Cyclists are able to interact with their Ethereum account through a browser-based Dapp (decentralized application). The Dapp, which might run on a laptop or mobile device, connects remotely to the Ethereum client running on their sensor device using a remote procedure call (rpc) protocol. The Dapp itself stores no blockchain data; it merely uses an authenticated procedure call to interface with cyclist's user account. The Dapp provides a simple visual interface that lets cyclists view their distance statistics and rewards balance, and offers a "cashout" option that would allow them to initiate a transfer of their Ether to U.S. dollars.
- **User interface (organizations):** Organizations are able to interact with their Ethereum account through a browser-based Dapp as well. Their Dapp is run on the same device as their Ethereum account, and connects to the local Ethereum client process using a remote procedure call. As with the cyclist's interface, the Dapp itself stores no blockchain data; it merely uses an authenticated procedure call to interface with organization's user account. The Dapp lets organizations set their payment preferences, define payment limits, and design promotions or contests to engage cyclists. These preferences are then deployed to the organization's contract via a transaction sent by the organization's Ethereum account.

5.3 Blockchain Implementation

A proof-of-concept implementation of the blockchain network was developed for this thesis. Because blockchain and Ethereum publish an open specification, there are many different open-source libraries and software clients available for building blockchain-based applications. This section discusses the design decisions and technology selection process behind the blockchain database and application built for this thesis work. This section also covers, in further detail, the mechanics of initializing and running the blockchain network and adding new users.

5.3.1 Ethereum Network Operation & Design

The first design decision we made was to use a private Ethereum test network for this project. Test networks—or *testnets*—are versions of the Ethereum blockchain that make it easy to develop and prototype applications because they support fast transactions and provide free Ether that can be mined quickly. Ethereum provides a

full public testnet that is essentially the same as the real blockchain, but doesn't use real money or currency.

Instead of using the public Ethereum testnet, however, we chose to run a local, private testnet, which is particularly fast and lightweight because it does not require nodes to download the entire blockchain history of the testnet. Our project does not require nodes to interact with external nodes on the Ethereum blockchain, so this scheme was appropriate for our proof-of-concept deployment. In our test network, each cyclist or organization ran their own Ethereum node (either a light node or a regular node), and interacted with other entities in the network via smart contracts and transaction calls to these contracts. Because mining is free and fast in a testnet, the organization nodes served as miners for transactions occurring in the system. If this project were to be deployed to the real Ethereum blockchain, the nodes would rely on dedicated miner nodes to mine transactions; a portion of the funds invested in the system by organization nodes would go towards paying the mining fees.

In order to run an Ethereum network, we had to first select an Ethereum client. An Ethereum client is software which implements the Ethereum specification, runs the Ethereum Virtual Machine (EVM), and provides a user interface, either through the command line or through a GUI. There are many different Ethereum clients, which each implement the Ethereum protocol. Based on the recommendations of the Ethereum community, the availability of client-specific documentation, and the support for running on embedded devices, we chose to use *geth* on both desktop and the Raspberry Pi-based sensor device. Geth is an Ethereum client written in the Go programming language that provides a command-line interface for running and interacting with an Ethereum node.

Ethereum nodes provide a JavaScript runtime environment, which can be used interactively, through a console, or non-interactively, by passing prepared scripts to a running instance of the client¹⁰. We used the console for testing and development, but the final version of the system uses scripts to instantiate and interact with Ethereum nodes. In order to run a private testnet, we had to hard code our own *genesis block*, which is the first block of the private blockchain. All nodes connected to the private blockchain must have identical genesis blocks; otherwise, they will not be able to reach consensus on the contents of the blockchain and send transactions on the same network.

Another key element of running a private blockchain is the *networkid*, which is provided as an argument on the command line, and identifies the network. All nodes wishing to connect to the same network must use the same network id parameter when starting a geth instance. Once a node is running, it can connect to peer nodes by passing the peer's network address into the `admin.addPeer()` command, which can be run interactively or via a script. In our case, we hard-coded the network addresses of static organization nodes into the `static-nodes.json` file for each cyclist node. Upon being instantiated, nodes will connect to the static nodes indicated in the file, and will attempt to re-establish a connection with them if they are disconnected¹¹.

¹⁰"JavaScript Console." Go Ethereum Wiki. N.p., n.d. Web. 01 May 2017.

¹¹"Connecting to the network." Go Ethereum Wiki. N.p., n.d. Web. 01 May 2017.

The network's node discovery process is discussed further in the subsequent section.

Example 5.1 demonstrates the commands used to: (1) initiate an Ethereum node with a custom genesis block, and (2) start an instance of the Ethereum client and open the interactive console. Please refer to Appendix A for a detailed description of the procedure used to install geth, run a private testnet, and connect to peer nodes.

```
geth --networkid 4507 --identity node1 --verbosity 3 --nodiscover
    --nat none --datadir=~/.code/ethereum/node1
    init ~/.code/ethereum/testnet/genesis-block.json

geth --networkid 4507 --identity node1 --port 35353 --rpcport 8901
    --verbosity 3 --nodiscover --nat none
    --datadir=~/.code/ethereum/node1 console
```

Example 5.1: Initializing and instantiating an Ethereum node on a private blockchain

5.3.2 Node Discovery

Once there are several nodes running on the same network, each with an active geth instance, they must be able to find and connect to each other on the network in order to interact. Though we can connect nodes manually, using the `admin.addPeers()` command, as described above, we wanted to build a more automated process that would allow deployed sensor nodes to join the network without needing manual intervention. For purposes of our proof-of-concept system, we orchestrated the node discovery process around a static organization node, assuming that this static node would always be up and running with a constant network address. This assumption may not be sophisticated enough for a larger network deployed in real-world conditions; however, it was appropriate for the scope of this project.

Given the assumption of an organization node with a constant network address, we also assume that node is connected to an Organization contract that also has a constant network address. Both the node address and the contract address are hard-coded on the sensor hardware that is distributed to new users. The static node's address is placed in the `static-nodes.json` file so that the new sensor node will connect to the static organization node when it starts running, and will attempt to reconnect to the static node even if it is disconnected. The static nodes file and associated functionality is a built-in feature of the geth Ethereum client. Connecting to the static node will give the new sensor node access to the rest of the blockchain network and allow it to add other peer nodes.

At the same time, the newly-created User contract will send its address to the hard-coded Organization contract, who will then broadcast the User contract's address to the rest of the network so that the new User contract is able to send and receive transactions from other contracts on the network. We developed this somewhat circuitous initialization procedure because we need to connect the new device to the rest of the network at both the node level, and at the contract level. However, nodes are not designed to store information easily, and they do not have any intrinsic

connection to a contract; thus, the system stores and connects to the static organization node and contract through different processes. Possible improvements to this procedure are discussed in Section 7.2.1.

5.3.3 Smart Contracts

Once the private blockchain testnet is running, and the nodes in the network are connected with each other, smart contracts provide a mechanism for the network to execute logic and manage the interactions between the nodes. Contracts are written in *Solidity*, a "contract-oriented, high-level language whose syntax is similar to that of JavaScript and...is designed to target the Ethereum Virtual Machine (EVM)"¹². The structure of smart contracts is similar to the structure of classes in object-oriented languages; contracts can contain declarations of variables, functions, events, and enum types, among others.

There are three types of contracts in our system: User contracts, Sensor contracts, and Organization contracts. The high level functionality of these contracts is described in Section 5.2.2, and the relationships between and data fields of the contracts are shown in Figure 5-3. There is a one-to-one relationship between Sensor contracts and User contracts, and a many-to-many relationship between User contracts and Organization contracts. User contracts maintain a list of all organizations in the system, and vice versa. Our contracts also keep track of state variables, such as lifetime distance, recent distance, account balance, and the network addresses of contracts with which they have a relationship.

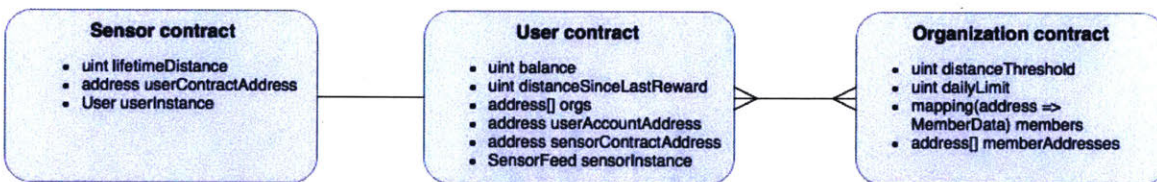


Figure 5-3: Contract Relationships and State Variables

Contracts are written like object classes, and can use functions to manage interactions with other contracts. For instance, Example 5.2 shows the code used by an Organization contract to manage its list of User contracts and update the distance variable stored for each user. We used the *Truffle* framework to develop the smart contracts for this system. Truffle is a software framework that handles contract compilation and deployment, facilitates testing, and exposes an interactive console for development¹³. For development and testing, we ran Truffle on top of *testrpc*, a lightweight JavaScript Ethereum client that simulates the operation of a real Ethereum node. To deploy the contracts to our private geth testnet, we started a geth network using parameters that matched the parameters in our Truffle project's `truffle.js` configuration file.

¹²"Ethereum Homestead Documentation." Ethereum Homestead 0.1 documentation. Ethereum, n.d. Web. 29 Mar. 2017.

¹³"Truffle Framework." Truffle Framework. N.p., n.d. Web. 01 May 2017.

For example, our configuration file specifies that it is looking for a network on port 8545 with network id 4507. By starting a geth network with flags ("--networkid 4507 --rpcport 8545") that match those parameters, the Truffle framework will automatically point its contract deployments to our geth testnet.

```
function getMemberExist(address member) constant returns (bool exist) {
    return members[member].exists;
}

function getMemberDistance(address member) constant returns (uint d) {
    return members[member].distance;
}

function getMemberAddresses() constant returns (address[] members) {
    return memberAddresses;
}

function update(uint newDistance) {
    members[msg.sender].distance += newDistance;
    if (members[msg.sender].distance >= REWARD_THRESHOLD) {
        reward(msg.sender);
    }
}
```

Example 5.2: Solidity contract code for managing list of users

In order to interact with and execute the code in the contracts, we must send transactions to the contracts, either from other contracts or from an Ethereum account. Transactions are similar to function calls, and include the name of the function the contract should execute, as well as arguments to that function. Example 5.3 demonstrates how a contract transaction is made from within another contract.

```
function sendDistanceToOrgs() {
    for (uint i = 0; i < orgs.length; i++) {
        Organization o = Organization(orgs[i]);
        o.update(distanceSinceLastReward);
    }
}
```

Example 5.3: Sending a transaction to an Organization contract from a User contract

In order to send a transaction from an Ethereum account or Ethereum node, the node must instantiate a contract object using the contract's network address, and then call the transaction on that object. This JavaScript code can be run in the Ethereum client's interactive console, or as a script that is loaded and executed by the node. Example 5.4 demonstrates how a Sensor contract object is created from the contract's Application Binary Interface (ABI) and then used to create an object instance at a particular address. A transaction call to update the Sensor contract's distance data field is then called on that object instance. Further code examples and

detailed descriptions of our smart contracts code can be found in Appendix B.

```
var sensorabi = [
  {
    "constant": true,
    "inputs": [],
    "name": "getUserInstance",
    "outputs": [
      {
        "name": "u",
        "type": "address"
      }
    ],
    "payable": false,
    "type": "function"
  },
  ...
  {
    "constant": true,
    "inputs": [],
    "name": "getUserContractAddress",
    "outputs": [
      {
        "name": "u",
        "type": "address"
      }
    ],
    "payable": false,
    "type": "function"
  },
  {
    "payable": true,
    "type": "fallback"
  }
];

var sensorfeedContract = web3.eth.contract(sensorabi);
sensorInstance = sensorfeedContract.at(sensorAddress);
sensorInstance.updateDistanceAndSendToUser(
  distance,
  {from: web3.eth.accounts[0]}
);
```

Example 5.4: Creating an instance of the Sensor contract object and calling a transaction on it to update the contract's distance data field

5.3.4 Browser-Based User Interface

To supplement the contract interaction logic that runs on the blockchain, we wanted to build a user-friendly interface to the network that would allow cyclists and organizations to view their account information and manage their preferences for interacting with the network. Fortunately, the Truffle framework integrates a convenient web application development environment that we used to develop and test a web-based front-end to our application. Truffle's web application framework relies on three primary technologies:

- **web3.js**: an "Ethereum compatible JavaScript API which implements the Generic JSON RPC specification," which is a remote procedure call protocol that uses JSON as a data exchange format¹⁴. Web3.js allowed us, given the correct address and port of an Ethereum node, to access information in the execution environment of that Ethereum node.
- **truffle-contract**: a node module that helps manage the asynchronous flow of JavaScript functions in Ethereum environments and simplifies the process of creating and interacting with contracts¹⁵.
- **webpack**: a module bundler that "takes modules with dependencies and emits static assets representing those modules"¹⁶. Webpack essentially simplifies package and library management in the development environment and build pipeline.

