Fleet Monitoring System

Sdmay18-18

http://sdmay18-18.sd.ece.iastate.edu/

Client/Advisor: Lotfi Ben-Othmane

Group Members: Venecia Alvarez, Kendall Berner, Matthew Fuhrmann, William Fuhrmann,
Anthony Guss, Tyler Hartsock

Problem Statement

Problem:

- Companies have many vehicles
- Vehicle fleets are inefficient and costly

Solution:

- Allow a fleet manager to see real time data of vehicles
- Allow a fleet manager to see periodic reports about each vehicle and each driver

Functional Requirements

The product shall:

- Gather data from a vehicle's ODB-II port
- 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

Non-Functional Requirements

The product shall:

- Be used by vehicles at any time and location
- Utilize Google Cloud services
- Only allow managers to view fleet data on the dashboard
- Have the server side code made in Node.js
- Use AngularJS on the client side

Market Survey

Other Fleet Management Applications:

- Mobile App vs OBD II
- Live tracking, statistics, vehicle data
- Live map of fleet

Our Application vs Rest of Market:

- More useful interpretations of internal data
- Automate tedious tasks

Basic Design

3 Components

- Microcontroller to retrieve vehicle data
- Server with database to receive, store, and relay
- Website to display data to manager

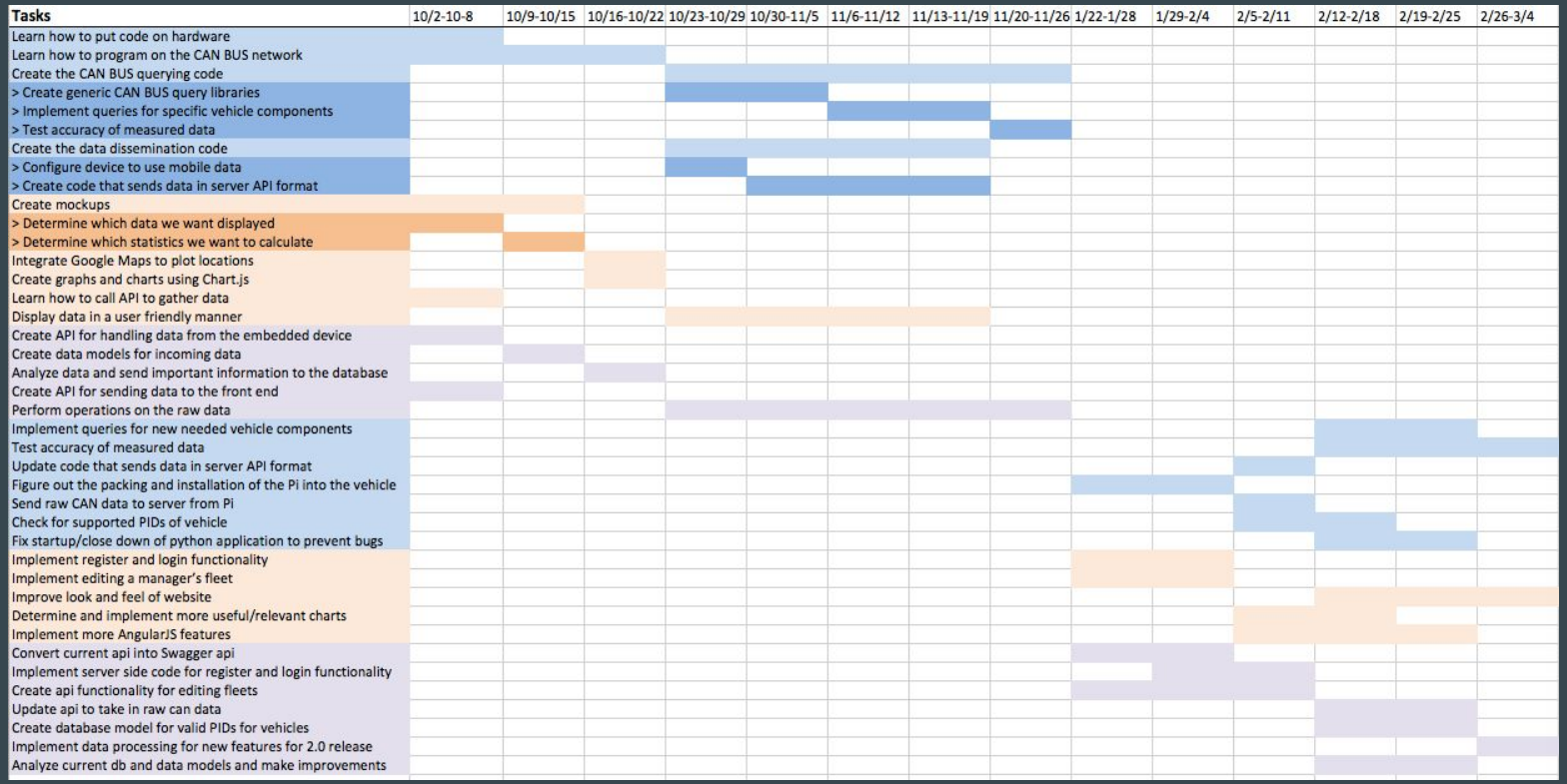
Risk Management

- Limited knowledge of embedded systems → Lots of time spent researching, working with other students with embedded systems experience
- Unable to use original hardware → Use a Raspberry Pi
- Difficulty testing with just one device → Making fake test data
- Race conditions with the map → Research common solutions and redesign

