ASHESI UNIVERSITY COLLEGE

VEHICLE CONDITION MONITORING

FOR FLEET MANAGEMENT

CAPSTONE PROJECT

B.Sc. Computer Engineering

Terry Selassie Tettey

2021

ASHESI UNIVERSITY COLLEGE

VEHICLE CONDITION MONITORING FOR FLEET MANAGEMENT

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Terry Selassie Tettey

2021

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

Candidate's Name:

Terry Tettey

Date:

27/04/2021

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgments

To my supervisor, Dr. Nathan Amanquah whose insight, guidance and encouragement aided me throughout this project.

Abstract

With the many advances in technology, the field of fleet management is becoming more relevant in everyday activities. The ability to keep track of service vehicles while they are on the move is very useful to as it helps businesses who operate such services to cut cost and monitor employees who are on the move. Currently, there are many variations of fleet management systems which depend solely on locational data from a GPS enabled device like a mobile device. There are also much more costly fleet management systems which can track the location of vehicles and do so much more. This project aims to provide a less costly fleet management solution which can track a vehicle's location, as well as observe data on a vehicle's on-board network for commercial use and for vehicle diagnosis.

Table of Contents

Chapter 1 - Introduction

1.1 Background

New technologies are being created, tested, and implemented every day at a pace that is more rapid than before. Looking at the current state of industry-grade machines and solutions, mobile phones, gadgets; it is very clear to see that technology is progressing quite quickly and is waiting for no man. This holds for the state of vehicle technologies as well. With all these new technological innovations being released in newer vehicle models, vehicular diagnostic methods of old which relied on taking apart a vehicle to determine a problematic or faulty area are quickly being thrown to the side and more convenient diagnosis methods and protocols are being instated in their place. These new vehicular diagnostic methods do not require taking apart some of the test vehicle's components. Instead, a peripheral device is plugged into the vehicle and all the vehicle's faults are scanned and logged. This approach greatly speeds up the diagnosis process and makes the work of automobile mechanics exponentially easier.

The technology that allows vehicles to diagnose themselves has been around for several years now. This is termed On-board Diagnostics (OBD). This has been implemented in most American and European automobiles since the early 1980s. Further revisions have been made since then and they have allowed for peripherals with data logging and analytical features to access the vehicle's data which consists of fault codes and sensor readings through a standardized port (usually found under the steering wheel of the vehicle). The newest revision of Onboard Diagnostics, OBD-II, has proved to be the industry standard for car diagnosis to date. Since 1996, all cars that were sold in the US were required to have an OBD-II port on their vehicle for diagnosis purposes.

OBD-II Scanners and Loggers can be quite costly but recent developments have brought about cheaper options to choose from. One of such options is thanks to the emerging field of Internet of Things (IoT); a dongle that plugs into a vehicle's OBD-II port and transmits data to a smartphone over Bluetooth or Wi-Fi. While these access technologies may be ideal for many smart car enthusiasts, most of these dongles have been commercialized in such a way that the data cannot be manipulated any further than the smartphone / smart device used, and often a particular application or software is required to use them. If such devices were designed with the idea of user customizability in mind, they would be a very ideal and less costly alternative for vehicular fleet nodes in a fleet management system.

The field of vehicular fleet management is also slowly growing in Ghana; tools are being produced to help track vehicles and manage their resources to keep operating costs low. Although the adoption rate for such solutions in the general market is quite low, for people looking to start a business such as a delivery service or a taxi service, a fleet management system would benefit them greatly. For this project, such a system would be designed using low cost and easily accessible components.

1.2 Problem Definition

Given the points raised in the Background section, the problem definition for this capstone project is that Delivery / Taxi Service Providers need a cheaper and more functional alternative to their proprietary fleet management system to manage vehicles on the move much more effectively.

A local car dealer may also be able to benefit from such a system especially in the context of predictive maintenance. In such a case, a vehicle dealer would need an affordable and easily accessible tool that would allow them to monitor their vehicles' performance and predict the next time one of their vehicles should be brought back for maintenance.

Incorporating the aspect of predictive maintenance in this project would help cut costs incurred from routine maintenance.

1.3 Objectives

Below are the objectives of this project.

- Design and implement a device that can take diagnostic data from a vehicle and send that data over an internet connection to a dashboard on a smart device or computer.
- Create the dashboard application to view and analyze the data coming from the device.
- Add GPS functionality to the device so that the location of the vehicle it is attached to can be tracked and logged on the dashboard application.