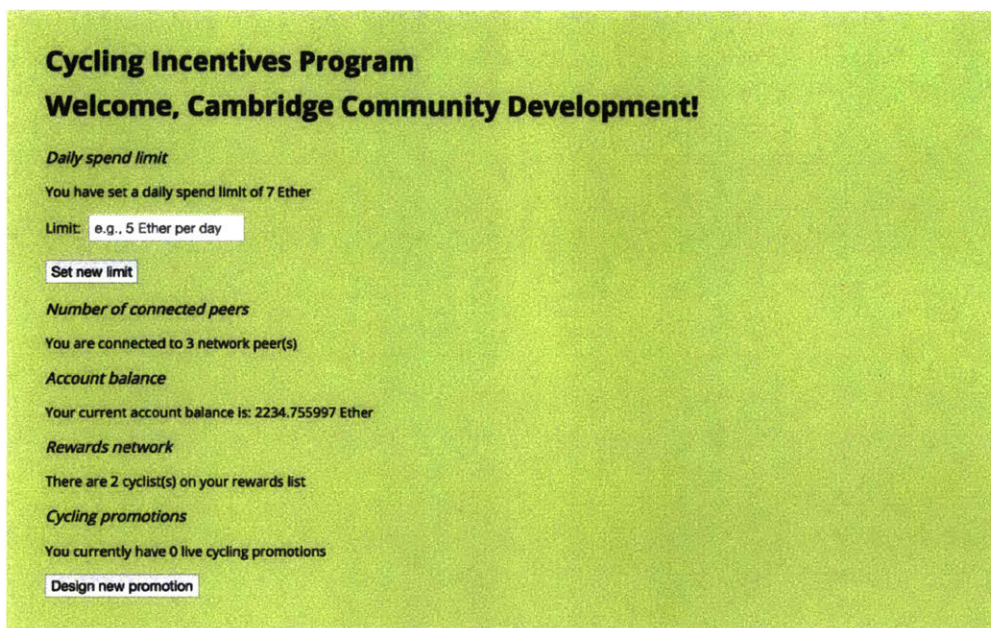


Figure 5-4: Web-Based User Interface for Example Organization

¹⁴"Web3.js." Ethereum/web3.js. GitHub, n.d. Web. 10 May 2017.

¹⁵"Truffle-contract." Npmjs.com. Npm, n.d. Web. 14 May 2017.

¹⁶Koppers, Tobias. "Webpack." Webpack module bundler. N.p., n.d. Web. 14 May 2017.

For purposes of our proof-of-concept deployment, we served the web application locally on our desktop hardware, which were able to do using the `truffle serve` command within the Truffle development environment. The application could connect to either a local geth node or a remote one (e.g. the node running on one of the sensor devices). This connection was established by using web3.js to create an `HttpProvider` object specified by the hostname and port of the desired geth node's remote procedure. The interface allows cyclists and organizations to manage their application preferences and perform simple actions like, for cyclists, "cashing out" their Ether earnings and exchanging them for U.S. dollars. Figure 5-4 pictures an example of the organization's user interface.

5.4 Embedded Software Implementation

The software running on the Raspberry Pi-based sensor device must collect GPS data, run a geth Ethereum node, and communicate with the system's Ethereum network. While the assembly and installation process for the Raspberry Pi is covered in Section 4.2.1, the following section explains the procedure used to collect data and allow the sensor device to participate in the blockchain network. Figure 5-5 visualizes the various scripts used to achieve this behavior and the relationships between them. Further detail and explanations of the code running on the Raspberry Pi are provided in Appendix C.

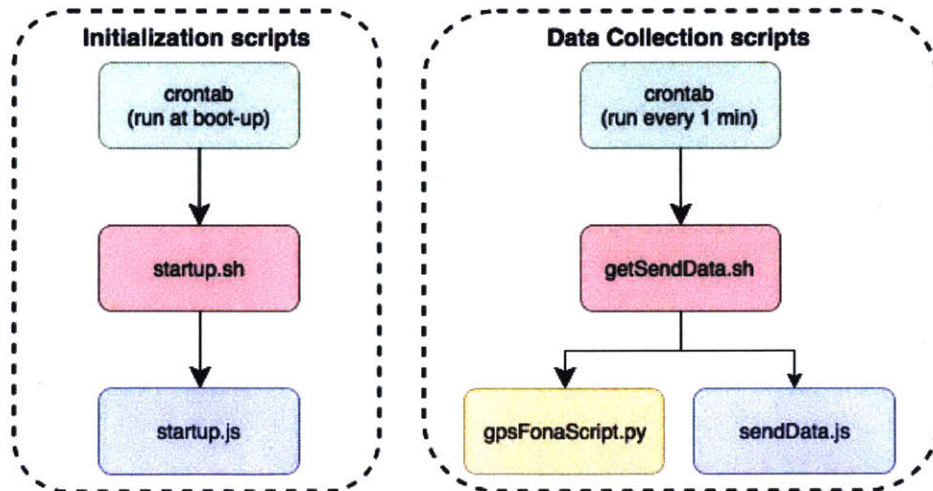


Figure 5-5: Raspberry Pi Script Relationship Diagram

5.4.1 Ethereum Node Initialization

The Raspberry Pi runs a light geth node that communicates with the rest of the Ethereum network over the GPRS connection provided by the FONA. When the Raspberry Pi is powered on, we want to check whether the device's blockchain node has been initialized yet, and initialize one if necessary, then run an instance of a geth

client. We achieve this behavior through a bash script, `startup.sh`, that runs when the Raspberry Pi is powered on. To run the script when the Raspberry Pi turns on, a command running the script is placed in the device's `crontab` file, which uses the Raspberry Pi's `cron` scheduling functionality to run different commands at specified times. In `startup.sh`, we check if the device is being turned on for the first time ever, or if it is merely restarting after being turned off or losing power. In the first case, it will need to create a new Ethereum account and initialize the blockchain using the genesis block. In both cases, the script starts an instance of the `geth` Ethereum client which will allow the device to participate in the blockchain network.

Once the `geth` client is running on the Raspberry Pi, its JavaScript runtime environment can be used to interact with contracts on the network by sending transactions. When the node is being initialized for the first time, we use the code in Example 5.5 to load and execute a JavaScript script, `startup.js`, that performs set-up functions like creating and initializing the Sensor contract and User contract that correspond to the new node. Further details on these scripts can be found in Appendix C, and a visualization of the relationships between these scripts can be found in Figure 5-5.

```
geth --exec 'loadScript("ethereum/testnet/scripts/startup.js");'  
attach ipc:ethereum/node3/geth.ipc
```

Example 5.5: Loading and executing a JavaScript script in `geth`'s non-interactive mode

5.4.2 GPS Data Collection

In order to record the activity data that cyclists will eventually leverage to earn financial rewards, our system needs to periodically collect location data and calculate the distances that cyclists have biked. To achieve this behavior, we use the `crontab` scheduling functionality of the Raspberry Pi to periodically run a bash script, `getSendData.sh`, that in turn runs a python script, `gpsFonaScript.py`, that uses the FONA module to collect GPS data.

The bash script, `getSendData.sh`, is scheduled to run every minute. Before calling the FONA module, this script checks the power input from the bicycle at pin 12 of the Raspberry Pi to check whether the cyclist is riding or not. If the pin is powered (or has been powered recently), indicating that the cyclist is riding, the Raspberry Pi uses the `gpsFonaScript.py` Python script to communicate with the FONA module. The script opens a serial port to communicate with the FONA module, and uses the AT Command set to turn on the GPS service, check for a GPS fix, and gather GPS NMEA sentences¹⁷. The script parses the NMEA sentences, logs the cyclist's current latitude and longitude, and uses the `geopy` library to calculate their distance from the previous GPS data point using Vincenty's Formula. This procedure is partially demonstrated in Example 5.6, and explained in further detail in Appendix C.

¹⁷AT Commands Reference Guide. N.p.: Telit Wireless Systems, 2006. Pdf.


```

def checkFonaOn():
    return writeParseReply("AT+CGNSPWR?\n", "+CGNSPWR: ")

def turnFonaOn():
    return writeCheckReply("AT+CGNSPWR=1\n", "OK")

def turnFonaOff():
    return writeCheckReply("AT+CGNSPWR=0\n", "OK")

def checkGPSFix():
    return writeParseReply("AT+CGNSINF\n", "+CGNSINF: ").split(',')[1]

```

Example 5.6: AT Commands used to interface with the FONA module's GPS service

5.4.3 Blockchain Network Communication

After the Raspberry Pi has retrieved and logged GPS and distance data, the bash script uses the `geth attach` command to run a final JavaScript script, `sendData.js`, that verifies the most recently calculated distance data. The script verifies the data by calculating the cyclist's speed during the time the data was collected, and checking that the speed falls within a reasonable speed range for a cyclist. This validation is performed to prevent cyclists from faking their data by, for example, driving their bicycle around in the back of a car. Once the data has been verified, the script will send it to the node's corresponding Sensor contract. The Sensor contract will log this data and forward it to the User contract so that the cyclist may solicit a reward for their cycling activity. This contract interaction flow is covered in greater detail in Appendix B.

Chapter 6

Development, Deployment & Evaluation

6.1 Development

This thesis, as a technical endeavor, consisted of designing, building, and integrating a number of complex technical subsystems, including the blockchain network itself, the smart contracts that run on top of the blockchain, the GPS data collection module, the device’s power system, and the device’s communication layer. We took advantage of the project’s modular design, and worked incrementally by first building and testing each subsystem, then integrating two related subsystems.

For example, after ensuring that the sensor device could collect GPS data, and separately testing that it could use the FONA module to communicate over a GSM cellular network, we wanted to send GPS data over the GSM cellular network. Often, these integration steps introduced further technical challenges that had to be addressed in their own right. In this case, the challenge was instructing the Raspberry Pi to utilize two different functionalities of the FONA module over the same communication channel. The Raspberry Pi only has one hardware serial connection, so we had to make sure that the communications meant to control the FONA’s GPS functionality and those meant to control the GSM functionality did not interfere with each other. We were able to accomplish this by encapsulating the control of each functionality into a single system command, so that we could toggle between the two feature sets with a single line in a bash script.