Project Budget

- Original hardware: Android Board, development kit
  - ~$500
- New hardware: Raspberry Pi, GPS, and connector
  - $50 + $30 + $10 = $90
- Google Cloud services
  - $40/month * 5 months = $200

Project Schedule



Detailed Design - Front End

User Interfaces: Emphasis on visualizing data

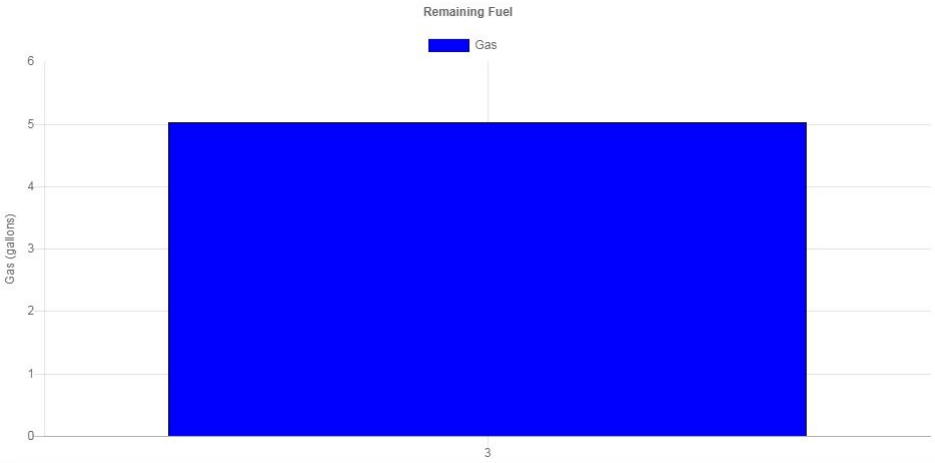
Technologies Used: AngularJS, Chart.js, Google Maps API, Bootstrap

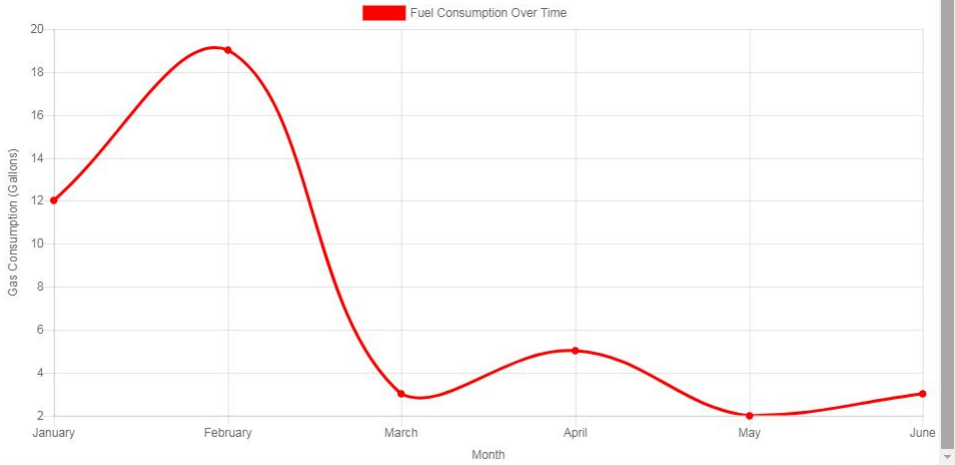
Test Plan: UwAmp/XAmpp for local testing

View All

Refresh Map

Stats





Detailed Design - Server

Technologies Used: NodeJS, MongoDB, Mongoose

Test Plan: Automated API Testing Using Postman, Unit Testing for Data Analysis

Hosted on Google Cloud Compute Engine

Swagger Documentation

Transitioning into a Swagger based API

Detailed Design - Microcontroller

Three Modules: gps_interface, can_interface, server_send

Technologies Used: Python, PiCAN2, Adafruit Ultimate GPS, gpsd, PyCan, requests

Test Plan: unittest.py, verify with simulator, compare with car actuals

Current Project Status

Prototypes for Python application, server, and front-end completed with interoperability.

Group Contributions

Venecia: Client side, emphasis on Google Maps API and AngularJS

Kendall: Client side, emphasis on Chart.js

Matthew: Android application for OBD-II, GPS, sending data. Attempted to fix Android hardware.

William: Android, Python, and ODB-II

Anthony:

Tyler:

Plan for Next Semester

Expand functionality for server and front-end.

Move to real vehicle use for Raspberry Pi.

Increase PID support for Python application.