BusPlan

ECE 445 Design Document

Scott Liu
sliu125@illinois.edu

Connor Lake
crlake2@illinois.edu

Aashish Kapur
askapur2@illinois.edu

TA: Evan Widloski

October 16, 2018

Contents

I. Introduction

i. Objective

Buses are a form of standard mass transit service and are usually dispatched at fixed frequencies [9]. For example, MTD deploys their buses roughly once every 10 minutes during school hours and once every 30 minutes otherwise [6]. However, during rush hours we have observed buses to be completely packed; while on other occasions we have also found buses to carry only very few passengers. This poor load-balancing results in inefficiencies at multiple stages: losing existing customers due to overwhelmingly full buses, deterring potential customers due to crowded first impressions, operating at extremely light loads, etc.

Herbert argued in his article within the context of urban bus transportation: "Fewer riders lead to less frequent service leads to fewer riders" [13]. Guihaire and Hao also speculate the car drivers might consider switching to public transport if we have a less crowded system [9]. It is therefore imperative that we develop a solution to address this commonplace issue.

Our team proposes the *BusPlan* project: a network of smart detectors that actively survey the amount of people waiting for a bus, gather information, and send this information to a central unit that performs aggregation and analysis. Ultimately, our BusPlan service would be able to find patterns, predict trends, and provide relevant visualizations back to the bus company, who will use this to make strategic business decisions.

ii. Background

Four stages of public transport operations have been identified by Ceder in his book *Public Transport Timetabling and Vehicle Scheduling* [3]:

1. Network route design
2. Setting timetables
3. Scheduling vehicles to trips
4. Assignment of bus crew

These steps together form a well established mode of public transport operation planning, and has already been adopted successfully worldwide. However, our team believes that the heart of the aforementioned problem lies within steps 2 and 3. The innovative network of devices that we propose would disrupt the traditional workflow, and greatly enhance the system's productivity and efficiency. By automatically detecting the amount of passengers waiting at a bug stop frequently and accurately, we would be able to obtain a sequence of numbers taken at successive equally spaced points in time. Furthermore, by analyzing and manipulating this time series data using modern tools, we would be able to achieve powerful forecasts that would replace timetables and scheduling completely. We believe that this would provide significant value to the bus company.

Many theories have already been proposed by experts to find an optimal way to form timetables or schedules [1] [10] [4] [8]. Some propose laying out optimal networks [5] while others focus on forming timetables after the fact [11] with real world case studies. They all offer incredible insight into formulating ideas to tackle this formidable challenge. However, most of them fall short when considering a truly dynamic environment that is subject to frequent change. Even theories that incorporate computer programs [17] [12] have been developed without this particular consideration. This is purely because they can only ever study past, outdated information as opposed to a stream of present, ever-changing information.

The buses that MTD use to provide service are already digitally augmented with STOPwatch technology [15]. It would be completely reasonable to add additional features to further enhance its efficiency. We also recognize the importance of a minimum bus service frequency [2] and will cater to delivering a model which takes this factor into account.

iii. High-Level Requirements

- Nodes must be able to collect an accurate measurement of number of people at bus stops by listening to WiFi probe signals
- Nodes must be able to relay data to a centralized server through other nodes
- We must be able to make predictions with reasonable accuracy and issue bus deployment recommendations that effectively alleviate unbalanced bus loads

II. Design

Each detector has three main parts: a power supply, a control unit, and a WiFi module. The power supply ensures that the system is powered continuously. The WiFi module listens for probe beacons from nearby WiFi stations (phones, laptops, etc). The radio chip connects and relays data from one node to surrounding nodes. The controller unit reads the data from the WiFi chip and intelligently caches and sends it using the radio chip.

The physical design should be a flat and thin box with a large surface area on top for the solar panel, preferably around the size of a MacBook trackpad (4" x 5.5"), with a height approx. equal to a laptop as well. Smaller is better for our device, so we will try and fit the components into as small of a case as possible.

Figure 1 illustrates the basic hardware design.

Figure 2 illustrates the high level design overview.

Figure 3 illustrates the control flow differences between the drive-by node and the bus stop node.

Figure 4 illustrates a more detailed, node-specific hardware architecture.