On the software side, the development of the blockchain network and smart contracts also demonstrates our incremental, modular approach. These components were developed separately, and then integrated once each could function on their own. Blockchain network development consisted of installing the geth Ethereum client, experimenting with different private test network settings, and writing scripts that would automatically initialize and run an Ethereum node using geth. This process is covered in detail in Appendix A. Smart contracts were developed separately, in the Truffle development environment, on top of the testrpc Ethereum client, which merely simulates an Ethereum environment. To merge the two subsystems, we first

adjusted the Truffle configuration settings to point to the geth client instead of the testrpc client, so that we could use the Truffle framework to deploy the contracts to our own test network. Eventually, instead of using Truffle for contract deployment, we developed scripts that would create and deploy the contracts to the geth network independently.

After incrementally integrating subsystems in this manner, we finally integrated the entire project, using the FONA module to collect GPS data and allow the sensor device to connect to our blockchain network, and using power from the bicycle to recharge the sensor device. In the latter stage of our development process, significant energy was spent automating subsystem interactions, building a smooth initialization process for users, and ensuring that the system was robust to technical failure.

For example, the Raspberry Pi runs two scripts (the bash script `startup.sh` and the JavaScript script `startup.js`) when the device is powered on. These scripts initialize the Ethereum network and connect the Raspberry Pi to other members of the network. These scripts mean that a potential user would simply be able to turn the device on without needing to perform any setup or calibration. These scripts also ensure that if the device loses power, it will be able to seamlessly reconnect to the same blockchain network when it receives power again. Careful consideration of user behavior and points of potential technical failure was built into the development of this system.

6.2 Deployment & Testing

6.2.1 Assumptions

To facilitate deployment, we make several assumptions about the conditions of the network. First, we assume that there is at least one static organization node—with an associated organization contract—that is always up and running. This node allows new system nodes to connect to the rest of the network and discover other nodes. As the system developers, we take responsibility for maintaining and servicing this node. Second, we assume that as "manufacturers" or producers of the sensor devices, we would have access to each piece of hardware before it was released to a cyclist, so that we could put custom embedded software on the device and hard-code a few important configuration parameters, like the blockchain node's identity name and the address of the permanent organization node and organization contract. We acknowledge that these assumptions, especially the first one, are somewhat restrictive, and discuss technical improvements that would allow the system to operate without these assumptions in Section 7.2.1.

6.2.2 Procedure & Testing

For the purposes of this thesis, our test deployment consisted of running organization nodes on two laptops, and installing a sensor device on one bicycle. At start-up, the sensor device, as discussed in Section 6.2.1, had been programmed to run the

embedded software scripts that allow it to initialize a blockchain node and connect to the blockchain network. The sensor node also has several hard-coded configuration settings that allow it to distinguish and identify itself on the network. These scripts and settings are covered in detail in Section 5.4 and Appendices B and C.

At the beginning of the test deployment, the sensor node is powered on using the portable battery back, and runs an initialization script that creates a blockchain identity for the node, initializes the blockchain using the hard-coded genesis block, starts a running instance of an Ethereum client, and creates a unique instance of the Sensor contract and User contract for the cyclist. Using the hard-coded address in the `static-nodes.json` file, the node connects to the static organization node, which gives it access to the rest of the blockchain network and allows it to add other peer nodes. The newly-created User contract then sends its address to the address of the hard-coded Organization contract, who broadcasts the User contract's address to the rest of the network so that the new user contract is able to send and receive transactions from other contracts on the network.

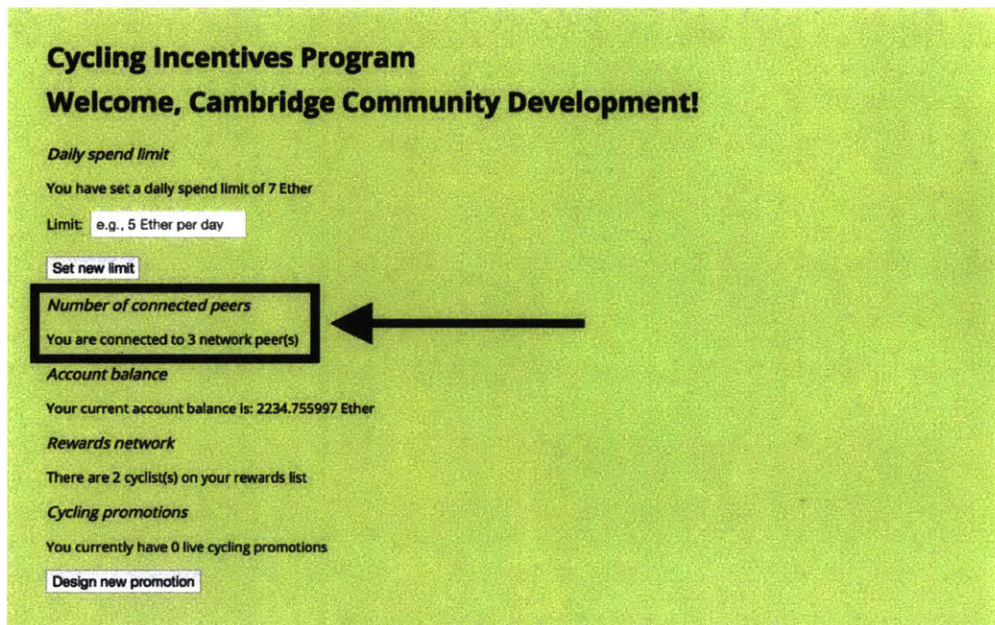


Figure 6-1: Web-Based User Interface for Example Organization Highlighting Connected Nodes

After powering on the device nodes, we were able to verify that they were connected to each other using the Dapp's GUI, as shown in Figure 6-1. We then took the sensor devices on 30-minute test ride, where the sensors were collecting GPS data and sending it to the blockchain network using the FONA's cellular network connection. At the end of the ride, we were able to examine the nodes using the command-line interface as well as the graphical interface to verify that the sensor had collected GPS data, and interfaced with the rest of the blockchain network to solicit rewards. A portion of the collected data is shown in Example 6.1; the data includes latitude, longitude, the calculated distance from the previous reading, and the reading times-

tamp. On a ride of about 3 miles, the cyclist in our test was able to earn 0.0203 Ether, or \$1.75. The cyclist's collected GPS data points are displayed in Figure 6-2. This test deployment verified the system's core technical components: the ability to start a blockchain node, to run a connected blockchain network, to collect and distribute GPS data, and to send and receive behavior-based financial rewards through that network.

#	Latitude	Longitude	Distance from previous	Timestamp
42.353432,	-71.106372,	0.415531268924,	2017-05-11 07:23:02	
42.356838,	-71.108623,	0.261813844869,	2017-05-11 07:27:03	
42.358715,	-71.106602,	0.165795011917,	2017-05-11 07:28:02	
42.359357,	-71.105987,	0.0543571645084,	2017-05-11 07:29:03	
42.361265,	-71.103922,	0.168870460548,	2017-05-11 07:30:03	
42.363443,	-71.101977,	0.180309384144,	2017-05-11 07:31:02	
42.363545,	-71.099905,	0.106292979784,	2017-05-11 07:32:03	
42.363275,	-71.097152,	0.142145071984,	2017-05-11 07:33:02	
42.363160,	-71.094935,	0.113759447939,	2017-05-11 07:34:03	
42.362905,	-71.091942,	0.15421166443,	2017-05-11 07:35:03	
42.362928,	-71.090192,	0.0895922255176,	2017-05-11 07:36:02	
42.360145,	-71.087518,	0.235867972352,	2017-05-11 07:38:03	

Example 6.1: Sample Data Collected from Test Deployment

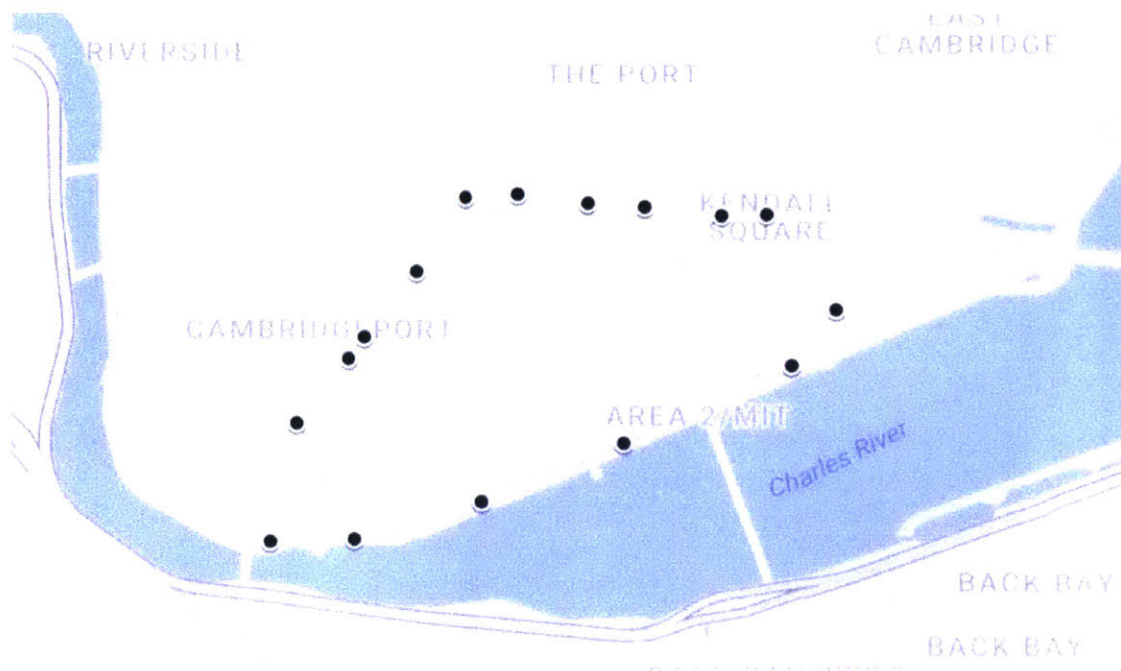


Figure 6-2: Cyclist's GPS Data Points from Test Deployment

6.3 System Evaluation

Given the novelty and significant technical complexity that was proposed by this thesis, our evaluation centered on, first, verifying whether it was possible to build a system that achieved the technical goals of the proposal, and second, considering the technical challenges and barriers to deployment. We believe the work presented here opens many avenues of further inquiry, which are discussed in greater detail in Section 7.2.

We were successfully able to design, develop, and deploy a blockchain-based financial incentives program that allows cyclists to monetize their cycling habits. The system consists of location sensors that are mounted on and indirectly powered by bicycles. These sensors collect GPS data and use a GSM cellular network to communicate with a wider network of cyclists and organizations. This network is supported by a blockchain database, and the relationships between different entities in the network are managed by smart contracts deployed on the blockchain. We were able to verify the successful operation of this proof-of-concept technical system through the test deployment discussed in Section 6.2.2.

The development of this system gave us insight into technical challenges related to the deployment of blockchain networks and distributed hardware sensing systems. These fields have developed independently in the past, and the interest in combining the two—using a blockchain database to manage and store data from distributed sensor networks—is largely new and unrealized.

