Team 21: Project Mesh Network

Advisor: Craig Rupp
Client: Radek Kornicki (with Danfoss)
Team: William Paul, Colton Smith, Gage Tenold, Ryker Tharp, Collin Vincent, Cody Lakin

Overview

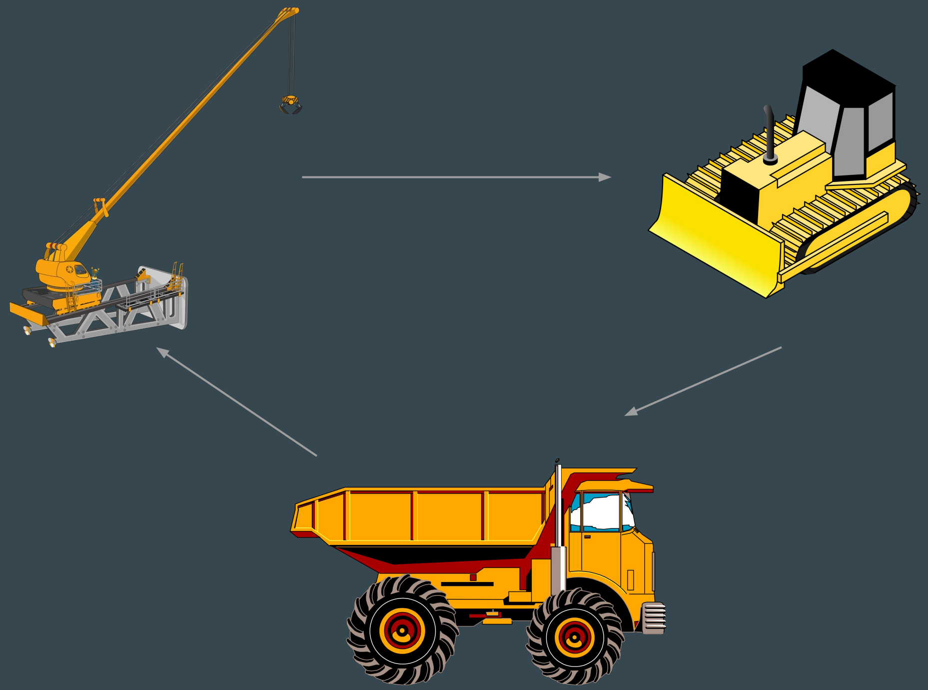
NETSTRUCTION

Advisor: Craig Rupp
Client: Radek Kornicki (with Danfoss)
Team: William Paul, Colton Smith, Gage Tenold, Ryker Tharp, Collin Vincent, Cody Lakin
Website: https://sdmay19-21.sd.ece.iastate.edu/

Problem Statement

- Danfoss supplies machinery to large worksites with a lot of moving parts
- Repairs to these systems can be very costly, but can avoided with proper upkeep
- Machine telemetry can help, but infeasible in areas without infrastructure
- Our goal is to build a local mesh network with a centralized hub to compile machine telemetry data

Conceptual Sketch



- Each vehicle equipped with Raspberry Pi capable of reading CAN data
- Pis equipped with custom Sqlite DB
- Each vehicle communicates to one another through a Wi-fi Ad Hoc connection
- Every node in the network will send updates
- Data is ultimately routed back to a hub
- The hub consists of an easily accessible UI with data visualization features

Functional Requirements

- Set up the local network on a user's device
- Add/Remove devices from the network
- Delete a devices information from the network storage
- View the information of all the vehicles
- View the information of a specific vehicle in detail
- View predictions made by the system for each vehicle
- Upload the information collected to a server

Non-functional Requirements

- Scalability: Easy addition of up to one hundred functional nodes
- Availability: Data always on central node, updated often
- Performance: Less than a minute of latency on front-end
- Reliability: Each node logs vehicle data for 30 days
- Usability: Intuitive front-end app, low maintenance node setup

Constraints and Considerations

Limitations
- Must use CANbus interface + Must use J1939 protocol
- Limited machinery access, must simulate data
- For our purposes, a range of at most 100 meters will be tested

Assumptions:
- End users will understand the information presented
- End users will not have a difficult setup phase
- Machinery expected to be on the network will have our hardware installed

