Fleet Monitoring System

PROJECT PLAN

Sdmay18-18
Client/Advisor: Lotfi Ben-Othmane
Venecia Alvarez – Point of Contact
Kendall Berner – Project Manager
Matthew Fuhrmann – Report Manager
William Fuhrmann – Test Engineer
Anthony Guss – Technical Lead
Tyler Hartsock – Web Manager
sdmay18-18@iastate.edu
https://sdmay18-18.sd.ece.iastate.edu

Revised: 9-21-2017/1.0

Table of Contents

1 Introduction

1.1 PROJECT STATEMENT

Our project is to make a fleet monitoring system. We have been provided an Android embedded device that can interface with a vehicle's CAN BUS network. We will develop an Android application for the Android embedded device that queries the vehicle's components and relays the collected data to a server. This server will process and store the data. The server will also provide a website with real-time and historical data on location of all vehicles and internal data from the vehicles such as gas consumption and vehicle diagnostics. The website will also show useful interpretations and statistics of the data, and will allow website users to message the Android devices that are in the vehicles.

1.2 PURPOSE

The purpose of this project is to give fleet managers a better way of tracking information on the vehicles in their fleet. Fleet managers often find it difficult to know the current location of their vehicles or where their vehicles have been, and knowing this information helps them ensure that their vehicles are being used correctly and efficiently. Fleet managers also need better ways of predicting maintenance needs and vehicle operation costs. Data on fuel consumption and vehicle diagnostics can help fleet managers predict the needs of their vehicles. Having statistics on fleet operation will help fleet managers who are trying to improve the efficiency of their fleet.

1.3 GOALS

We want to create a system that efficiently gathers and transmits information from vehicles, processes and stores that data, and displays it in a meaningful way for fleet managers. Our project has several different components involved. Our main goal is to have all those components working and integrated with each other before we get to our final deadline. In regards to soft skills, our team has set a goal to learn more about project management and agile development. Through this project experience, we will see what the entire software development process and life cycle really look like.

2 Deliverables

Our project will have three components. The first is an Android microcontroller that will get data from the CAN BUS network of a vehicle. The second will be a server that will take in data from the Android microcontroller and store the data, as well as send necessary data to the website. The third is a website that will serve as a dashboard for managers of fleets to be able to view specific data about their vehicles.

The webpage will be made for use by the manager of the fleet. It will have a simple and intuitive design made for ease of use. This web page will display a live map tracking the locations of all vehicles in the fleet in real time. A single vehicle can be selected, which will bring up a profile that will display information about that vehicle such as remaining gas, miles driven, driver, etc. The webpage will also have a page that details the stats of the fleet as a whole, so that the manager can see how much gas the entire fleet uses over a month, for instance. Graphs will be provided to make the statistics and information easier to parse.

Our server will be created using Node.js. The server will handle incoming raw data from the embedded devices and process this information into usable information that it will then put into a database. The server will also handle data request from the web page. Both the incoming and outgoing data will be transmitted through REST API calls. This will allow us to use a single protocol across our entire system. The Node.js server will be hosted on the Google cloud platform, allowing us to scale our resources to the demands of our services.

The Android microcontroller will run an Android app that is used in the vehicles. It will connect to the vehicle's CAN BUS network and query for useful information. It will also acquire GPS data from a separate GPS hardware device. The microcontroller will store the data locally so that it can guarantee that it gets sent to the server. The microcontroller will be able to send data that it collects to the server, and will be able to receive messages from the server and display them on a screen connected to the microcontroller.