While blockchain technology and sensor networks are similar in that they are both predicated on the practice of coordinating distributed computing efforts, they are at odds in other ways. For example, in order to maintain its distributed consensus mechanism, most blockchain networks require that all participating network nodes store a full local copy of the blockchain ledger. For a space-constrained distributed device, this requirement could be prohibitive. Similarly, certain blockchain operations, like mining or examining the blockchain ledger to help the system reach consensus, are resource-intensive activities which could be prohibitive for a battery-powered sensor device. Furthermore, though the blockchain is robust to nodes periodically dropping their connection to the network, an intermittent network connection is not ideal for participating nodes. On the other hand, in many sensor networks, devices prefer intermittent connections because it helps them conserve battery life.

We were able to address some of these challenges through our design and testing framework. For example, we addressed the challenge of having sensor devices perform power-intensive blockchain operations by designing a system that would take advantage of the bikes’ motion to automatically recharge the sensor. Storage was not an issue for us in this proof-of-concept deployment because we used a private blockchain that has a very small transaction history. In a deployment to a live blockchain with a long transaction history, the use of light nodes—which store only a subset of the blockchain’s transaction history and thus reduce nodes’ storage burden—would be critical. For example, a geth light node running on the live Ethereum network cur-

rently requires only 200Mb of space, while a full node requires around 11Gb¹. The required bandwidth for a light node is about 1Mb per hour when running idle, with an additional 2-3kb for an average state or storage request.

In addition to the navigating trade-offs in storage and power, we also faced a challenge in the cost of running the system. The sensor nodes in the network use a 2G cellular data connection to communicate with the blockchain network. In our implementation, we used a Ting Mobile data plan to provide this cellular service². At an estimated bandwidth of 1Mb per hour, a light node might require approximately 720Mb of data per month. According to Ting Mobile's website, this data usage would cost \$16 per month, plus a \$6 flat fee per device per month, for a total of \$22 per month. Depending on the rewards a cyclist is able to earn, this cost could be prohibitive. For an effective deployment, we would need to mitigate these costs, either through technical (e.g. increasing connection downtime or using a different communication method) or social means (e.g. having organizations cover some of the data costs). While we did not explicitly address the challenge of cost in this thesis, it is certainly something that would need to be considered for a future deployment.

In addition to these technical challenges, another significant hurdle we encountered was the current status of the Ethereum development environment. Though Ethereum is well-documented compared to other blockchain programming environments, it is still a rapidly-evolving platform that suffers from some inconsistencies and inconveniences. These difficulties included outdated or incomplete documentation, and the need to integrate half a dozen different libraries, programming languages, and frameworks in order to build this project.

On occasion, we discovered undocumented and unexpected system behavior that significantly hindered our development. For example, at one point in the project, we tried to use the `"stopAutoDAG()"` function, which is defined in geth's reference library³. However, when calling the function from geth's JavaScript runtime environment, we received an "invalid function" error, saying that geth did not recognize this function. In this case, we were not able to solve the discrepancy between documentation and behavior, and had to turn to a different implementation technique. In another case, we discovered a bug in the current implementation of geth's light client that prevents light clients from connecting to peers. Because we use light clients on the sensor devices, this bug was a significant inconvenience. Fortunately, given that we were working with a very small test network, instead of the full Ethereum network, we were able to implement a workaround using regular nodes. Light nodes are currently a beta feature, and we fully expect that they will be ready to use in a future deployment of this system. In both cases, we plan to contact the geth developers to report the inconsistencies and errors we discovered.

Despite these challenges, we are excited about the potential of the proof-of-concept system we deployed and tested, which offers creative solutions to some of the technical challenges discussed here. Furthermore, by incorporating an understanding of

¹Felföldi, Zsolt. "Introduction of the Light Client for DApp developers." Ethereum Blog. N.p., 7 Jan. 2017. Web. 15 May 2017.

²"A smarter way to do mobile." Ting. N.p., n.d. Web. 15 May 2017.

³"JavaScript Console." Go Ethereum Wiki. N.p., n.d. Web. 01 May 2017.

behavioral psychology and previous work in urban studies into the mechanisms of the technical system, we believe the thesis work presented here suggests not only a novel engineering contribution but also a societal one.

Chapter 7

Conclusion

7.1 Contribution

The contributions of this thesis are to: (1) design and analyze a blockchain protocol that supports a financial incentives system for cyclists; (2) build and deploy a proof-of-concept implementation of the GPS sensor device and blockchain application; and (3) evaluate the protocol from a technical standpoint. This thesis draws upon research in urban planning, transportation engineering, behavioral psychology, distributed sensing, and blockchain development to present a novel engineering solution to increase the prevalence of urban cycling, which in turn, would improve productivity and public health in cities while fighting urban pollution and congestion. The financial incentives structure implemented in this thesis uses principals from behavior psychology—including nudging and loss aversion—to deliver a lightweight sociotechnical system that is likely to engage the people using it.

The motivation behind this work is simple: the positive externalities of urban cycling are well-documented, but rarely explicitly recognized. Cycling improves public health, increases worker productivity, boosts local retail sales, and mitigates congestion and pollution. Beyond the intrinsic benefit of these outcomes, cycling can have a positive financial impact on the institutions that nurture it. Nonetheless, the organizations and institutions that stand to benefit the most from cycling have done little to internalize the costs of cycling. The motivation behind this thesis is thus to give organizations a venue to recognize and reward the benefits of cycling, in hopes of encouraging further cycling and improving health, environmental, and economic outcomes for everyone.

In this thesis, we designed, developed, and deployed a proof-of-concept implementation of a blockchain-based financial incentives program for cyclists. The system connects cyclists and organizations who would like to invest in cyclists and reward cycling activity. Our implementation uses GPS-collecting sensor devices that are affixed to bicycles, and gather cyclists' activity data. These devices are connected to a blockchain network that connects cyclists to organizations, and allows them to receive financial rewards for their cycling activity. We developed a small test deployment of this system, and were able to verify that the system successfully implemented the

proposed behavior. This work represents the technical and design groundwork for a blockchain-based system that would manage the data generated by distributed urban sensing networks. Ultimately, we see this work as a building block for a new paradigm around urban data collection and monetization, which could be a powerful tool for rewarding and incentivizing sustainable behavior.

7.2 Future Work

Due to the novelty and complexity of this technical project, this thesis presents many avenues for further research and investigation. These ideas, discussed in the subsequent sections, range from engineering challenges to proposals for user testing to the advancement of a high-level vision for this incentives platform. Many of these ideas were considered or researched during the course of this thesis, but were not addressed formally due to time and space constraints.

7.2.1 Technical Improvements

The technical system produced through this thesis work, while functional, is still in a prototype phase. The following are technical improvements that would make the system more robust to environmental and resource challenges:

- **Improve charging system:** currently, only the FONA module uses the bicycle's bottle dynamo to recharge. Using the bottle dynamo to recharge the Raspberry Pi's battery pack would make the system significantly more portable and robust; we unfortunately were not able to implement this feature during the thesis due to time constraints. This engineering upgrade would involve designing a charging circuit for the Raspberry Pi's battery pack and developing a better understanding of the power output capabilities of the bottle dynamo. Alternately, it might be preferable to consolidate the power systems for the two boards by having them share a battery pack. Further investigation would be necessary to understand the requirements of that power solution.
- **Optimize communication protocol to improve system's energy- and cost-efficiency:** using the FONA's GSM module to communicate with the blockchain is a power- and cost-intensive solution. In the current system, the sensor device communicates with the blockchain every minute that the cyclist is moving. This protocol could be optimized to strike a better balance between the frequency of communication and the cost of carrying out that communication. For example, the sensor could still collect GPS data every minute, but only communicate it to the rest of the blockchain network every ten minutes. As a further improvement, we could replace the current components with more energy-efficient alternatives. For instance, 3G cellular networks, compared to the 2G GSM network used for this project, have been shown to "offer higher data rates with lower consumption in terms of energy per bit"¹.

¹Perrucci, Gian Paolo, Frank H.p. Fitzek, Giovanni Sasso, Wolfgang Kellerer, and Jorg Widmer.

- **Strengthen node discovery algorithm:** the current node discovery algorithm, as discussed in Section 5.3.2, relies upon a single static node that we assume is always running and connected to the network. New sensor nodes use the hard-coded address of this node to connect to other nodes on the network and to access the smart contracts on the network. While the node discovery mechanism on the actual Ethereum network is quite similar—new nodes look for network "bootnodes" that will broadcast their address to the rest of the network—we believe our method could be more robust and flexible if we used additional dedicated static "bootnodes" and hard-coded their addresses into the network protocol itself instead of onto the sensor hardware².
- **Improve sensor casing:** in order to be used for an extended period of time under real-world conditions, we would need to significantly update the sensor's casing and the device's mounting system. An ideal casing and mounting solution would be waterproof, tamper-proof, and well-suited for the geometry of the bicycle frame.

7.2.2 User Evaluations

The original proposal for this thesis involved the completion of several rounds of user testing, to understand how cyclists and organizations alike would respond to the financial incentives program we had developed. As the technical complexity of the project became more clear, we chose to focus our efforts on the engineering side of the proposal.

However, we regard testing with real cyclists and organizations as a crucial next step. We would like to develop a better understanding of how cyclists adjust their transit mode choice to immediate financial awards, and would also like to understand how organizations might regard the program. We are convinced that organizations would be willing to invest in the program due to the long-term economic and health benefits they could redeem, but would like to better understand how they would want to engage with the program on a daily or weekly basis. On that note, we would also be interested in offering organizations the opportunity to build more customized incentives structures, that, for example, rewarded cyclists for riding in a certain area or accumulating a "streak" of riding days.

7.2.3 Future Research

In the course of researching and developing this thesis work, we developed several avenues of inquiry that were particularly exciting. The following ideas would involve significant further research efforts and would represent major contributions to the current state of the field:

¹"On the impact of 2G and 3G network usage for mobile phones' battery life." 2009 European Wireless Conference (2009): n. pag. Web. 15 May 2017.

²"Connecting to the network." Go Ethereum Wiki. N.p., n.d. Web. 01 May 2017.

- **Proof of Pedal:** in most blockchain networks, nodes earn cryptocurrency (e.g. Bitcoin or Ether) through mining. In networks that utilize the "proof-of-work" consensus mechanism—including the two largest networks, the Bitcoin Blockchain and the Ethereum Blockchain—nodes must demonstrate a certain amount of computational power by solving hard mathematical problems, and in doing so, provide a mechanism for reaching network consensus and adding new blocks to the blockchain. In return for their work maintaining the network, they are rewarded with a small cryptocurrency payment. The idea is to make it difficult to add new transactions to the network to reduce the likelihood of a rogue node adding fraudulent transactions.

Proof-of-work is a cryptographically sound protocol that has proven quite successful in supporting several different blockchain implementations. However, under the proof-of-work protocol, mining computers expend an enormous amount of computational power on empty calculation; the mining problems are hard just for the sake of being hard, not because they are computing something important or difficult. In response to this paradigm, we suggest a new consensus mechanism called "proof-of-pedal". In this mechanism, a node's probability of mining a block, and therefore earning the cryptocurrency associated with the block, will be correlated to the mileage they have logged. Like "proof-of-work", it is hard to accumulate the proof (it must be done by actually biking), but it is easy to verify a cyclists' mileage, because the mileage must be recorded using the GPS sensor, which will only indicate mileage readings when the bike is actually moving from place to place at a certain speed.