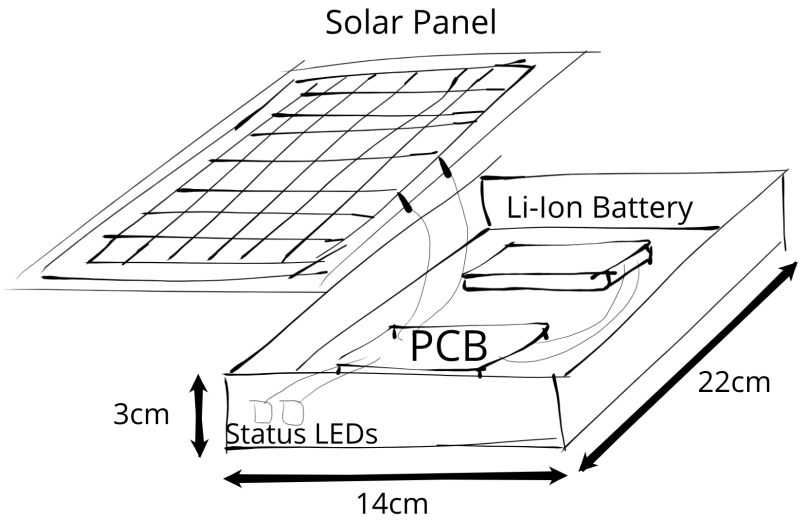

Figure 1: *Basic Hardware Design*

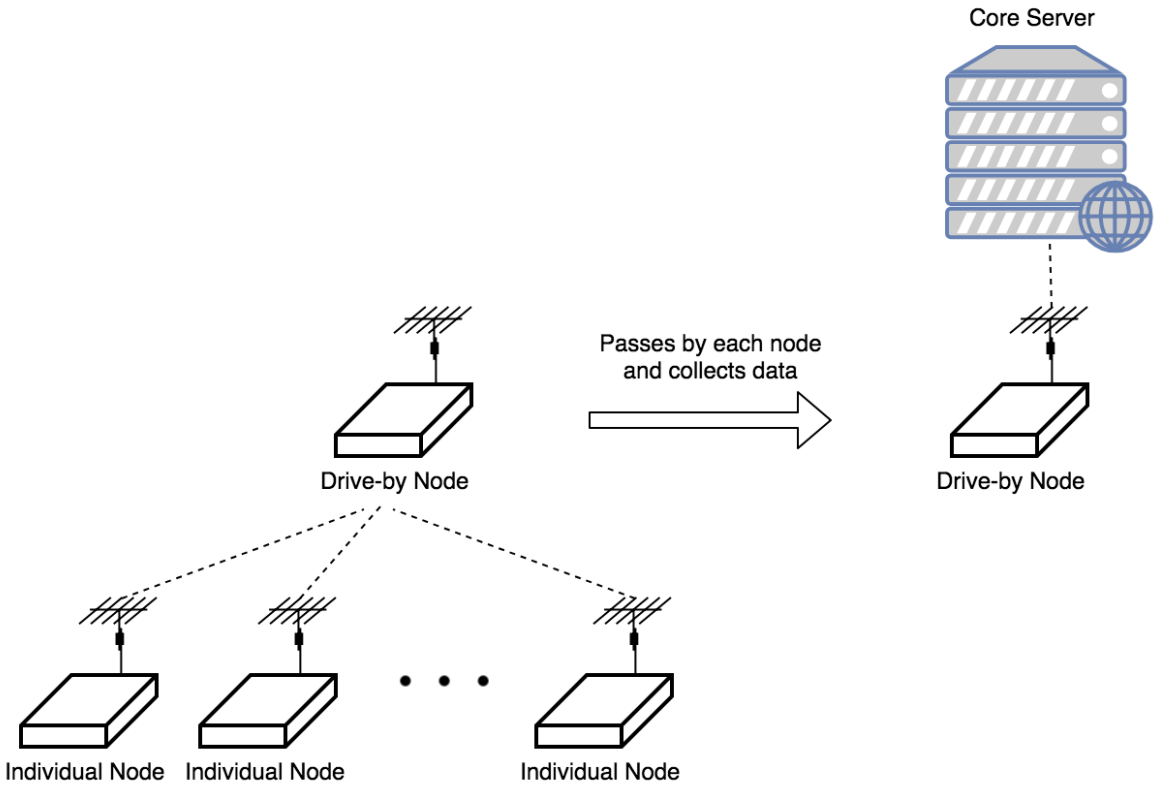

Figure 2: *High Level Overview*

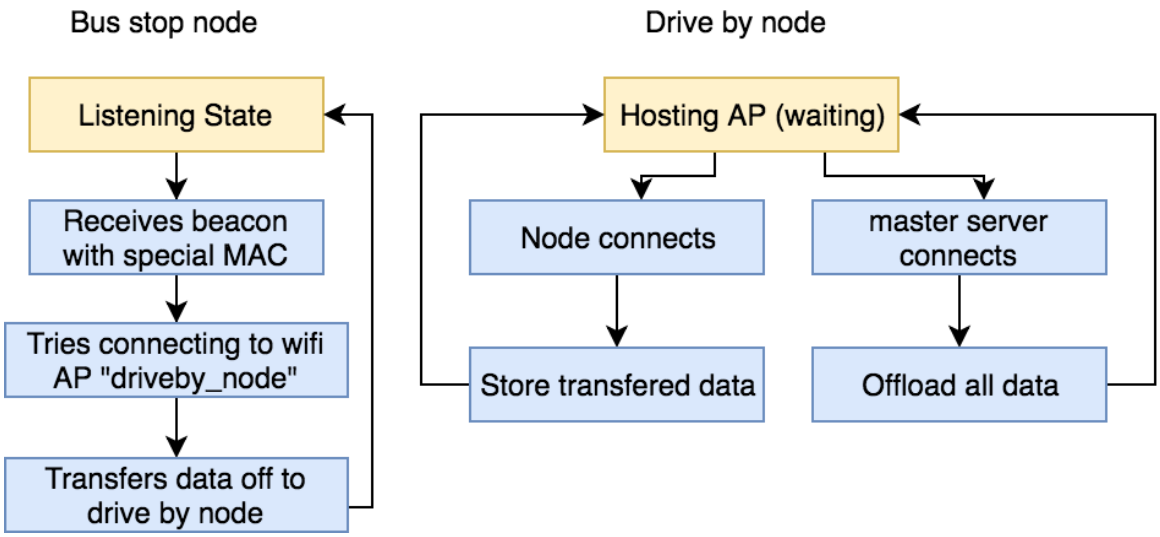

Figure 3: *Node Control Flow*

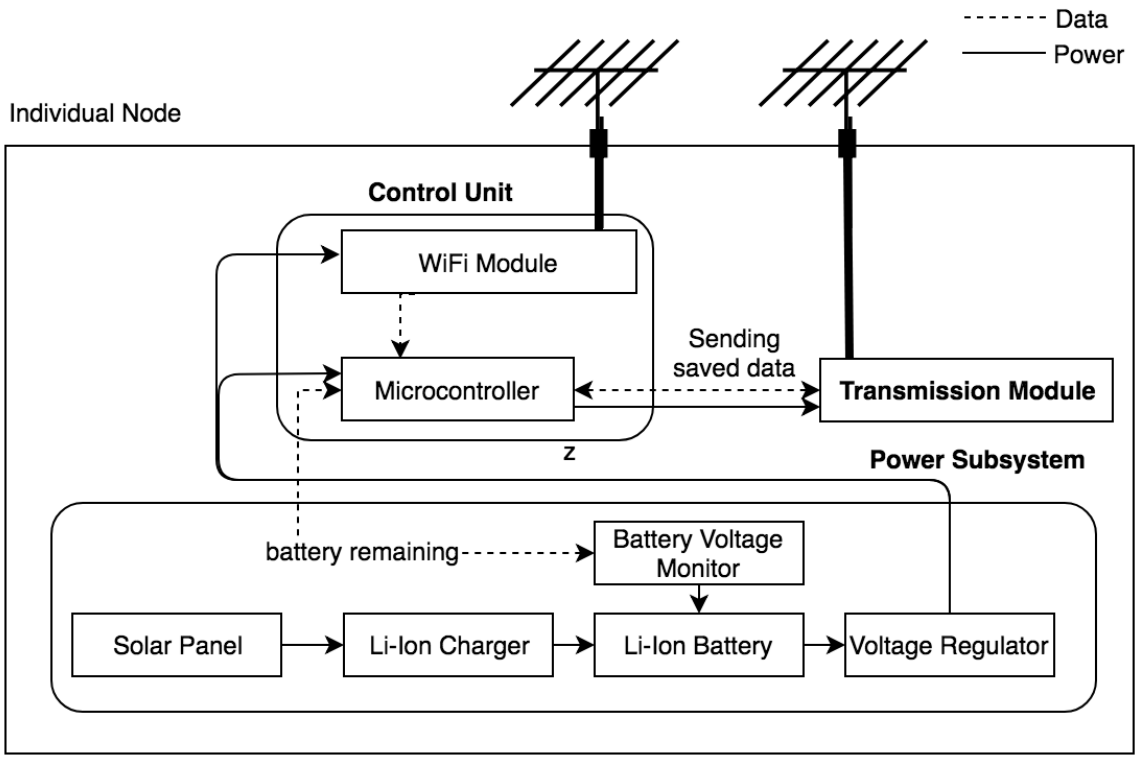

Figure 4: *Node Level Overview*

i. Bus Stop Node

The bus stop node listens for WiFi probe requests, and logs the data. When the drive-by node passes by the bus-stop node, the bus-stop node will send all of the collected data to the drive-by node, which will eventually send it to a central server

i.1 Power Subsystem

This subsystem is responsible for powering the node, and uses a combination of a solar panel and a battery to ensure continuous power.

Requirement	Verification
• The power subsystem should be able to charge the battery enough to cover the daily power consumption of the device over a day, assuming standard weather conditions. • The power subsystem should be able to deliver enough power for one full day, even if it has not recieved any power from its power source	• In a lab environment, expose the power subsystem to 20,000 lux(daylight) for 2 hours. Verify that the battery has been charged by at least 800 mAh. • Expose the power subsystem to one full day's worth of light, which is roughly equal to 2800 mAh. Create a test circuit which draws 175mAh, the estimated hourly power usage of the node, and connect it to the power subsystem. Ensure that the battery powers the test circuit for 16 hours.

Solar Panel Each detector will be powered using a solar panel, which will charge the battery and power the device. Bus service runs 7am to 2am the following day, however the load at the edge hours will be much less, so we may have the scanning frequency lower than normal, saving power. Worst case, the solar panel must provide enough power for 19 hours of use.

Li-Ion Charger The charger takes power from the solar panel and converts it to the correct voltage/amperage for the battery. It makes sure to use constant current or constant voltages stages to charge the battery without charging it too fast.

Requirement	Verification
• Li-ion battery charges to 3.65-3.85V when an input voltage between 3.4-5.5V is applied	• Charge battery with a voltage of 3.4-5.5V, and verify that it fully charges after 8 hours by checking that the STAT pin of the IC chip is high • Verify that the battery does not reach temperatures greater than 125 degrees C(using an IR thermometer) while charging, indicating a fault with a battery

Li-Ion Battery The lithium-ion battery will act as a buffer to aid in power delivery from the solar panel. We do not expect to collect and store enough power to last the device through the night. Instead we will optimize the power usage and size of the battery through empirical power draw testing.