3 Design

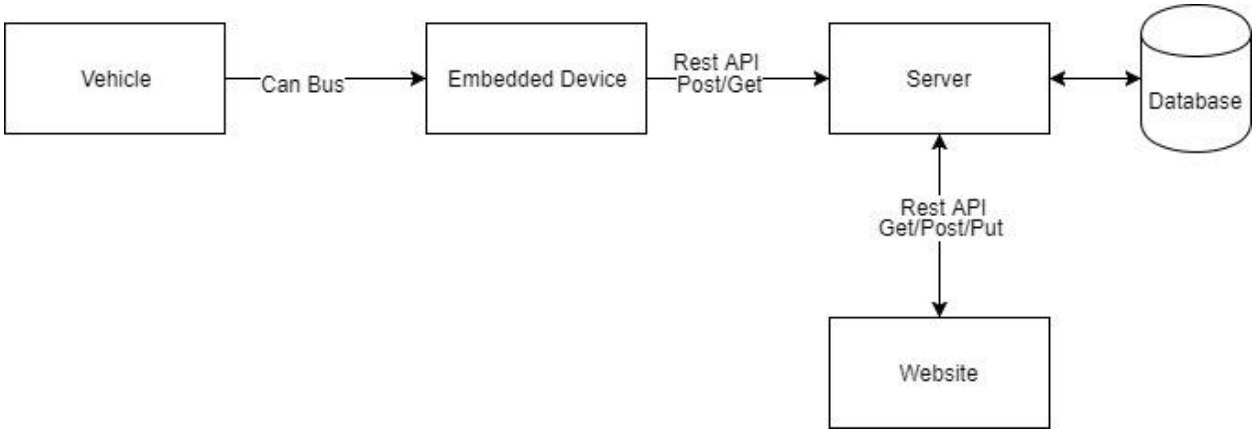
Describe any possible methods and/or solutions for approaching the project at hand. You may want to include diagrams such as flowcharts to, block diagrams, or other types to visualize these concepts.

3.1 PREVIOUS WORK/LITERATURE

We researched several other fleet management applications to get an idea of what kind of features people would want, as well as to see where there are some holes in the market that we could fill with a superior design.
Most applications we found came as two applications, one being a mobile app that the drivers would use, and one being a web app that the manager would use. We like this design and will likely move to it after we complete our 1.0 version. For now, we don't intend to have any manually entered data on the driver side.
We also noticed that a lot of things needed to be entered manually for these applications. We intend to improve on these systems by automating as much as

possible. For example, several apps required the drivers to enter how much fuel they got at a gas station. We are going to try to get information like automatically.
Other common features were fuel and speed tracking and the real time map displaying the locations of other drivers. For our 1.0 we selected the most ubiquitous and most useful seeming features to include. In the future we intend to add functionality that goes beyond anything we've seen in the market.
See the fleet management software we reviewed in the references section.

3.2 PROPOSED SYSTEM BLOCK DIAGRAM



The vehicle in the block diagram represents the vehicle that our embedded device is connected to via the can bus. The vehicle will send data from the can network through the can bus to the embedded device. The embedded device will take in data from the Can Bus and send the important data to the server via Rest post and get calls. The server will receive data from the embedded device and put this data into a database. This database is managed with MongoDB and Mongoose. The server will receive data from the website. The data the server receives from the website includes new manager accounts via Rest post calls, and manager vehicle updates via Rest put calls. The server will also send requested data to the website with Rest get calls. The website will send data creating or updating manager account with Rest post and put calls. The website will also request data using Rest get calls.

3.3 ASSESSMENT OF PROPOSED METHODS

For the embedded code, we will create an app for Android. An Android app will work well for our purposes because the hardware we were provided runs for Android. We will be able to send data to the server on our application. The app will also allow us to receive information from the server, such as messages that we can display on the screen attached to the board. We can also use the Android app to locally store data before we send it to the

Our server code will use Node.js because it is a client requirement. A disadvantage of using Node.js is that it is slower than a Java server. Node.js is also single-threaded, so it is slow when many requests are being made. The main advantage of Node.js is that it has high throughput, and therefore it has high scalability. The server will be hosted on Google Cloud. Google Cloud will be useful to us because it is well-documented and easy to use. A disadvantage is that although they have competitive prices, it is still a cost that our team has to account for. Our server will be connecting to a SQL database for data storage. SQL has a lot of documentation, a well-defined standard, and has a lot of power and speed in its queries. SQL's interface can be difficult for user's to access, but it should not be difficult to make more queries once we successfully make one.

For the front end we will use HTML and JavaScript to display information. We will be using the
AngularJS framework. AngularJS reduces the need for functions. It is faster because it modifies the page DOM directly instead of adding inner HTML. AngularJS gives multiple ways to accomplish a task. This can be a disadvantage because it is difficult to know what is the best way to do one thing. It is not a very scalable framework because it requires a lot of refactoring when trying to extend existing functionalities. We will plot data using the Google Maps API and Chart.js. We have found both of these to be very well-documented and easy to implement. They work very well with our design plans for the front end.