- **Optimize Ethereum blockchain for distributed sensor development:** most computers that participate in today's blockchain networks in a meaningful way are computers with substantial computing resources and a direct power supply. These system requirements are often prohibitive for small, lightweight devices that might otherwise be well-suited to a distributed database solution. In conducting the work for this thesis, we often encountered situations where the strengths of the blockchain network and the capabilities of the sensor devices were at odds with each other.

Given this experience, and the significant enthusiasm around the idea of using blockchain databases with distributed sensor systems, we believe an exciting area of future research is developing a blockchain implementation (e.g. an Ethereum client) that is geared towards embedded and distributed devices. In particular, this implementation might focus on improving the stability of light nodes to reduce the storage burden on nodes, optimizing communication algorithms so that embedded nodes don't need to maintain a consistent network connection, and improving the blockchain deployment toolchain to better support software installation and development on embedded devices.

- **Clarify network access policies and improve verification:** in terms of access, the current status of the network is that anybody with the correct system parameters (e.g. network id) and node address (e.g. the enode address of the

static node) can join the network. While we want the network to be open so that it can scale organically and with low friction, a completely open network could lead to fraudulent use; that is, anybody could create their own Sensor and User contract and use them to solicit rewards from an Organization contract. Thus, there must be some sort of gating mechanism to limit who can join the network, or verify participants once they have joined the network.

One option would be to completely control distribution of system hardware, keep a list of identifiers for the official hardware, and only allow network transactions that were signed by validated hardware. A different option would be to open the network, and allow anyone to join, but require some other form of rigorous validation to make sure that participating cyclists' nodes actually lived on sensor devices and were actually sending real data. In either case, the smart contract code that accepts and processes requests for financial rewards needs an improved validation mechanism. Rigorous security development was out of scope for this project; however, improving the gating mechanism for distributing financial rewards would need to be the focus of thoughtful design before a full network deployment.

- **Develop data marketplace:** in the current implementation of this thesis, cyclists are rewarded simply for their cycling activity, which they prove through collecting GPS data and reporting the distances they've travelled. While this system is self-sustaining in that participation incentives are provided to both cyclists and organizations, the program could provide even stronger incentives with the introduction of diverse data streams. That is, in a future implementation, we would like to equip bicycles with more sensors—air quality, road quality, safety—that could provide a richer, more detailed picture of a cityscape. This data would already be geolocated, because the bicycles are carrying GPS receivers, and cyclists could use the existing blockchain network infrastructure to distribute and monetize the data they collect.

We believe this type of geolocated, detailed data set would be extremely valuable to a wide range of organizations who would be willing to join the incentives system, and use the platform to gain access to these data streams. These organizations would not only incentivize cycling activity, as they do in the current formulation of the system, but also the collection of these data streams. In this view, this thesis work is not just a financial incentives system, but a data platform that serves to improve urban knowledge, and cultivate a community around sustainable transportation.

Appendix A

Installing an Ethereum Client & Running a Private Ethereum Testnet

The following Appendix covers in detail how to install the *geth* Ethereum client on a laptop and on the Raspbian embedded operating system (Section A.1) and how to start a private Ethereum test network on multiple devices from the command line (Section A.2). Due to Ethereum's rapidly evolving development environment, these instructions may be outdated; however, they are accurate as of May 2017.

A.1 Installing an Ethereum Client

There are many different Ethereum clients, which each implement the Ethereum protocol by providing an Ethereum Virtual Machine that can support a network node and run the code in smart contracts. The Ethereum documentation provides the following reflection on the diversity of the current development environment:

From the earliest days of the project there have been multiple client implementations across a range of different operating systems. That client diversity is a huge win for the ecosystem as a whole. It lets us verify that the protocol...is unambiguous. It keeps the door open for new innovation. It keeps us all honest. However, it can be very confusing for end-users, because there is no universal "Ethereum Installer" for them to use¹.

Based on the recommendations of the Ethereum documentation, and the availability of client-specific support material, we chose to use the *geth* Ethereum client on both our desktop and Raspberry Pi platforms. *Geth* is an Ethereum client written in the Go programming language that provides a JavaScript Runtime Environment for interacting with the Ethereum network, that can be managed interactively, through the command line, or non-interactively, by loading and running JavaScript scripts.

¹"Ethereum Homestead Documentation." Ethereum Homestead 0.1 documentation. Ethereum, n.d. Web. 29 Mar. 2017.

A.1.1 Desktop Installation

Geth is easy to install on all desktop platforms. Installers for stable versions of geth for Linux, MacOS, and Windows are available for download on the geth website at <https://geth.ethereum.org/downloads/>². To install, simply unzip the downloaded installation file and make sure the included binary is available on the machine's system path.

A.1.2 Raspberry Pi Installation

Ethereum client installation on an embedded system (like Raspberry Pi) is much less common than desktop installation, and as such, we found conflicting documentation on the best, most up-to-date way to install an Ethereum client. Ultimately, we were able to use one of the precompiled binaries from the geth website to install geth on the Raspberry Pi in much the same way we installed it in the desktop setting². We used the `uname -a` command to check the Raspberry Pi's chip architecture (e.g. ARMv6 or ARMv7), and downloaded the geth binary appropriate for that ARM processor. Then, we were able to copy this binary to the Raspberry Pi, move the geth binary to the device's system path, and run the client using the `./geth` command on the command line³.

A.2 Running a Private Ethereum Testnet

Ethereum testnets are a simulation of the Ethereum network that allow developers to develop, test, and debug their smart contracts and system processes without having the mine real ether or pay transaction costs. Ethereum provides a public testnet called *Ropsten*, but it is also possible to set up a private testnet to ensure exclusive access to the code and data running on the testnet. A private testnet is also faster and more lightweight because it does not require downloading the entire transaction history of the public testnet. The following process was used to set up a private Ethereum testnet across multiple devices. This procedure was informed by several Internet sources, and in particular by tutorials on the LaBlockchain blog and Stack Exchange^{4,5}.

A.2.1 Genesis Block & Starting First Node

- Create folder structure of choice on your local machine. For this thesis, testnet materials were stored in `~/code/ethereum/testnet`, and each respective node

²Go Ethereum Downloads. N.p., n.d. Web. 10 May 2017.

³"Installation Instructions for ARM." Ethereum/go-ethereum. GitHub, n.d. Web. 10 May 2017.

⁴[EN] Create your private Ethereum blockchain." LaBlockchain. N.p., 31 May 2016. Web. 10 May 2017.

⁵"How to connect to geth from two local machines ethereum." Ethereum beta. Stack Exchange, 12 July 2016. Web. 10 May 2017.

had an associated account with its own data directory labelled by number, e.g. `~/code/ethereum/node1`.

- Create a genesis block for the testnet, such as the sample block in Example A.1, which was used in our implementation, and save it on your machine. The "config" and "alloc" arrays do not need to hold content, but they must be included in the genesis block for initialization to run smoothly. Also note that the "parentHash" and "mixhash" parameters must be 64 bytes long.

```
{
  "config": {},
  "nonce": "0x00000000000000052",
  "timestamp": "0x00",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData": "0x00",
  "gasLimit": "0x8000000",
  "difficulty": "0x40",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x3333333333333333333333333333333333333333333333333333333333333333",
  "alloc": {}
}
```

Example A.1: Custom Genesis Block for Ethereum Private Testnet

- Create an account that will be associated with your first node using the command in Example A.2. Where appropriate, replace command line arguments with appropriate values. Arguments you may want to customize include:
 - networkid: choose a value >100 so that your network doesn't collide with existing networks
 - identity: the name of the account that will be running on top of the node
 - datadir: specifies where the node's local copy of the blockchain will live on your machine

After running the command, you will be prompted to create a password for the account. Be sure to remember this password as it is nonrecoverable and your only way of interacting with this account and accessing the ether inside it.

```
geth --networkid 4507 --identity node1 --verbosity 3 --nodiscover
--nat none --datadir=~/code/ethereum/node1 account new
```

Example A.2: Create an Account to Interact with the Ethereum Network

- Initialize the node, as shown in Example A.3, passing in the account name as the identity you will use to interact with the node. Where appropriate, replace command line arguments with your own values.

```
geth --networkid 4507 --identity node1 --verbosity 3 --nodiscover
--nat none --datadir=~/.code/ethereum/node1
init ~/.code/ethereum/testnet/genesis-block.json
```

Example A.3: Initializing a Private Ethereum Blockchain with a Custom Genesis Block

- Start a geth node running on your computer using the command shown in Example A.4. Where appropriate, replace command line arguments with your own values. Note that all nodes running on the same device must have different `--port` and `--rpcport` arguments. Not all of these arguments are necessary; the `ipc` and `rpc` arguments control how other processes are able to access this node. This was the configuration we used to be able to test and run our complete system, which included remote process communication between geth and the Truffle framework.

```
geth --networkid 4507 --identity node1 --verbosity 3 --nodiscover
--nat none --datadir=~/.code/ethereum/node1 --port 35353
--rpc --rpcaddr 127.0.0.1 --rpcport 8545 --rpcapi "eth,net,web3,personal"
--ipcpath ~/.code/ethereum/node1/geth.ipc --rpccorsdomain "*"
```

Example A.4: Running a Geth Node

- User the `attach` command, as shown in Example A.5, to attach to the running geth process and open an interactive console with the geth node. Note that this functionality can also be achieved by running the `geth console` command; however, separating the console and the node's logging makes testing more convenient.

```
geth attach ipc:~/.code/ethereum/node1/geth.ipc
```

Example A.5: Attaching to a Running Geth Node

- Once you are in the interactive console, you can interact with the node using the command line. For example, to start or stop mining Ether, use the commands `miner.start()` and `miner.stop()`. The `web3.js` module, which is a JavaScript API that facilitates inter-process communication with Ethereum nodes, is available in the geth console⁶. For example, `web3.eth.accounts` will list all the Ethereum accounts associated with your Ethereum node.
- If you have the Ethereum Mist Wallet installed, you can connect the wallet to your running geth instance by running it with the `-rpc` path of your geth process, as shown in Example A.6. The Ethereum Mist Wallet is a graphical interface for the Ethereum blockchain.

⁶"Web3.js." Ethereum/web3.js. GitHub, n.d. Web. 10 May 2017.

```
./Ethereum\ Wallet.app/Contents/MacOS/Ethereum\ Wallet
--rpc ~/code/ethereum/node1/geth.ipc
```

Example A.6: Running the Ethereum Mist Wallet on Top of a Running Geth Node

- Leave this `geth` process running while you start a second node in a new terminal.

A.2.2 Starting Second Node & Adding Network Peers

- Follow the instructions in Section A.2.1 to create an account for a second node, initialize the blockchain in that node's data directory, start a running `geth` process, and connect to that process using the `attach` command. Note that the genesis block must *exactly* match the genesis block used to initialize the first node. The `--networkid` parameter also needs to match the one used with the first node. If you are running this node on the same device as the first node, the new node **must** have unique values for *data directory*, *port*, and *rpcport*. Example A.7 shows the commands we used to initialize and run a second node.

```
geth --networkid 4507 --identity node2 --verbosity 3 --nodiscover
--nat none --datadir=~/.code/ethereum/node2 account new

geth --networkid 4507 --identity node2 --verbosity 3
--datadir=~/.code/ethereum/node2
init ~/.code/ethereum/testnet/genesis-block.json

geth --networkid 4507 --identity node2 --port 30802 --rpcport 8902
--verbosity 3 --nodiscover --nat none
--datadir=~/.code/ethereum/node2

geth attach ipc:~/.code/ethereum/node2/geth.ipc
```

Example A.7: Starting a Second Geth Instance

- Return to the `geth` console of the first node and run `admin.nodeInfo.enode` to get the address of that node. The result should look like the following:
`enode://1eb1fc28ee71.....30c67b10ed071cc@[::]:35353`
- In the console of the second node, run `admin.addPeer()`, passing the enode address of the first node in the command as an argument. For example:
`admin.addPeer("enode://1eb1fc28ee71.....ed071cc@[::]:35353")`
- If you are running the second node on a separate device, you must replace the `[::]` in the address string with the IP address of the first device. For example:
`admin.addPeer("enode://1eb1fc28.....ed071cc@18.85.24.145:35353")`

- It may take a few moments to connect to the peer node, especially if the secondary node has a slow network connection, as in the case of the Raspberry Pi. You can run `admin.peers` or `net.peerCount` in the `geth` console of either node to see if the second node has successfully been connected to the other node in the network.