Market Research

Fleet Genius

- Uses OBD2 to track vehicle information
- Uploads car data to the cloud
- Uses a “Base Station” and will only share data from cars when connected to its hotspot
- Allows viewing information on base station from mobile app.
- $139/year for up to 25 vehicles
- J1939 compatible
- Zigbee, wifi, bluetooth, cellular, and gps compatible

Market Research
Zubie

- Uses OBD2 to track vehicle information
- Uses cellular connection to upload info real time to the cloud
- App store for extended functionality
- Allows viewing information on base station from mobile app.
- $240/year
- Does not seem to support J1939

Potential Risks	Mitigation Strategies
If the nodes cannot connect reliably at a decent distance away the project will fail	Danfoss will use the highest legal power Wi-Fi cards in their final product stages
Hardware incompatibilities & lack of hardware experience	Do extensive research on hardware options and compatibility
Customers may be unable to use the solution setup and user interface isn't simple	Automate setup as much as possible, develop a very user friendly front end

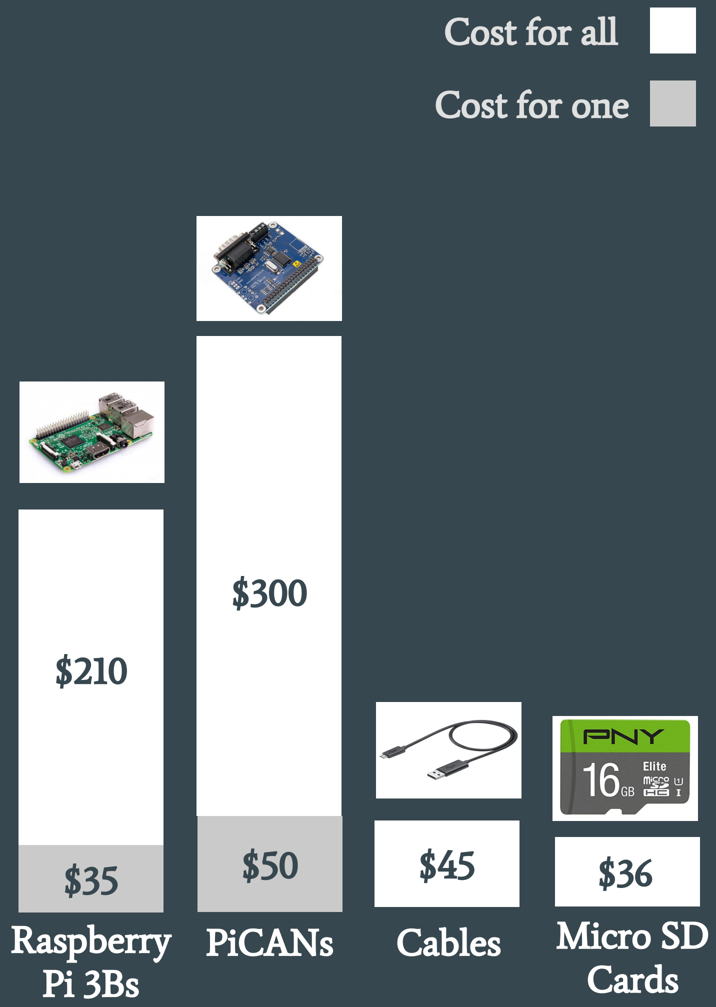
Resource Cost

Software Cost:
All of the software we are using is open source and free to use, so there will be no incurred cost here for us.

Hardware cost:
Fortunately, our client has paid for all of the hardware costs associated with our project.

The total cost comes out to $591:

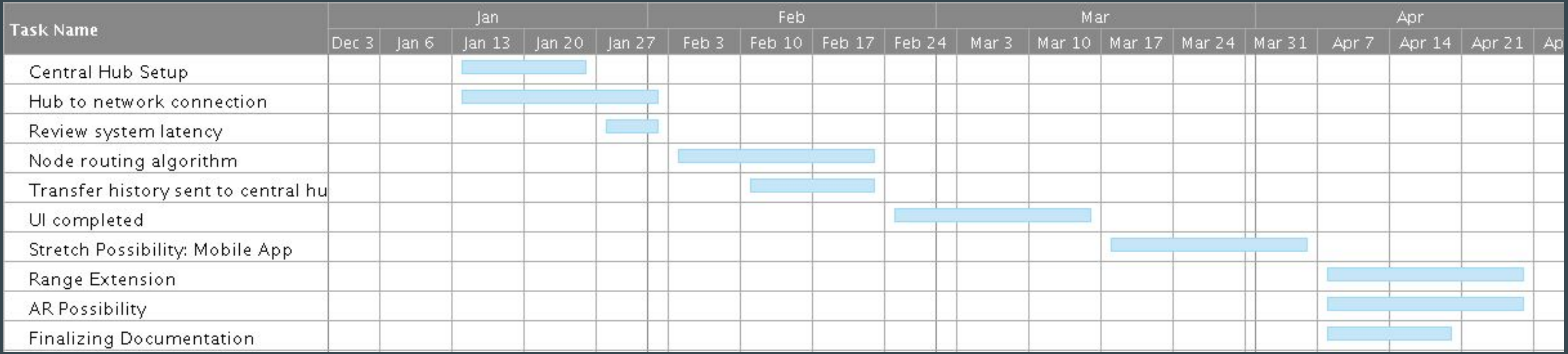
(6) Raspberry Pi 3Bs, (6) PiCANs, (3) OBD2 to DB9 Cables, (6) Micro USB Cables, (6) 16GB Micro SD Cards



Milestones Achieved

- Developed a database schema running with SQLite
- Developed a Python program to parse J1939 CAN data from a VCan interface
- Developed a base UI with Electron
- Designed API for database interaction
- Completed image for Raspberry Pis

Spring Schedule



Raspberry Pi's + CAN Bus

Raspberry Pi 3B's

They are affordable wi-fi enabled computers suitable for use as data collection devices in our proof-of-concept.

Functions:

- Connect to machine sensors/CAN bus - using a PiCAN add-in board
- Collect and store data
- Route data to/from other nodes

CAN bus

CAN is a serial bus protocol used in most automotive systems. It specifies hardware and rules for subsystems, like brakes, to send data. A majority of industrial vehicles use it.

Functions

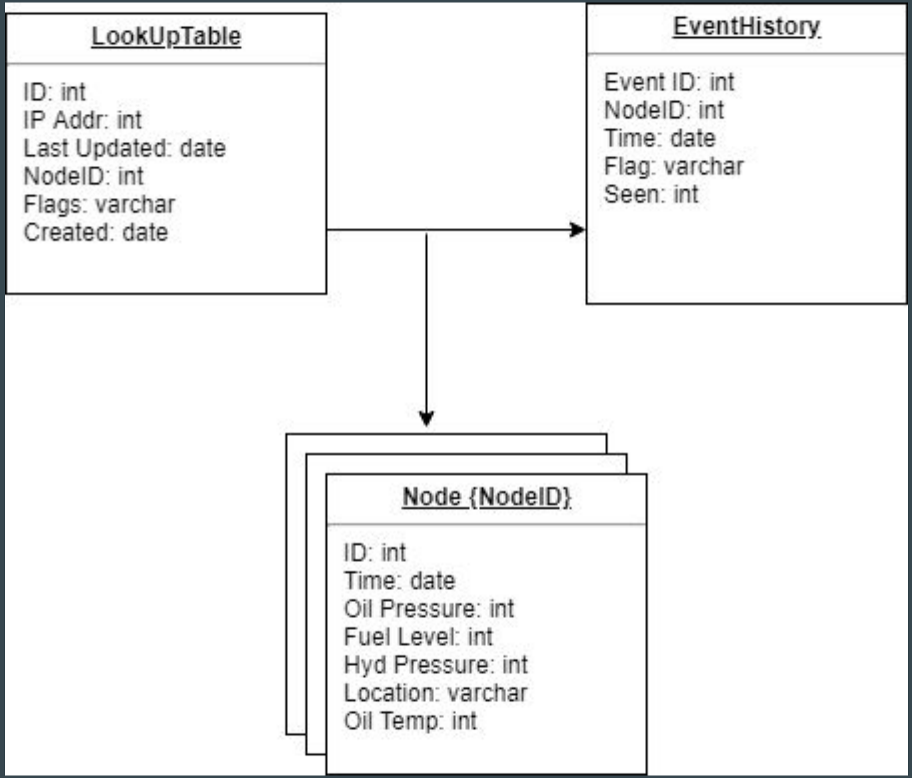
- Broadcasts data from machines to our devices in J1939 format
- Interpreted on Pi's using SocketCAN drivers and Python scripts