1.4 Requirements

Listed below are the requirements that this solution should meet.

- The device should successfully read data from a vehicle and present that information in a dashboard.
- The dashboard application must be able to show the location of the vehicle the device is attached to.
- The solution should be able to work with any vehicle the device can be attached to.
- The device should allow the user to identify the data coming from the vehicle.

1.5 Proposed Solution

The proposed solution is to create a peripheral device that can connect to an automobile's On-Board Diagnostics Port to retrieve vehicle sensor data and diagnostic information. The device should feature GPS functionality and must then be able to send data to another device via an IoT access technology for analysis and fault checking.

1.6 Scope

The scope of this project assumes that the device would be powered by the vehicle it is connected to. Implementation of security measures for safe networking between the device and the dashboard application would not be covered.

Chapter 2 - Literature Review

2.1 What is CAN?

One of the communication solutions which you can often find under most OBD-II specifications is the CAN Bus. Most of the sensors and actuators in a vehicle are categorized into Engine Control Units (ECUs). These ECUs can send data/information to each other through the CAN Bus. In the CAN Bus, signals/messages from one ECU are sent to all ECUs in the CAN Bus network. In the vehicle, data on the CAN Bus is transmitted through a differential voltage solution; There are two lines, CAN-H and CAN-L, carrying similar signals but opposite in polarity. This helps protect the signal traveling on the CAN Bus from significant sources of noise. To easily receive or transmit data on the CAN bus, a CAN Transceiver is needed. This changes the CAN-H and CAN-L differential voltage signals into CAN Serial data and vice-versa. A CAN controller can then be used as an interface in conjunction with a CAN transceiver to send frames/data packets over the bus.

The CAN Bus protocol dictates the structure of the CAN Frames / Data Packets being sent. The general format of a CAN Frame is as follows:

- The Start of Frame bit: - This bit indicates the beginning of a message on the CAN Bus.
- The Identifier bits: - These 11 bits determine the message's priority over other messages. The lower this identifier bit is, the more priority the message has.

- The Remote Transmission Request bit: - This bit is used to distinguish between a data frame and a remote frame. While data frames are the typical messages used to communicate over the CAN Bus, remote frames are used specifically to request data from other ECUs on the CAN Bus.
- The Identifier Extension: - This bit determines whether the message is in standard or extended format. The standard CAN message format can take up to 8 bytes of data whereas the extended CAN message format can take up to 64 bytes of data.
- The Data Length Code Bits: - These 4 bits give the number of bytes of the message being transmitted.
- The Data: - This is where the actual message data is being held. Standard CAN format allows up to 8 bytes to be transmitted in this part of the frame whereas Extended CAN format allows up to 64 bytes to be transmitted.
- The Cyclic Redundancy Check bits: - These 15 bits are used to detect any errors in the transmission of the message. The transmitting ECU calculates a checksum from the bits being transmitted and puts that checksum in this field of the CAN frame.
- The Acknowledgment bit: - This bit is altered by receiving ECUs/nodes when they successfully receive the message. If a receiving node detects an error in the received message, the message is discarded, and the sending node tries to send the message again. This bit has another bit that acts as a delimiter.
- The End of Frame bits: - These 7 bits show the end of the frame.
- The Interframe Space Bits: - These bits determine the minimum allowed space between CAN frames. It acts as a sort of intermission between CAN Frames where no node can transmit data.

- Out of all these parts of a CAN Frame, the parts which are of interest to this project are:
- The Identifier – Helps identify which node / ECU the message is coming from or is specifically addressed to.
- The Data Length Code – Helps us know the amount of data coming in.
- The Data –The important bytes to be analyzed.