Complete versions of all referenced code can be found in the thesis GitHub repository: <https://github.mit.edu/cjaffe/bikesblockchain>

Appendix B

Smart Contracts

The following Appendix covers our implementation of the smart contracts that enforce and carry out the logic of interactions on the blockchain. This Appendix focuses on the code used to build and manage these contracts, while a discussion of the contracts' interactions and high-level functionality is presented in Section 5.3.

B.1 Contract Behavior

Smart contracts are pieces of code that live at a specific address on the blockchain. They hold pieces of executable code that can manage interactions between users of the blockchain and store state information. The ability to program and deploy logic on the Ethereum blockchain is one of the major features that distinguishes the Ethereum blockchain from the Bitcoin blockchain, and has led to its use in a wide variety of applications.

For purposes of this thesis work, we designed and implemented three contract types to represent different entities in our system. These contract types were the:

Sensor contract: represents an individual sensor device in the system, and manages the location and distance information that comes from that device. The contract is linked to a specific user contract and passes data to that contract when it receives updates from the device.

User contract: represents a cyclist user of the system, and keeps track of the distance the cyclist has pedaled, in addition to the financial rewards they've earned. The user contract also keeps track of the organizations who participate in the system that may send the user a financial reward.

Organization contract: represents an organization that has agreed to sponsor cycling activity, and keeps track of the users in the system to whom they may periodically send a reward.

An instance of the appropriate contract is instantiated for each entity in the system, and linked to related contracts on the network. These contract instances are created and linked upon initialization of the blockchain; the creation process is covered in further detail in Section B.3.3.

B.2 Truffle Development Framework

For the Ethereum blockchain, smart contracts are typically written in the Solidity language, a "contract-oriented, high-level language whose syntax is similar to that of JavaScript and it is designed to target the Ethereum Virtual Machine"¹. We used the Truffle software framework to develop and test our contracts; Truffle is a software framework that, among other features, handles contract compilation and deployment, facilitates testing, and exposes an interactive console for development². Truffle abstracts many of the low-level configuration tasks necessary for testing and deploying smart contracts on the blockchain, which facilitates fast development.

After installing Truffle via the node.js package manager (npm), Truffle exposes an initialization command that creates a project file structure with folders for contracts, compiled files, tests, and a frontend web application. Truffle also provides an interactive console that allows you to easily interact with the contracts you are currently testing. Example B.1 shows a typical interaction cycle with Truffle, from installation through deployment.

```
# Install truffle
npm install -g truffle

# Initialize truffle project with web application component
truffle init webpack

# Compile contracts and migrate them to the connected Ethereum network
truffle compile
truffle migrate

# Open the interactive truffle console
truffle console
```

Example B.1: Installing and Interacting with the Truffle Development Framework

Contract deployment and many contract transactions must be run on top of an active Ethereum network with active miners who are adding blocks to the blockchain database and supplying Ether to fuel the contract transactions. The Truffle documentation suggests using the testrpc Ethereum client for contract development and testing. The testrpc client simulates an Ethereum network and allows for free and rapid testing. Using settings in Truffle's `truffle.js` configuration file, it is also possible to deploy the Truffle contracts to different networks so that you can test your Truffle contracts on top of alternate Ethereum clients.

¹"Solidity." Solidity 0.4.12 documentation. N.p., n.d. Web. 10 May 2017.

²"Truffle Framework." Truffle Framework. N.p., n.d. Web. 01 May 2017.

B.3 Contract Implementation

B.3.1 Writing Contracts

Smart contracts are similar in structure to an object class. The contract is a "collection of code...and data that resides at a specific address on the Ethereum blockchain"¹. The code of our Sensor contract is presented in Example B.3. The Sensor contract keeps track of the lifetime distance it's recorded, its corresponding instance of the User contract, and the address of that User contract instance. The User and Organization contracts, which can be found in this project's GitHub repository, have similar structure³. The relationship between the contracts and the contract's data fields are illustrated in Figure B-1 (which is also presented in Section 5.3). Additionally, Figure B-2 graphically demonstrates the function call flow used to send data from the sensor to the Sensor contract, the User contract, and the rest of the blockchain network.

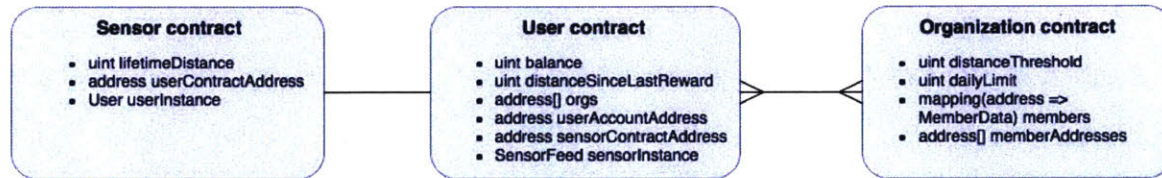


Figure B-1: Contract Relationship Diagram

Ethereum accounts and contracts interact with each other by sending transactions to the address of the account or contract on the blockchain. Accounts can only send and receive transactions related to the transfer of Ether. Contracts, however, can send and receive transactions with many different purposes, using syntax similar to function calls. For example, when the Sensor contract wants to pass distance data to the User contract, it calls a function on its local copy of the User contract's address using the following line of code: `userInstance.updateDistanceSinceLastReward(d)`. Though accounts cannot directly send transactions, it is possible to interact with contracts from an Ethereum node; function calls on contracts can be made from the JavaScript runtime environment provided by Ethereum clients. For example, when the Ethereum node running on the Raspberry Pi wants to send new data to the blockchain network, it uses the following line of code in Example B.2.

```
sensorInstance.updateDistanceAndSendToUser(d, {from: web3.eth.accounts[0]})
```

Example B.2: Sending a Transaction with Data to the Sensor Contract

The transaction call in Example B.2 also indicates which account will supply the Ether to fuel this transaction using the "{from: account}" syntax. Transactions that modify state on the blockchain require a small fee, called "gas", to pay the miners who will eventually mine the transaction. When a function is called from another contract, that contract will automatically pay the gas to fund the transaction. In Example B.2,

³Jaffe, Caroline. "Cjaffe/ bikesblockchain." Github.mit.edu, n.d. Web. 11 May 2017. <<https://github.mit.edu/cjaffe/bikesblockchain>>.

which is coming from an Ethereum node instead of a contract, the gas source must be explicitly specified. Interactions between the Raspberry Pi's Ethereum node and the system's contracts are discussed in more detail in Appendix C.

```
pragma solidity ^0.4.4;

import "./User.sol";

contract SensorFeed {
    uint lifetimeDistance;
    address userContractAddress;
    User userInstance;

    function() payable { }

    /** GETTER METHODS **/
    function getLifetimeDistance() constant returns (uint d) {
        return lifetimeDistance;
    }

    function getUserContractAddress() constant returns (address u) {
        return userContractAddress;
    }

    function getUserInstance() constant returns (User u) {
        return userInstance;
    }

    /** SETTER AND OPERATION METHODS **/
    function setUserAddressAndInstance(address addr) {
        userContractAddress = addr;
        userInstance = User(userContractAddress);
    }

    function updateDistanceAndSendToUser(uint d) {
        lifetimeDistance += d;
        sendDistanceToUserContract(d);
    }

    function sendDistanceToUserContract(uint d) {
        userInstance.updateDistanceSinceLastReward(d);
    }
}
```

Example B.3: SensorFeed Contract

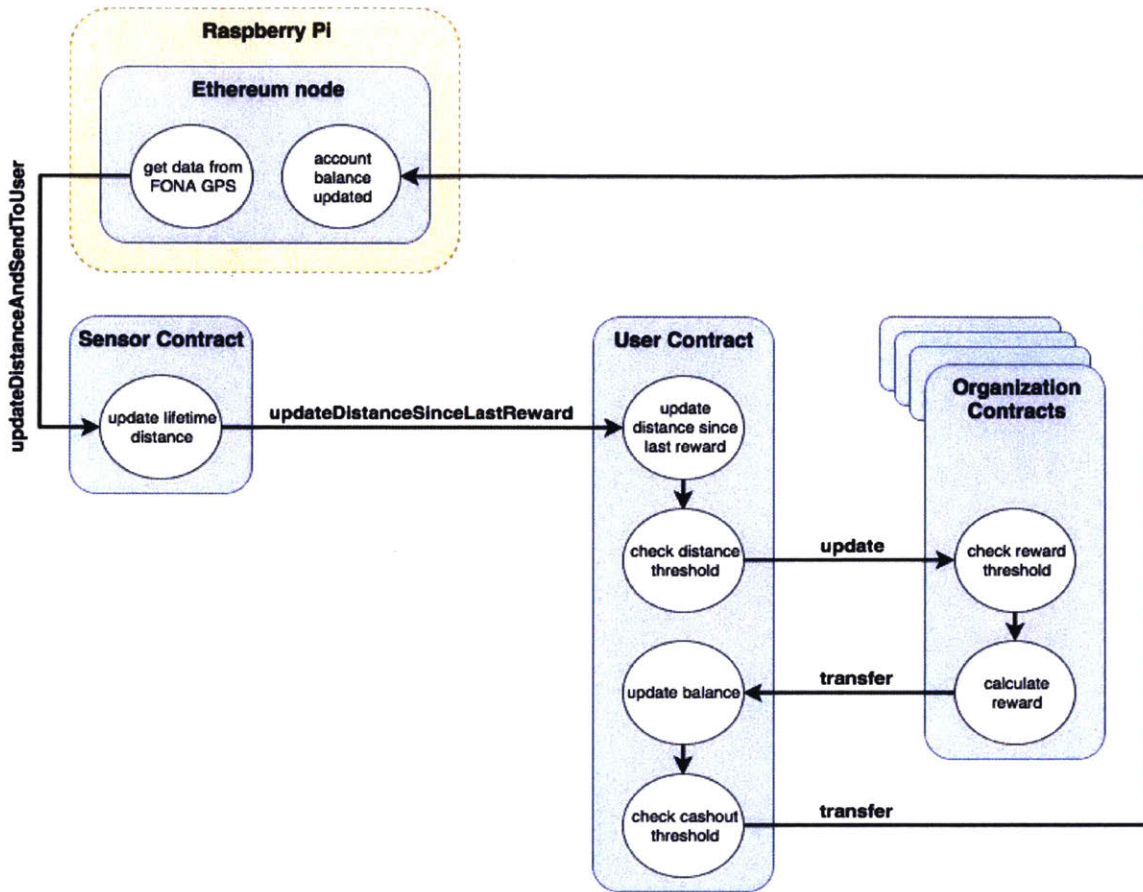


Figure B-2: High-Level Contract Interaction Flow for Sending Distance Data

B.3.2 Testing Contracts

To verify the behavior of our contracts, we developed a multi-tiered testing procedure. First, we used the Truffle framework to develop unit tests, which can be written in Solidity or JavaScript. Our unit tests were written in JavaScript and used the Mocha and Chai testing frameworks which streamline the tests' format; Example B.4 shows two unit tests which verify that a User contract can add an Organization contract to its organizations array^{4,5}.