Voltage Regulator This integrated circuit supplies the required 3.3V to the node system. This chip must be able to handle the peak input from the battery (4.2V) at the peak current draw (~300mA).

Requirement	Verification
• The voltage regulator must provide 3.3V $\pm$ 5% from a 3.7V to 4.2V source. • Must maintain thermal stability below 125° C at a peak current draw of 200mA.	• Plug in a power supply to the input of the Voltage regulator, and a multimeter to the output, and verify that, while changing the voltage between 3.7-4.2V, the output is 3.3V $\pm$ 5% • Create a test circuit which draws 200mA, and connect it to the output of the voltage regulator. Connect the input to a power supply, and use a IR thermometer to verify the temperature of the regulator

i.2 Control Subsystem

The control subsystem contains the logic for the node to count the number of WiFi probe requests and communicate it to the central server.

Control Unit We would like to use esp8266 WiFi chip to power the logic of the device. It will be listening for probe requests, keeping track of the history of what we have seen, and ultimately encrypting and sending it off the chip. It will take a snapshot of the mac addresses we have seen every 30 seconds. It will remove mac addresses that we have not seen for more than 60 seconds. We also log the amount of times we have seen the mac address each snapshot. We plan on having the WiFi chip listen for a special probe request that will prompt it to stop listening for probe requests, and connect to a WiFi hotspot hosted by the drive-by node. It will then send off the encrypted data. The data will probably be sent over a raw socket instead of a HTTP request. There is a possible concern for a denial-of-service attack where an attacker will keep broadcasting the special beacon to prevent the device from listening. We plan to mitigate this by having encryption verification on the node to make sure that the drive-by node is the one sending the probe request.

Requirement	Verification
• The WiFi chip must listen for and remember which mac addresses we have seen over time and be accurate to the amount of nearby devices • The node must be able to successful transfer data to the drive-by node	• We will run the device in a room full of people and test to see if the number of mac address detected are accurate to the number of devices actually in the room • We will run a test where the drive-by node will be in range of a node, and test to see if the drive-by node can successfully gather the data from the node

Visual Feedback Our device would have several status LEDs to indicate the overall status of the device (e.g. if it has successfully been powered on). It would also have two 7 segment displays that shows the current number of nearby WiFi devices.

Requirement	Verification
The LEDs should display the current power state of the board	Once we plug in the board, the power LED should light up

Data Storage Our device needs to store temporary data before it is transmitted to our central server. The esp8266 has built in flash memory that we will use to store the detected MAC addresses and their timestamps.

Requirement	Verification
The device should cache the data of seen mac addresses locally.	After a certain amount of time we will request data from the device and confirm that it successfully kept track of all the data it was logging, and sends it all.

ii. Drive By Node

The drive-by node is located on a bus or other vehicle which drives between bus stops, and collects WiFi probe data from each of the bus-stop nodes located on its route. Once it has collected this data, it will connect to a base station node and communicate the WiFi probe data to the central server.

ii.1 Power Subsystem

This subsystem is responsible for powering the node, and ensuring that it stays on despite any interruptions from the power source.

Power Source Each drive by node will be connected to a power source located within the vehicle, which will be able to provide X amount of power, enough for the node to operate. However, there is the chance that this source could get interrupted, which is why the power subsystem will contain a small battery to smooth out the disruptions.

Li-Ion Charger The charger takes power from the source and converts it to the correct voltage/amperage for the battery. It makes sure to use constant current or constant voltages stages to charge the battery without charging it too fast.

Requirement	Verification
• Li-ion battery charges to 3.65-3.85V when an input voltage between 3.4-5.5V is applied	• Charge battery with a voltage of 3.4-5.5V, and verify that it fully charges after 8 hours by checking that the STAT pin of the BQ2416 chip is high • Verify that the battery does not reach temperatures greater than 125 degrees C(using an IR thermometer) while charging, indicating a fault with a battery

Li-Ion Battery The lithium-ion battery will act as a buffer to aid in power delivery. We do not expect to collect and store enough power to last the device through the night. Instead we will optimize the power usage and size of the battery through empirical power draw testing.

Voltage Regulator This integrated circuit supplies the required 3.3V to the node system. This chip must be able to handle the peak input from the battery (4.2V) at the peak current draw (~300mA).

Requirement	Verification
• The voltage regulator must provide 3.3V $\pm$ 5% from a 3.7V to 4.2V source. • Must maintain thermal stability below 125° C at a peak current draw of 200mA.	• Plug in a power supply to the input of the Voltage regulator, and a multimeter to the output, and verify that, while changing the voltage between 3.7-4.2V, the output is 3.3V $\pm$ 5% • Create a test circuit which draws 200mA, and connect it to the output of the voltage regulator. Connect the input to a power supply, and use a IR thermometer to verify the temperature of the regulator

ii.2 Control Unit Subsystem

ESP8266 Controller We would like to use esp8266 WiFi chip to power the logic of the node. This device will obtain all data from various bus stop nodes, and save it in its internal memory until it's able to communicate this data to the central server through a base station node. To obtain the probe

data, the controller will emit a specific MAC address within a WiFi probe request, and then search for a connection request from a bus stop node. It will keep on repeating this cycle until it acquires a connection to a bus stop node, and it will then receive data from the bus stop node. Once this is completed, it will start over and continue looking for another bus stop node. Once the drive-by node is done accepting data and is returning to base, there will be another node, the base station node that will join the network and make a special request to the drive-by node. The drive-by node will then reply with all the data that it has collected to send it to the central server.