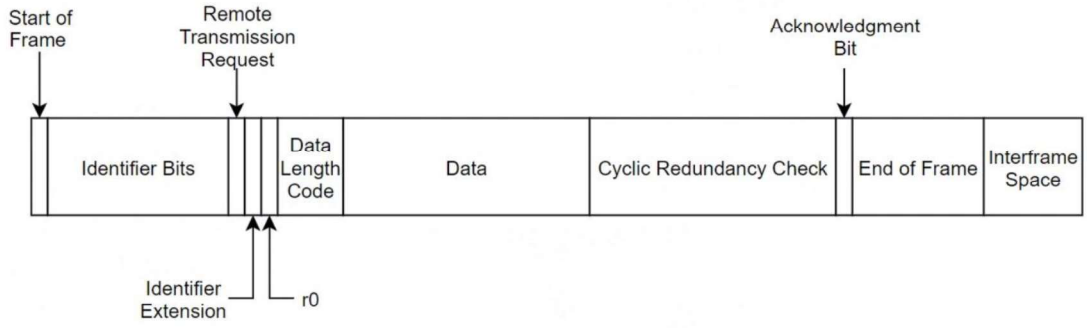


Figure 2.1: Standard CAN Frame

Because of the nature of CAN Frames and how their sizes may differ, a CAN Controller is usually needed to manage the Serial CAN data and provide the data through an interface that is available on most microcontrollers like Serial Peripheral Interface (SPI) or UART.Figure 2.1

2.2 Reference Review

While OBD ports are essential, it has been reported that OBD-II dongles often open the vehicle up to auto-attacks via mobile applications [1]. Tests on a somewhat small scale have shown that 50% of OBD-II dongles have poor security measures that make them easy to hack. Some other flaws in some vehicle security systems which are the responsibility of some car manufacturers and some car dealers are also covered. While the literary depth of this article may not be the best, it seems to be a good place to start looking for systems that need

improvement in the field of automotive mobile connectivity and the “Internet of Vehicles”. The scope of this project does not deal with security measures as it may incur additional costs.

Connecting to a vehicle’s CAN Bus through an OBD-II port requires a transceiver integrated circuit that converts the differential CAN-High and CAN-Low signals into digital CAN signals which can then be handled by most microcontrollers. Some ICs manage to do this and more, the most popular example being the ELM327. This is a PIC18F2480 microcontroller programmed to communicate with several protocols coming from the OBD-II port coming from a vehicle [2]. The ELM327 is a standalone solution that can translate most of the data coming through the CAN bus and send it over a wireless transmission solution like Wi-Fi or Bluetooth.

development of a vehicle monitoring and positioning system that uses values obtained from the vehicle through the OBD-II port as well as a GPS module has been reported in [3]. Such a setup would prove very beneficial for a fleet management system. This paper is very insightful because it highlights some of the prominent challenges that one would face if they tried to implement such a system or any IoT application using a vehicle's OBD-II port. Here instead of using ELM327 for interfacing, it seems that a solution similar in architecture to the AGV4000 Diagnostic tool was used. The paper did not go into much detail in this regard.

An onboard automotive diagnostic system that accounts for flaws/faults on a vehicle’s network-level physical layer is shown in [4]. The paper manages to give a brief review of the Controller Area Network (CAN) bus architecture used in most vehicles and highlights some of its shortcomings. The article may be dated but it shows some problems that may be experienced when working with older car models.

In this paper [5], the authors show their efforts towards building a PC-Based Vehicle Diagnosis System which is based on On-Board Diagnostics standards. The authors, in their efforts to generalize the different physical and link layers of the various protocols used in automobile diagnosis, created a Vehicle Communication Interface as well as some software that would help with protocol conversion. They also compare and analyze some of the protocols used in the On-Board Diagnostics system of most vehicles. They conclude that their solution runs faster and is much more informative than most hand-held automobile diagnostic tools. This paper is very insightful in the design process of such a system.

Although the GPS Module used in this project was quite straightforward to use, this website [6] gave some insights into the limitations of the module. The website also provided some libraries like TinyGPS++ which help parse the data the GPS module gives out. One noteworthy point about the GPS module is the antenna does not have enough of a range to be used indoors in most cases.

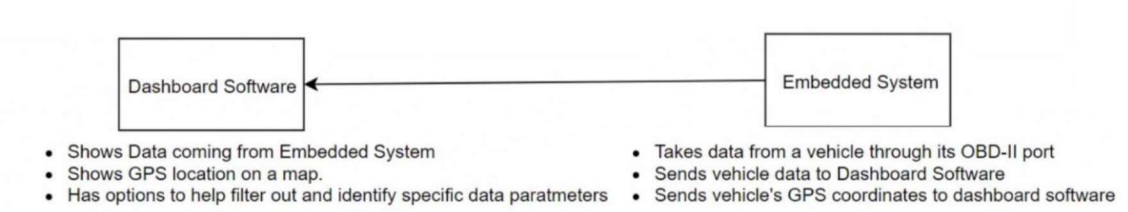
Chapter 3 - Requirement Specification

3.1 Introduction

This chapter shows the requirements that this project needs to meet for interested parties or customers to be satisfied with the product/solution. This is to serve as a guideline and checklist for criteria that need to be met.