After unit testing in Truffle, we used the Browser Solidity interface to test interaction flows between contracts⁶. The browser-based graphical interface allows you to compile contracts, create test instances of contracts, and manually experiment with transaction calls. Browser Solidity also provides a step-by-step debugging tool. An example of the testing interface is shown in Figure B-3.

Finally, we tested our contracts and contract interactions on a live test network by using the `truffle migrate` command to deploy certain contracts to the network, and writing JavaScript code that would create other contracts and send them trans-

⁴"Mocha." Mochajs.org. N.p., n.d. Web. 11 May 2017.

⁵"Chai Assertion Library." Chai. N.p., n.d. Web. 11 May 2017.

⁶"Browser Solidity." Remix - Solidity IDE. N.p., n.d. Web. 11 May 2017.

actions. These scripts were executed in geth's JavaScript runtime environment and are discussed in great detail in Section B.3.3.

```
it("add organization", function() {
    return user.addOrganization.sendTransaction(org.address);
});
it("get organizations list", function() {
    return user.getOrgs.call().then(function(organizations) {
        organizations = [org.address];
        assert.include(organizations, org.address,
            "Organizations list includes the organization just added");
        assert.lengthOf(organizations, 1, "Only one organization in list");
    });
});
```

Example B.4: JavaScript Unit Tests in Truffle



Figure B-3: Browser Solidity Testing Interface (Remix Solidity, <https://ethereum.github.io/browser-solidity>)

B.3.3 Creating & Deploying Contracts

After developing and testing contracts, we had to devise a way to actually instantiate contracts in a real deployment setting. There are several ways to create contracts; indeed, the Truffle framework command line can be used to create instances of contracts. However, we wanted to be able to create a different instance of each contract

for each new Ethereum node (e.g. cyclists and organizations) that joined the network. We also wanted to be able to create those contract instances at different times, and link them with different Ethereum accounts.

To provide this flexibility, we wrote a contract creation script in JavaScript that runs in an Ethereum node's JavaScript runtime environment. The contract creation code, shown in Example B.5 and edited for clarity, is part of the `startup.js` script and runs when a new Raspberry Pi sensor device is initialized. The orchestration of the Raspberry Pi initialization is discussed in further detail in Appendix C.

In the script, we first create a generic "Sensor Contract object" using the contract's Application Binary Interface (ABI), which describes the functions defined on the contract's interface. Then, we use the `new()` command to create a new instance of that object, with the following parameters:

from: the account that supplies a small amount of Ether (the "gas") to pay for the cost of mining the contract creation transaction

data: the contract bytecode, which is a compiled representation of the contract

gas: the maximum amount of gas the "sending" account (in this case, the Ethereum account on the new sensor device) is willing to pay for the contract creation transaction to be mined

The `new()` command does not immediately return the new contract. Instead, it returns a JavaScript Promise, which lets the program retrieve the results of the asynchronous contract creation command later in the program by exposing the transaction's hash address. Later in the code's execution, after being certain that the transaction has been mined, we retrieve the address of the new Sensor contract.

Complete versions of all referenced code can be found in the thesis GitHub repository: https://github.mit.edu/cjaffe/bikesblockchain
--

```

var sensorabi = [
  {
    "constant": true,
    "inputs": [],
    "name": "getUserInstance",
    "outputs": [
      {
        "name": "u",
        "type": "address"
      }
    ],
    "payable": false,
    "type": "function"
  },
  ...
  {
    "constant": true,
    "inputs": [],
    "name": "getUserContractAddress",
    "outputs": [
      {
        "name": "u",
        "type": "address"
      }
    ],
    "payable": false,
    "type": "function"
  },
  {
    "payable": true,
    "type": "fallback"
  }
];

var sensorfeedContract = web3.eth.contract(sensorabi);
var sensorPromise = sensorfeedContract.new(
  {
    from: web3.eth.accounts[0],
    data: '0x6060604052341561000c57fe5b5b6103b38061001c60...8852265bb10029',
    gas: '4700000'
  });

receipt = web3.eth.getTransactionReceipt(sensorPromise.transactionHash);
sensorAddress = receipt.contractAddress;

```

Example B.5: Sensor Contract Creation in JavaScript

Appendix C

Raspberry Pi Configuration & Usage

The following Appendix covers our use of the Raspberry Pi single-board computer for this thesis project. The documentation covers installation and set-up, and the procedures we used to initialize an Ethereum node on the Raspberry Pi and collect and transmit GPS data.

C.1 Raspberry Pi Single-Board Computer

The Raspberry Pi is a low-cost, single-board computer that has been popular among engineers, teachers, and hobbyists alike. Raspberry Pi computers are under active development, and there have been several generations of the device. For this thesis project, we used the Raspberry Pi 2 Model B V1.1, pictured in Figure C-1, which features four USB ports, a quad-core ARMv7 processor, and 1G of RAM¹. The Raspberry Pi runs at 5V with a recommended current supply of 2A.

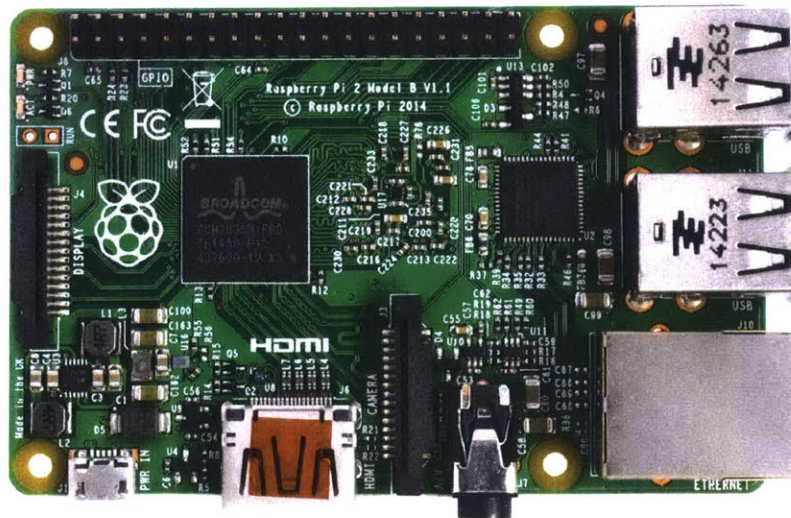


Figure C-1: Raspberry Pi 2 Model B V1.1 (<https://commons.wikimedia.org>)

¹"Raspberry Pi 2 Model B." Raspberry Pi. N.p., n.d. Web. 11 May 2017.

C.2 Installation & Preparation

The Raspberry Pi runs the Raspian operating system, an embedded version of Debian that is optimized for the Raspberry Pi's hardware. In order to use the Raspberry Pi for the system proposed by this thesis, we had to change some of the Raspberry Pi's default configurations, install several software libraries, and complete several additional set-up steps. These procedure are described in detail in this section.

C.2.1 Using the Raspberry Pi with the Adafruit FONA

In order to use the Adafruit FONA GPS/GSM module with the Raspberry Pi, we had to go through several set-up steps. This procedure is adapted from a tutorial on the Adafruit website². The process is as follows:

- Using the Raspberry Pi's configuration interface, disable the Raspberry Pi kernel's use of the board's hardware serial connection, so that the FONA could communicate on that channel instead. Type `sudo raspi-config`, navigate to "Interfacing Options", and disable the hardware serial connection. Reboot the Raspberry Pi if necessary.
- Make sure the Raspberry Pi's UART pins are configured for serial communication by making sure the `/boot/config.txt` file contains the line: `enable_uart=1`. If `enable_uart` is set to 0, you may need to change it to 1, and reboot the Raspberry Pi.
- Wire the Raspberry Pi's GPIO UART pins to the Adafruit's UART pins, as shown in Figure C-2. The TX pin on the Raspberry Pi should connect to the RX pin on the FONA, and vice versa. You should also connect the Raspberry Pi's 3.3V reference to the FONA's V_{io} pin, and connect a ground reference between the two.
- Download a point-to-point protocol software library using `sudo apt-get install ppp`. Then, in the `/etc/ppp/peers` directory, download Adafruit's FONA configuration file from:

`https://raw.githubusercontent.com/adafruit/FONA_PPP/master/fona`

Configure the file with your GSM network's APN identifier. In our case, the APN value was "internet". You should now be able to connect to the internet over the FONA's GPRS connection. Use `sudo pon fona` and `sudo poff fona` to toggle this connection.

- Download a serial interface, such as screen, with the `sudo apt-get install screen` command. Use `screen /dev/ttyAMA0 115200` to test the serial connection to the Adafruit FONA: `/dev/ttyAMA0` is the serial port connection on the Raspberry Pi and 115200 is the baud rate.

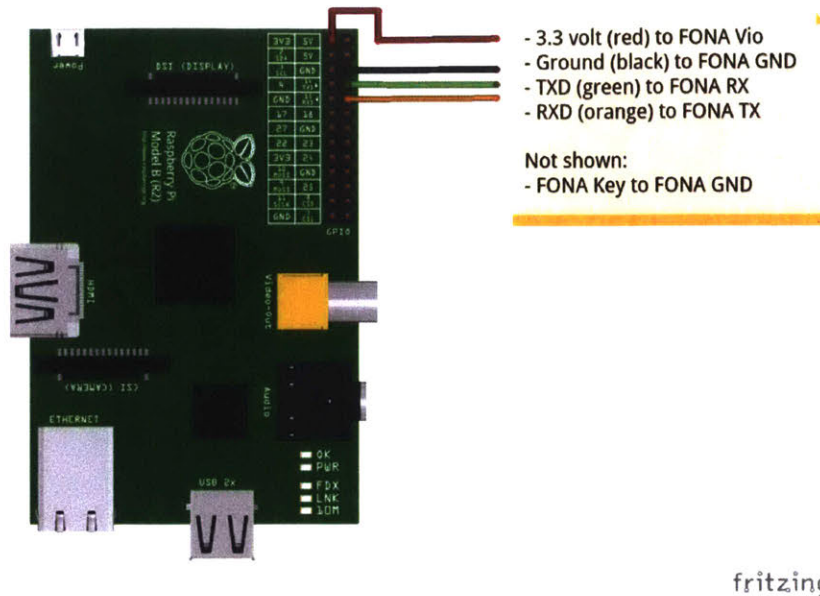


Figure C-2: Wiring Connections Between the Raspberry Pi and the Adafruit FONA (<https://learn.adafruit.com/assets/17881>)

The above instructions facilitated our use of the Adafruit FONA as a communication channel. To collect GPS data with the FONA, we wrote a Python script that communicated with the FONA using the AT command set over a serial port. For this script, we installed the following Python packages using the `pip install <package name>` command:

serial: manages access to and use of the device's serial port

geopy: geocoding library that provides a great-circle distance calculation

pytz: provides accurate and cross-platform timezone calculations

C.2.2 Using the Raspberry Pi as an Ethereum Node

To use the Raspberry Pi as an Ethereum node, we had to select and install an Ethereum client. Based on the availability of documentation and support for embedded chip architectures, we chose to use `geth`, for which detailed installation instructions are covered in Section A.1.