Requirement	Verification
• The WiFi chip must be able to emit WiFi probe requests with a specific MAC address • The node must be able to connect to and receive data from a bus stop node.	• We will place a drive by node near a bus stop node, and will check if it discovers the MAC address of the drive by node. • We will run a test where the drive-by node will be in range of a bus stop node, and try to transfer data between the two.

Data Storage Our device needs to store temporary data before it is transmitted to our central server. The esp8266 has built in flash memory that we will use to store the detected MAC addresses and their timestamps before the NRF24L01 can connect to a nearby device and pass the data on.

Requirement	Verification
The device should cache the data of seen mac addresses locally.	After a certain amount of time we will request data from the device and confirm that it successfully kept track of all the data it was logging, and sends it all.

iii. Server

The server takes in data from the drive-by nodes, and performs data analysis.

iii.1 Central Server

We will be building an API to be hosted on a central server which will be in charge of getting the gathered and transmitted data from all the devices, performing the bus frequency optimization, and giving data visualizations. The encrypted data will be submitted through an endpoint, decrypted, and placed in a database for processing. The endpoint will accept POST data via a python / flask backend. The API will also supply data to the frontned as specified in the next section. This process is illustrated in the flowchart in Fig 5.

API / Database / Web interface The central server will run on a docker stack on a linux server. It will be exposed via an API that will listen for new data from the âĂIJdrive-byâĂİ node. It will store this information in a mongodb database and make it available in a web interface written in React and also through the Elasticsearch Logstash and Kibana stack (ELK). We chose to use mongodb because

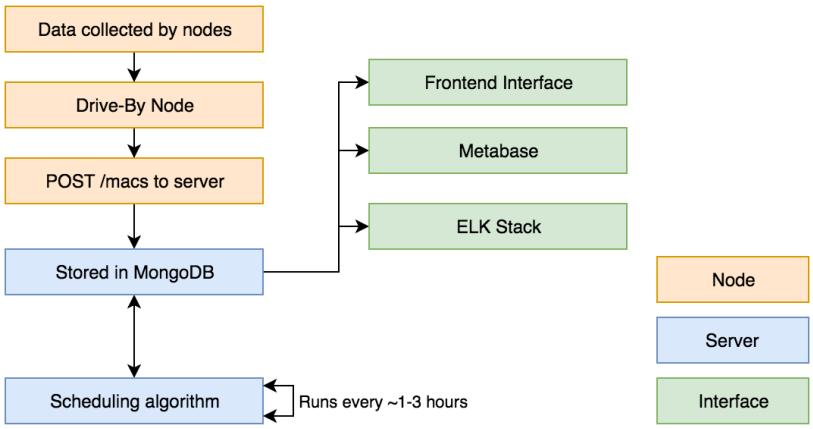

Figure 5: *Software Flowchart*

most of our data will be in JSON format from the API. We will also have a dynamic number of MAC address per every timestamp, which will be easier to group together using JSON format. We could use a relational database for this, but it would mean extra work and more complicated quiries, not to mention the API already will be using JSON so it would need additional data transformation. The API will also host some endpoints to supply data to the frontend, such as the count of mac address over time and the list of the most crowded times for each bus stop. We will display this data in charts using React and recharts.

Requirement	**Verification**
Server should accept MAC data via an API endpoint and store in a mongo database	Test the API using Postman and mongo client to verify data is properly accepted and stored in the database

Security Protocol In order to securely transmit the data form the nodes to the server, we plan on using asymmetric cryptography. Each node will have the public key of the central server and encrypt all the data it sends to the drive by node with this public key. Only the server then can decrypt the data once it arrives.

Requirement	**Verification**
Nodes can successfully encrypt data that only the central server can decrypt	Check to make sure all data leaving each node is fully encrypted by monitoring the air and network traffic. Check that the decrypted data once received by the server is indeed the same exact contents as what was originally encrypted by hashing both the pre-encrypted value and the post-encrypted value

iii.2 Data Analysis

Prediction Models The data will be reorganized in a way such that each entry corresponds to a count and a timestamp. Since we want to make reliable time series predictions, we would want to train a machine learning model that could make full use of our data. A powerful type of neural network designed to handle sequence dependent data is called recurrent neural networks. In particular, we choose the LSTM model to make our predictions.

Requirement	Verification
Predictions must be made and be within a reasonable accuracy range	1. Deploy node in a populated space like ECEB for a week to accumulate data 2. Train model using this data 3. Make predictions using this model and check that the predictions are within 20% interval of the ground truth by taking pictures of the area and manually counting the amount of people

Scheduling Algorithms To schedule the buses, we will first predict the amount of people required at each bus station, and then followed by finding out how many bus should be deployed in advanced to ensure that every bus has the optimal load.