3.4 VALIDATION

Validation and testing will be done by taking in raw data from a car and comparing the data recorded by our system to the actual observed data. Validation from this method will be limited because we only have access to one device and one car meaning that there will only be one data source. In order to solve this issue, we will be creating data sources that we will feed into our system allowing us to have a much larger data pool than only using one source. Doing this will verify that data from the vehicle matches the data being sent and save to the database. Data from multiple trips will be saved and then changed to run as if it were multiple vehicles travelling at once. Unit testing will also be used to test the completeness and correctness of each function. Vulnerability testing will be used to ensure that transmitted data will be secure and can only accessed by the processes or processors with permission.
The connection between the database, server, and website also needs to be verified. Data will be stored in the database and accessed on the front end using the server API. If the data that was put in the database matches what is received on the front end, then the connection is validated.

4 Project Requirements/Specifications

4.1 FUNCTIONAL

The product shall:

- Gather data from a vehicle
- Transmit data from the vehicle to the server
- Process raw data from the vehicle on the server
- Record vehicle data into a database
- Display a map with a location of all vehicles in the fleet
- Display historical data for a certain vehicle (location, gas usage)
- Allow managers to register vehicles to their fleet
- Allow managers to view information about the vehicles in their fleet

4.2 NON-FUNCTIONAL

The product shall:

- Be able to be used by vehicles at any time and location
- Utilize Google Cloud services
- Send messages from the manager to the driver(s) in a timely manner
- Only allow managers to view fleet data on the dashboard
- The server implemented using Node.js

4.3 STANDARDS

We will use the agile development method. Our coding standards will align with the basic coding standards for each programming language used. We will also use vulnerability testing to ensure that the data can only be accessed by processes or processors with permission. This ensures that we will not be doing anything unethical. Coding standards are important for our project, because the client intends to modify our software in the future to fulfill the needs of future clients. Following coding standards for the programming languages we use will make fixing, and modifying the software set much quicker, easier, and cheaper.

5 Challenges

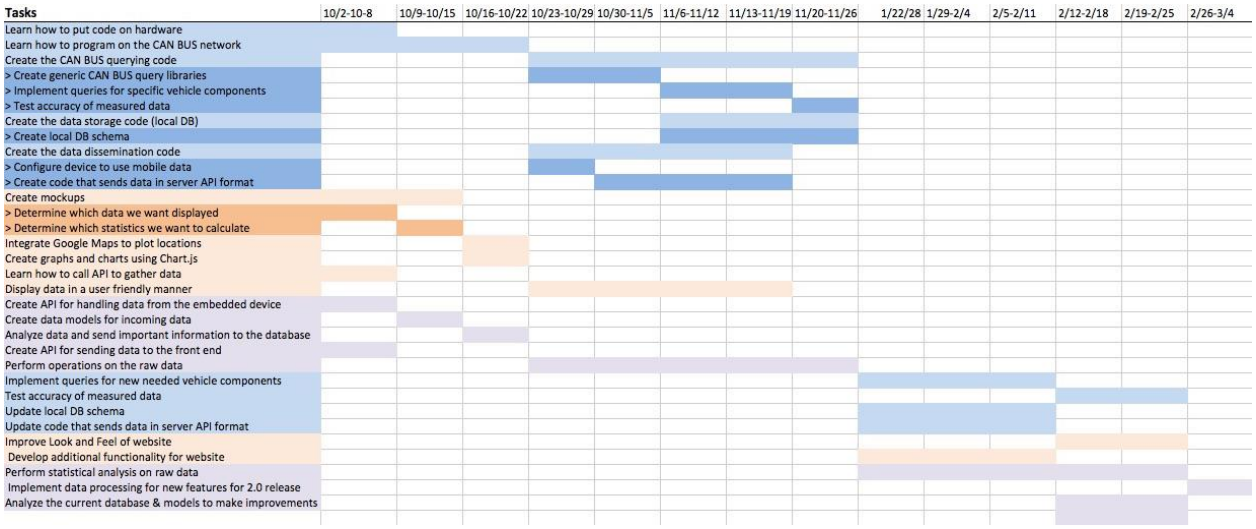
We have only one embedded device, so we will only have one vehicle in our fleet. Any real data we wish to use will require us to drive around in that vehicle, so we will not have data in abundance. Issues may arise for larger fleets that we never anticipated because we will only have the one vehicle for testing.