C.3 Ethereum Node Initialization Scripts

In order for the Raspberry Pi-based sensor device to serve as a node in our Ethereum network, it needs to run a persistent Ethereum client that will allow it to communicate and synchronize with the rest of the network. We wanted to implement an

²"FONA Tethering to Raspberry Pi or BeagleBone Black." Setup | FONA Tethering to Raspberry Pi or BeagleBone Black | Adafruit Learning System. Adafruit, n.d. Web. 04 May 2017.

initialization procedure for this client that would be seamless for a user, and also provide convenient testing functionality for us as developers. To accomplish this aim, we wrote a bash initialization script, `startup.sh`, that runs every time the Raspberry Pi turns on, or "boots up". We scheduled the script using the cron scheduling functionality on the Raspberry Pi. The cron scheduler can be edited using the `crontab -e` command; our script is scheduled using the following line in the crontab file:

```
@reboot /code/scripts/startup.sh.
```

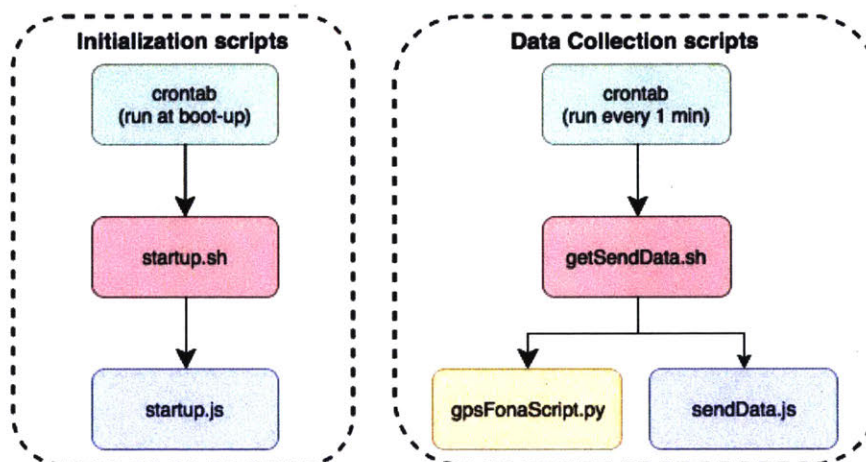


Figure C-3: Raspberry Pi Script Relationship Diagram

Figure C-3 visualizes the relationships between the different scripts running on the Raspberry Pi. The initialization bash script, called `startup.sh`, is shown in Example C.1 and lightly edited for clarity. This script executes our initialization procedure the very first time the Raspberry Pi sensor device is powered on, and also on subsequent instances (e.g. if the Raspberry Pi loses power and needs to restart). For the purposes of our current proof-of-concept deployment, we assume that as manufacturers and system developers, we have been able to hard-code some configuration parameters that will allow the new node to connect to the rest of the network. While a future implementation may use a more sophisticated or automated node discovery algorithm, this system was appropriate for our usage.

The script first sets environment variables and creates a configuration files to be used later in the initialization process. The script then checks if a blockchain identity—indicated by the presence of the `keystore` directory—exists on the device. If not, the script creates a blockchain identity and initializes the blockchain, using commands described in Appendix A. Next, the script starts a `geth` light node process running in the background. A light node, which only downloads the blockchain's block headers instead of the full transaction record, alleviates some of the power and storage demands of participating in the Ethereum network. This step happens every time the Raspberry Pi is powered on, even if it's not the first time the sensor is turned on, because we always want a `geth` process running so that the sensor can participate in the Ethereum network.

Finally, this bash script checks whether the contract creation script has been

executed by looking for the `geth.log` file. If not, the script will use bash's here document format to pass data to and execute the `startup.js`.

```
#!/bin/bash

# SET ENVIRONMENT VARIABLES
export NODENAME="node3"
export NETWORKID="4507"
export GETHPORT="35354"

# MAKE PASSWORD FILE
mkdir ~/code/ethereum/$NODENAME
echo $NODENAME > ~/code/ethereum/$NODENAME/password

# MAKE STATIC NODES FILE
echo '["enode://aee41a221f...c1b8c2c0d5838aa018.85.24.145:35353"]' >
    ~/code/ethereum/$NODENAME/static-nodes.json

# GET ORG ADDR
ORGADDR=$(cat ~/code/scripts/orgaddress.txt)

# MOVE TO CORRECT DIRECTORY
cd ~/code

# TURN GPRS ON
sudo pon fona

# CHECK IF BLOCKCHAIN HAS BEEN INITIALIZED YET AND INITIALIZE IF IT HASN'T
if [ ! -d "~/code/ethereum/$NODENAME/keystore" ]; then
    # CREATE GETH IDENTITY
    ./scripts/geth --fast --networkid $NETWORKID --identity $NODENAME
        --verbosity 3 --nodiscover --nat none
        --datadir=~/code/ethereum/$NODENAME
        --password ~/code/ethereum/$NODENAME/password account new

    # INIT BLOCKCHAIN
    ./scripts/geth --networkid $NETWORKID --identity $NODENAME --verbosity 3
        --datadir=~/code/ethereum/$NODENAME
    init ~/code/ethereum/testnet/genesis-block.json
fi

# START GETH NODE IN BACKGROUND
./scripts/geth --networkid $NETWORKID --identity $NODENAME --verbosity 3
    --nodiscover --nat none --datadir=~/code/ethereum/$NODENAME --light
    --port $GETHPORT --ipcpath ~/code/ethereum/$NODENAME/geth.ipc &

while [ ! -e ethereum/$NODENAME/geth.ipc ]
do
```

```

    sleep 2
done

# CHECK IF STARTUP SCRIPT HAS BEEN RUN YET AND RUN IF IT HASN'T
if [ ! -f "geth.log" ]; then
# UNLOCK GETH ACCOUNT, GET ORG CONTRACT ADDR, RUN GETH STARTUP SCRIPT
./scripts/geth attach ipc:../ethereum/$NODENAME/geth.ipc << EOF > geth.log
personal.unlockAccount(web3.eth.accounts[0], $NODENAME, 150000);
orgAddress = "$ORGADDR";
loadScript("ethereum/testnet/scripts/startup.js");
EOF
fi

```

Example C.1: The startup.sh Initialization Script

While the mechanics of smart contract creation are discussed at length in Section B.3.3, we present in Example C.2 the contract initialization steps. In essence, these lines of code link the specific contract instances that have been created for the Ethereum node running this code. That is, the Sensor contract address is added to a data field on the User contract (and vice versa), the User contract address is added to a data field on the Organization contract (and vice versa), and the address of the Ethereum node is added as a data field on the User contract. These actions will allow these contracts to interact with the correct instance of each contract in the future. These contract addresses are also stored locally, on the Raspberry Pi, so that the data distribution script can send data to the correct contract address. These scripts are discussed in detail in the subsequent section.

```

sensorfeedContract.at(sensorAddress).setUserAddressAndInstance(
    userAddress, {from: web3.eth.accounts[0]});
userContract.at(userAddress).setSensorAddressAndInstance(
    sensorAddress, {from: web3.eth.accounts[0]});
orgContract.at(orgAddress).addMember(
    userAddress, true, {from: web3.eth.accounts[0]});
userContract.at(userAddress).addOrganization(
    orgAddress, {from: web3.eth.accounts[0]});
userContract.at(userAddress).setUserAccountAddress(
    web3.eth.accounts[0], {from: web3.eth.accounts[0]});

```

Example C.2: Initializing Contract Connections on Newly Created Contracts

C.4 GPS Data Collection

Beyond providing a network connection, the purpose of the Adafruit FONA module is to provide a mechanism to collect GPS data that can be used to measure and verify cyclists' activity so that they can be rewarded appropriately. In order to balance data accuracy and device battery life, our system collects GPS data one a minute. We

devised a set of scripts, visualized in Figure C-3, to carry out the intended behavior of the system. The main bash script, `getSendData.sh` is run every minute using the cron scheduler. The bash script sets a few environment variables, turns the FONA's GPRS connection off so that the serial port is available for querying for GPS data, and runs the Python script, `gpsFonaScript.py`, that will make those calls.

The main content of `gpsFonaScript.py` is shown in Example C.3. Communication with the Adafruit FONA is achieved using the AT Command set, which is a command language typically used in communication applications³. Each command is prefaced with the "AT" characters, and interpreted by the receiving module. For our purposes, we used five AT commands to establish a connection with the FONA and query for GPS data, which are:

AT: initializes and verifies the connection with the FONA; this command returns an "OK" if the connection is successful

AT+CGNSPWR?: queries the power status of the navigation submodule; this command returns 1 or 0

AT+CGNSPWR=1: sets the power of the navigation submodule to 1 ("on"); this command returns an "OK" if the request was successful

AT+CGNSPWR=0: sets the power of the navigation submodule to 0 ("off"); this command returns an "OK" if the request was successful

AT+CGNSINF: requests location information; if successful, this command returns a timestamped NMEA sentence with location information and metadata

The script establishes communication with the FONA ("AT"), makes sure the navigation submodule is powered on ("AT+CGNSPWR?" and "AT+CGNSPWR=1") and then queries the FONA for a GPS fix and location data ("AT+CGNSINF"). Before the FONA is able to return accurate location information, it must get a GPS fix; that is, it must learn the location of enough GPS satellites to accurately calculate its position. The process of getting a GPS fix can take a few minutes, depending on how recently the GPS receiver has been used, so our script sends multiple, staggered requests for location data. Once the module finds a GPS fix, it is able to start reporting location data. The script parses the data from the FONA module to extract latitude and longitude data points, and uses the `geopy` Python library to calculate the distance between the latest reading and the previous reading. The script then saves these data points, as well as the calculated distance, to a text file.

³AT Commands Reference Guide. N.p.: Telit Wireless Systems, 2006. Pdf.

```

# Connect to FONA
connect = False
while connect == False:
    connect = writeCheckReply("AT\n", "OK")

# Turn GNS power on
turnFonaOn()

# Check that GNS power is on, and make FIX_ATTEMPTS attempts to find GPS fix
if checkFonaOn():
    GPSfix = 0
    counter = 0
    # Try to get a GPS fix
    while GPSfix == 0 and counter < FIX_ATTEMPTS:
        res = int(checkGPSFix())
        time.sleep(.5)
        if res == 1 or res == 0:
            GPSfix = res
        counter += 1
    # If there's a fix, get data and calculate distance
    if GPSfix == 1:
        latLon = getParseGPSReply()
        distance = getDistanceAndLog(latLon)
        logging.info("LAT_LONG: " + str(latLon[0]) + ',' + str(latLon[1]))
        logging.info("DISTANCE: " + str(distance))
    else:
        logging.info("NO_GPS_FIX")
else:
    logging.info("FONA_CANNOT_COMMUNICATE")

```

Example C.3: Python Script for Collecting GPS Data

When the Python script finishes, execution returns to the `getSendData.sh` bash script. This script turns the FONA's GPRS service back on so that the node can connect to the blockchain network, pulls the new location and distance data from the data log, and instructs the running geth node to execute the `sendData.js` JavaScript script. This script uses the address of the Sensor contract that was created in the the set-up script to send the distance data to the Ethereum network using the code in Example C.4. The subsequent smart contract interaction flow is described in detail in Appendix B.

```

sensor.updateDistanceAndSendToUser(distance, {from: web3.eth.accounts[0]});

```

Example C.4: Python Script for Collecting GPS Data

Complete versions of all referenced code can be found in the thesis GitHub repository: <https://github.mit.edu/cjaffe/bikesblockchain>