Additionally, we may include a manual adjustment variable to fine-tune the bus stops with a lot of interference. This would allow us to make corrections to the people-count for bus stops that have restaurants or apartments next by.

Requirement	Verification
We will need to provide effective scheduling to reduce the unbalanced bus load problem	• We will use software to simulate the effects of our scheduling, and verify that our scheduling does in fact alleviate the problem by comparing the non-scheduled results and the scheduled results and see that, on average, the standard deviation of the load on the bus has decreased by at least 20% • Additionally, we will also need to guarantee at least one bus per 20 minute time window

iv. Schematics

Figure 6 illustrates the circuit schematics.

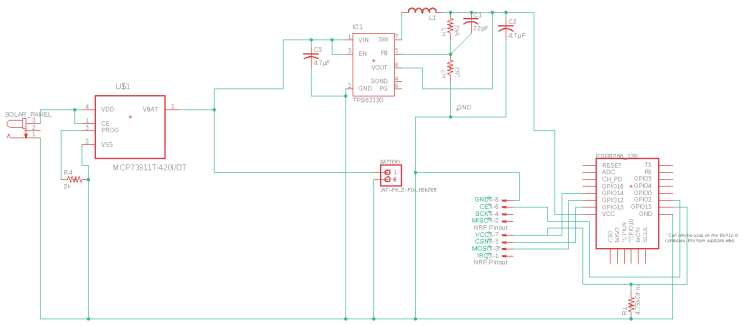

Figure 6: *Basic Hardware Design*

v. Board Layout

Figure 7 illustrates the PCB design and layout.

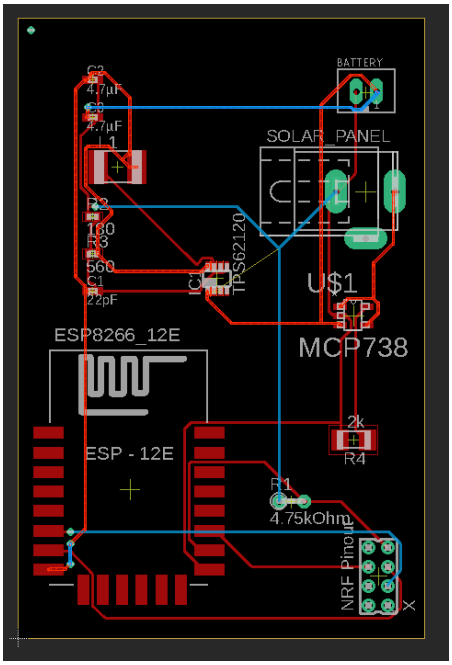

Figure 7: *PCB Design*

vi. Tolerance Analysis

We are using a solar panel that can vary a lot in the amount of power it provides depending on the sun and light it receives. We want to maintain an average power to sustain the device, plus a little extra to charge the battery. On the esp8266 datasheet, which will be the device that draws the most power, the power draw is between 0.35 W to 1 W max during continuous transmission. However since we are in monitor mode, the chip could have higher or lower power requirements. We will assume the ESP6288 chip will need approximately 0.5W of power continuously. The NRF chip has a max power draw of 0.3W. We will assume we need approximately 1 W to power the chip and charge the batter considering about 0.2W of loss through heat. This means that our solar panel should average 1W of power over the course of the sunlit day.

The average light intensity (Outdoor average daily light integral) in Illinois can be measured in moles of light per square meter per day, and it ranges from 15 - 50 $mol \cdot m^{-2} \cdot d^{-1}$ based on a relevant report [18]. The Growth Chamber Handbook [16] estimates approximately 219 $W \cdot m^2 \approx 1$ $mol \cdot m^2 \cdot s^{-1}$. Given a 11$cm$ by 6cm solar panel, the surface area is 0.0066m^2. This suggests that it will be able to receive between 2.3 W - 7.9 W, assuming that the solar panels are at least 10% efficient.

The *Solar Energy Local* organization also provides us with local solar energy data and statistics across the US. In particular, since our preliminary device deployment location is physically situated in Illinois, we obtained the monthly solar radiation levels chart in Chicago, IL. This chart is reproduced in Fig 8.

Once again, given a 11cm by 6cm solar panel, the surface area is 0.0066m^2. The monthly radiation plot suggests that the device will be subject to a radiation between 2.25 $\frac{kWh}{m^2 d}$ and 5.93 $\frac{kWh}{m^2 d}$ from January to December. Furthermore, this suggests that it will be able to receive a power between 2.97 W and 7.83 W, assuming that the solar panels are approximately 20% efficient. This would be a more realistic figure.

This estimation (within one order of magnitude) means that our device should be able to work under standard conditions.