Several members of our team have very limited knowledge of embedded systems. Progress could slow substantially if our embedded systems experts were to fall ill or be otherwise incapacitated. Even those with experience in embedded programming have no experience in CAN BUS programming, so they will have to quickly come up to speed on how to program using the network's protocol.

Our server will be created using Node.js. This is a platform that we as a team do not have a lot of experience in and is not optimal for our server processing needs. This will

require our group members to perform additional research in order to learn and adapt to this platform.

6 Timeline



6.1 FIRST SEMESTER

Breakdown your timeline into detail of what needs to be done by the end of the first semester. You may want to include division of work amongst the team.

Tasks for embedded side:
Learn how to put code on hardware (10/2-10/9)
Learn how to program on the CAN BUS network (small example prototypes for querying, network interaction/connection) (10/2-10/23)
Create the CAN BUS querying code (10/23-11/27)
- Create generic CAN BUS query libraries (10/23-11/6)
- Implement queries for specific vehicle components (11/6-11/20)
- Test accuracy of measured data (11/20-11/27)

Create the data storage code (local DB) (11/6-11/27)
- Create local DB schema (11/6-11/27)

Create the data dissemination code (send data to server) (10/23-11/20)
- Configure device to use mobile data (10/23-10/30)
- Create code that sends data in server API format (10/30-11/20)

Tasks for front end:
Create mockups (10/2-10/16)
- Determine which data we want displayed (10/2-10/9)

Determine which statistics we want to calculate (10/9-10/16)
Integrate Google Maps to plot locations (10/16-10/23)
Create graphs and charts using Chart.js (10/16-10/23)
Learn how to call API to gather data (10/2-10/9)
Display data in a user friendly manner (10/23-11/20)

Tasks for the server:
Create API for handling incoming data from the embedded device. 10/2-10/9
Create data models for incoming data.10/9-10/16
Analyze data and send important information to the database.10/16-10/23
Create API for sending data to the front end. 10/2-10/9
Perform operations on the raw data to create useful information for the graphs on the front end.10/23-11/20

6.2 SECOND SEMESTER

Detail what needs to be done in the second semester. You may want to include division of work amongst the team.

Embedded:
Implement queries for new needed vehicle components (1/22-2/12)
Test accuracy of measured data (2/12-2/26)
Update local DB schema (1/22-2/12)
Update code that sends data in server API format (1/22-2/12)

Front-end:
Improve Look and Feel of website (2/12-2/26)
Develop additional functionality for website (1/22-2/12)

Server:
Perform statistical analysis on raw data (1/22-2/26)
Implement data processing for new features for 2.0 release (2/26-3/1)
Analyze the current database and data models and make improvements (1/12-2/26)

After the first semester, we plan to have version 1.0 of the software set completed. During the second semester we aim to complete version 2.0 which includes improved UI, and adding features to make the software set more complete and easier to use. There are many possible features that could be added, and we are still in the process of deciding which features will create the best end product.

7 Conclusions

Our project is a fleet management system. We have split our team into three parts, the embedded systems, server, and front-end webpage. The embedded systems will need to efficiently read and transmit data to the server. The server will process and store this data in a database. Our webpage will make requests to the server which will supply it with the necessary data to populate its graphs and other displayed data.

By dividing up the tasks among our specialized teams, we will be able to make efficient progress as each team will be able to focus primarily on just one area.

8 References

List all the sources you used in understanding your project statement, defining your goals and your system design. This report will help you collect all the useful sources together so you can go back and use them when you need them.

High Point GPS (http://www.highpointgps.com/)
Fleet Complete Fleet Tracker (https://www.fleetcomplete.com/en/products/fleet-tracker/)
Fleetmatics Now (https://www.fleetmatics.com/now)
collectiveFleet (http://www.collectivedata.com/fleet_management_software.html)
ManagerPlus (http://www.managerplus.com/lps/managerplus)
Rhino Fleet Tracking (http://www.rhinofleettracking.com/)
Linxup (http://www.linxup.com/)
Silent Passenger (https://vehicletracking.com/features/)
Azuga Fleet (https://www.azuga.com/)
Onfleet (https://onfleet.com/)
Fletilla FleetFACTZ (https://fleetilla.com/products/fleet-management-solutions/real-time-tracking)
Fleetio (https://www.fleetio.com/)

9 Appendices

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. You may also include your Gantt chart over here.