Database

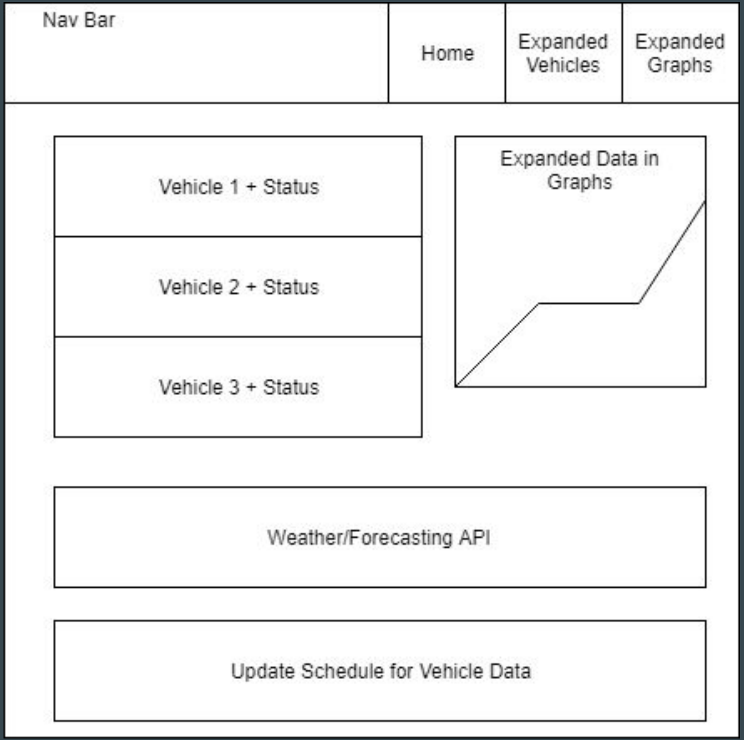
LookUpTable: This contains the specific network information about each node in the network.

EventHistory: This contains the various events that occur in the network, examples include data spikes, and network connections.

Node(ID): One of these tables exist for each node in the network, making data access and update faster and easier.



UI/Frontend



- Designed to be easy to digest, accessible
- Load data from the nodes in JSON format
- Show what vehicles are currently connected
- Breakdown telemetry data into graphs/charts
- Weather API, Push Notifications

Hardware and Software Decisions

NodeJS: The team had familiarity with the software, and has an easy way for generating API’s

SQLite: Solution that let us manipulate and transfer information across the network without internet access.

Pis: Wifi compatible, can be extended to read CAN data, lots of supplemental material up online

Electron: Popular new framework for generating desktop application, fit our needs and was ok’d by the client

Test Plan: Functional

- Test adding and removing nodes
  - Nodes will be added and removed in patterns designed to test this function
  - They should remain functional and the data should be accurate after a successful test.
- Test Accuracy of Data
  - Information throughout the network should be up-to-date, a node’s database will be compared after tests with other nodes.
  - Information being collected must be recorded accurately after being processed through CANBus and stored to the database.
- All functionality must be tested and function without an internet connection.

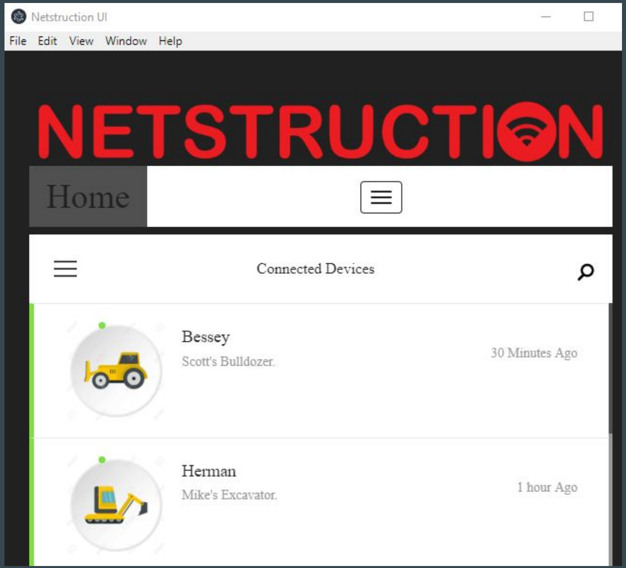
Test Plan: Non-Functional

- Scalability: Functional tests and timed tests with large data sets
  - System expected to work with up to one hundred devices/nodes in the network
  - Each functional requirement should be tested with the data
  - Each other non-functional requirement should be tested with the data
- Performance: Timed tests for each transfer of data / key operation
  - Overall latency from device to front-end should always be less than 1 minute
  - Timed tests for each component with variable data loads and routes
- Usability: Use case tests for functionality, ease, and client satisfaction
  - Team members, client, and external volunteers will test walk through use cases and give feedback