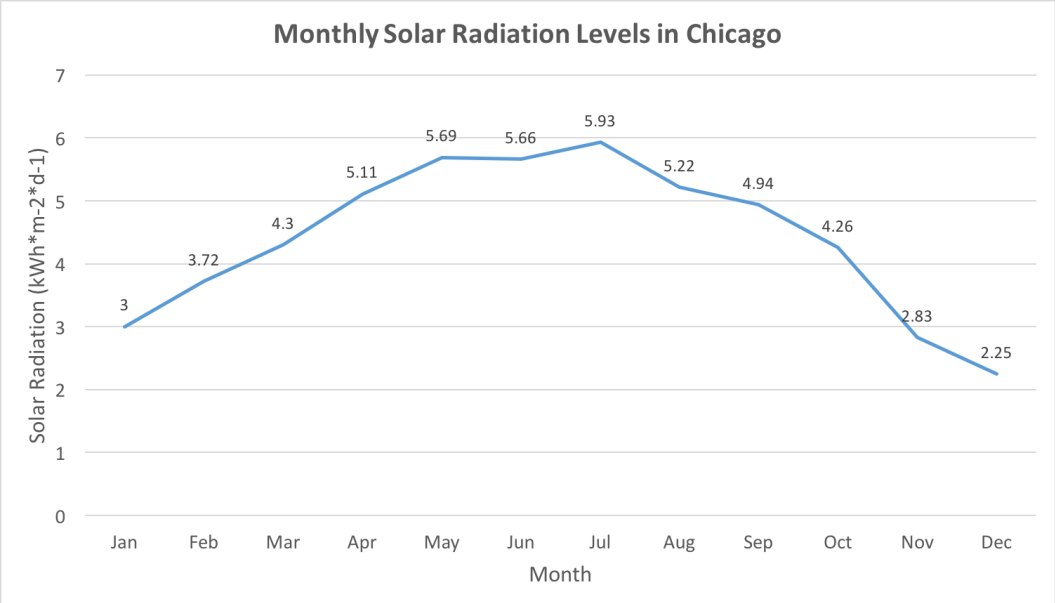

Figure 8: *Solar Radiance*

III. Cost and Schedule

i. Cost

i.1 Labor

Rate: $50/hour

Total estimated hours
Connor: 80
80 x 2.5 = 200
Scott: 80
80 x 2.5 = 200
Aashish: 80
80 x 2.5 = 200
Total: 240
240 x 2.5 = 600
$50 * 600 = $30,000

i.2 Parts

```
ESP8266 from ebay
    Description: WiFi chip for listening to probe beacons
    Quantity: 5
    Cost / item: $3.11
    Total: $15.55

ESP8266 from amazon
    Description: WiFi chip for listening to probe beacons
    Quantity: 2
    Cost / item: $6.5
    Total: $13

NRF24L01+PA+LNA
    Description: Radio chip for transmitting captured data to drive-by node
    Quantity: 5
    Cost / item: $2.53
    Total: $12.65

ATMEGA328P
    Description: Microcontroller for controlled logic between NRF radio chip and ESP8266
    WiFi chip
    Quantity: 2
    Cost / item: $1.88
    Total: $3.76

4200mAh Battery, Panasonic 565068
    Description: 4200mAh standard Li-ion rechargeable battery
    Quantity: 2
    Cost / item: $10.88
    Total: $21.76
```

```
Medium 6V 2W Solar panel - 2.0 Watt (Adafruit)
    Description: 11cm x 14cm solar panel
    Quantity: 1
    Cost / item: $29.00

BQ24168RGET Battery chargers
    Description: IC Battery charger for li-ion battery
    Quantity: 1
    Cost / item: $5.39

JST 2 pin connector
    Description: A connector mounted on the PCB that connects the battery
    Quantity: 1
    Cost / item: $0.75

TPS62021
    Description: 3.3V voltage regulator
    Quantity: 1
    Cost / item: $2.13
```

Total cost for prototype: $55.67

ii. Schedule

Date	Scott	Connor	Aashish
9/24	Research for the project proposal	Set up gitlab repos, add basic shell and templates for development. Start writing some basic API code for accepting MAC addresses	Research for project proposal
10/1	Generate dummy data to simulate the passenger count time series	Finish basic API for accepting data from the "drive-by" node	Design and order the PCB
10/8	Start writing code for data analysis	Finish basic API for devices to use	Test hardware. Test capabilities of WiFi and RF chip
10/15	Put the node together after the PCB arrives. Start testing and debugging.	Basic frontend to display and manage data. ELK setup	Verify PCB is completed, hardware testing, micro controller code
10/22	Work on fixes from the previous week	Work on getting multiple devices to talk to each other. Get encrypted communication setup	Get the microcontroller code working. The communications need to be completed at this stage.
10/29	Arrange several nodes together to simulate potential scenarios and collect data	Start trying to get full devices working and continue working on frontend	Second round of PCB ordering if needed. Testing and debugging.
11/5	Collect more data. Perform field tests.	Continue working on full system and continue on frontend	Field testing hardware nodes, simulating bus stop and drive-by node
11/12	Perform analysis on field test results, train machine learning models	Goal to have full working setup with multiple devices gathering data and drive-by node fetching data and uploading to master server	More field testing
11/19	Prepare and present predictions	Cleanup, final touches, productizing	Final touches
11/26	Prepare final demo	Prepare final demo	Prepare final demo
12/3	Prepare final report	Prepare final report	Prepare final report

IV. Safety and Ethics

i. Potential Hazards

Our devices will be outdoors in potentially harsh conditions, we need to make sure the internals of the device are properly insulated from rain, excessive heat, and other harsh conditions that could cause the battery to explode or ignite. We plan to use a rather small lithium ion battery, but even so they can release a large amount of power in a short amount of time is the battery is ruptured. We plan to use a properly insulated container that will be sealed to prevent water or snow to enter.

ii. Data Privacy

Because we are collecting data from phones and associating it with unique identifiers, this gives us the ability to track the movement of these devices, and therefore most likely the owner of the device as well. This information can be used for inappropriate uses if it does not remain anonymous. Because of this, there is an ethical dilemma of how the data will be used or available to companies that will pay for the service. We plan to keep the data anonymous and detached from any PII (personally identifiable information). We plan to look at the data as a whole, not on an individual level. Our goal is to optimize bus routes and give bus companies insights into how their customers are using their service, not identify and track individual people or devices. We also will have each node encrypt all the data it sends using the public key of the server to make sure nobody has access to the data besides the central server.

iii. Digital Security

Security is a big concern with the data we are collecting. We plan to use asynchronous cryptography to have a public key that will encrypt all data that is sent from each individual device. Theoretically anybody could pull the data, but it would be encrypted with a public key. The private key would be stored only on the central master server that will take the collected data, decrypt it, and store it.

iv. General

In addition to the above, our team pledges to follow the IEEE Ethics guidelines [14] as well as the ACM Ethics guidelines [7] as closely as possible.

We address the following terms that are relevant to our project.

`1. To hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment`

Like previously stated we will properly insulate the physical components of the nodes

`2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist`

All our data will be viewed on a high level and not on the granularity of the specific consumer. This will prevent an individual from being targeted or tracked from the bus company.

`6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations`

We will make sure that we keep up to date with the most recent technological updates and security issues. All technicians shall be trained before handling the electronics. All our data will be viewed on a high level and not on the granularity of the specific consumer. This will prevent an individual from being targeted or tracked from the bus company.

`9. To avoid injuring others, their property, reputation, or employment by false or malicious action`

All of our data will be anonymized and not be looked at on an individual level.

v. Current Competition

Here is a link to devices that track WiFi devices for commercial clients. It helps detect rouge devices and keeps track of potential WiFi enabled threats. It is however in a different market and industry than the one we are currently targeting. They focus on more commercial grade office building types, not the data aggregation that we do.

https://www.accuware.com/products/locate-WiFi-devices/

References

[1] Simon J. Berrebi, Kari E. Watkins, and Jorge A. Laval. A real-time bus dispatching policy to minimize passenger wait on a high frequency route. *Transportation Research Part B: Methodological*, 81:377 – 389, 2015. Optimization of Urban Transportation Service Networks.

[2] Larry A. Bowman and Mark A. Turnquist. Service frequency, schedule reliability and passenger wait times at transit stops. *Transportation Research Part A: General*, 15(6):465 – 471, 1981.

[3] Avishai Cede. *Public Transport Timetabling and Vehicle Scheduling*, chapter Chapter 2, pages 30–57.

[4] Avishai Ceder. Bus frequency determination using passenger count data. *Transportation Research Part A: General*, 18(5):439 – 453, 1984.

[5] Avishai Ceder and Nigel H.M. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 1986.

[6] Champaign Urbana Mass Transit District. 1 yellow: Weekday - day. `https://mtd.org/maps-and-schedules/routes/1-yellow-weekday-day/`. Accessed: 2018-09-13.

[7] Association for Computing Machinery. Acm code of ethics and professional conduct. `https://www.acm.org/code-of-ethics`. Accessed: 2018-09-14.

[8] B. Gavish, P. Schweitzer, and E. Shlifer. Assigning buses to schedules in a metropolitan area. *Computers & Operations Research*, 5(2):129 – 138, 1978.

[9] Valerie Guihaire and Jin-Kao Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251 – 1273, 2008.

[10] Jan Owen Jansson. A simple bus line model for optimisation of service frequency and bus size. *Journal of Transport Economics and Policy*, 14(1):53–80, 1980.

[11] W. Lampkin and P. D. Saalmans. The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study. *Journal of the Operational Research Society*, 18(4):375–397, 1967.

[12] Joao Mendes-Moreira, Luis Moreira-Matias, Joao Gama, and Jorge Freire de Sousa. Validating the coverage of bus schedules: A machine learning approach. *Information Sciences*, 2015.

[13] Herbert Mohring. Optimization and scale economies in urban bus transportation. *The American Economic Review*, 62(4):591–604, 1972.

[14] Institute of Electrical and Electronics Engineers. Ieee code of ethics. `https://www.ieee.org/about/corporate/governance/p7-8.html`. Accessed: 2018-09-14.

[15] University of Illinois Faculty and Services. University of illinois faculty and services: Transit. `http://www.fs.illinois.edu/services/more-services/tdm/transit`. Accessed: 2018-09-09.

[16] John C. Sager and J. Craig McFarlane.

[17] N. Seshagiri, R. Narasimhan, S. L. Mehndiratta, and B. K. Chanda. Computer generated timetables and bus schedules for a large bus transport network. *Transportation Science*, 3(1):69–85, 1969.

[18] Ariana Torres, Roberto Lopez, and Purdue Floriculture. Measuring daily light integral in a greenhouse. 05 2012.