Prototype and Components

Four Project Components

- Sqlite Database
- Electron Frontend
- Pi Images and Network Configuration
- Hardware



Prototype Status

- Components tested
- Ready to link together for system prototype

Status of the Project

CAN Team

Pythons scripts for data harvesting has been completed, had been working with VCAN data

- All hardware has now been received, moving into integrating real CAN data and sending it to our Back End

Networking Team

Researched and began working with OSLR protocol for routing, configured Pis to communicate via Ad Hoc network

- Connections between Pi have all been established and are working
- Further testing with OSLR is required and will be the next step

Status of the Project

Back-end Team

Database schema is up and running remotely, initial tests have been successful

- Database needs to be loaded up onto all of the Pis
- API design has been laid out and implementation will be the next step

Front End Team

Base UI was created and evaluated by the client, feedback was recorded and taken in for consideration

- UI was rebuilt in Electron and implemented Push Notifications
- Data visualization options were looked into, displaying JSON data in the desired format are is the next step

Responsibilities Next Semester

Will Paul and Cody Lakin- Finish, test, and document CAN bus to database operations

Ryker Tharp- Assist Data Visualization, Database Management, Data Analytics

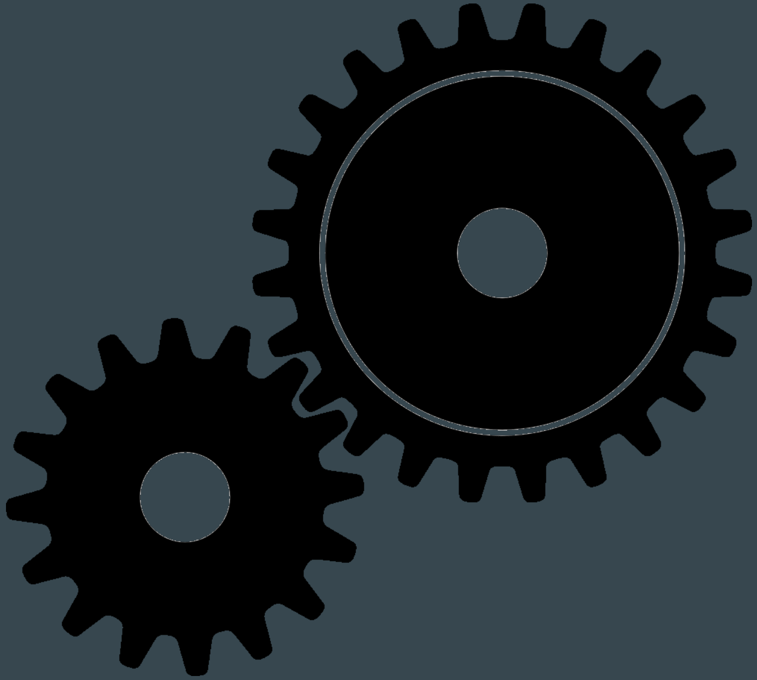
Colton Smith- NodeJS API and testing and Data Analytics

Collin Vincent- Managing network node image and structure

Gage Tenold- Implement data visualization using JSON from devices, Finish UI

Objectives for Next Semester

- Link all components together
- Prototype
- Implement data analytics
- Range improvement
- Get project in hand-off condition



Questions,
Comments,
Concerns?

References

Websites Referenced-

https://www.fleet-genius.com/
https://zubie.com/

Stock Images Used-

https://pixabay.com/en/crane-machine-heavy-equipment-158339/
https://www.iconspng.com/image/94360/isometric-bulldozer
https://www.iconspng.com/image/100913/torex-dump-truck
https://www.freeiconspng.com/images/laptop-png
https://www.fleet-genius.com/wp-content/uploads/2016/06/VHMConnector.jpg
http://zubie.com/fleet/wp-content/uploads/sites/3/2015/07/zubiekey100054948orig500-620x354.png
https://medium.com/ibm-watson-data-lab/installing-web-apps-with-electron-7a8fa1b12744