3.2 Scope

The parts/components that make up this project are shown and explained below.



1. Software that runs on a mobile device and/or personal computer. The purpose of this software is to present the data acquired from the automobile in different analytical representations like tables and graphs. This would help inform the user of abnormalities in the automobile's systems during diagnosis. The software may also send requests to a vehicle for data through the embedded system discussed below.
2. An embedded system that connects the Controller Area Network (CAN) in an automobile to a device running the designed software through a wireless communication solution. This system is mainly concerned with handling communication between the device running the software and the automobile's CAN network. The system communicates with the automobile's CAN network through the automobile's CAN network. The system communicates with the device running the software through a wireless connection medium such as Wi-Fi, Cellular Network, or Bluetooth. This is because with a wireless connection, embedded system devices can be easily connected to several vehicles and all their data would be sent to the dashboard application.

3.3 Overall Description

3.3.1 Product Perspective

While there are already many different variations of tools that can be used in a similar way to this project's solution in automobile diagnosis and fleet management, an area where most of these variations fall short is in user customizability. The automobile diagnosis tools

and software usually comes as a package and often, the data is not presented in a form that a user can easily use for further processes.

The way this project aims to overcome such an issue is to allow users to have access to the information coming straight from the embedded system so that they can use it with their own projects; user's may even implement their own dashboard to view the data coming from the embedded system. The following block diagram shows the major components that would be worked on within the scope of this project.

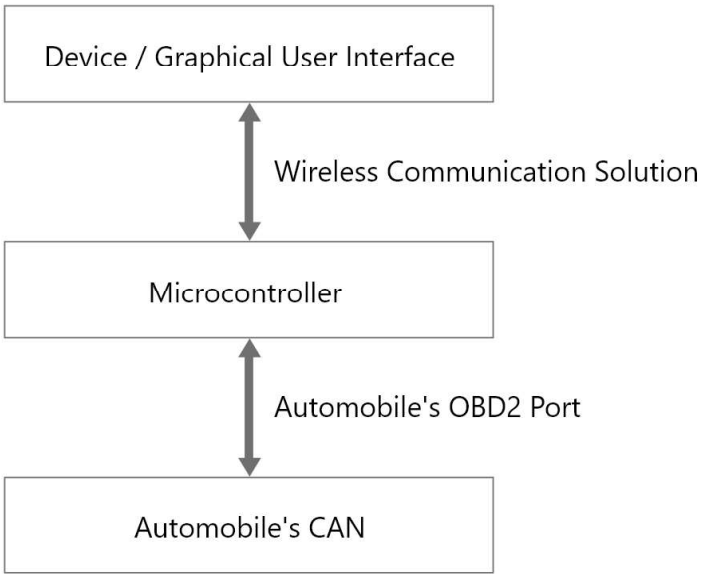


Figure 3.1: Block Diagram showing the relation between the major components in the project's scope.

3.3.2 Component Functions

The major functions of this project's products are stated below.

- Transfer CAN Frames between Automobile and Mobile Device / Personal Computer. The Controller Area Network (CAN) in the automobile has messages sent in frames. These frames can be picked up through the CAN Bus outlets on the OBD2 ports of most

automobiles. The major function of the microcontroller is to receive these CAN Frames from the automobile and send them to the mobile device / personal computer and vice versa.

- Present CAN Frame data in different forms for analysis. The main role of the mobile device / personal computer software is to interpret the CAN Frames and represent them in a graphical format that would help with further analysis / diagnosis. Graphical formats may include Cartesian Graphs, Pie Charts, Tables and more.

3.3.3 User Classes and Characteristics

Some user classes who are likely to use this solution are listed below along with their expected characteristics.

- Delivery / Taxi Service
    - They have experience dealing with Fleet Management Systems.
    - They know how to get data they want from the CAN Bus of a car.
    - They have web development experience for integrating the solution into their service.
- Automobile Mechanical Engineers
    - They have experience dealing with many automobile problems.
    - They have attained secondary or higher education in mechanical and / or electrical engineering.
    - They are “tech-savvy”. They must know how to use basic technology around them.
    - They must be able to refer to a user-manual when a problem arises, and they do not know how to solve it.
    - They must know how the Controller Area Network Bus works in a car.

- IoT Hobbyists
    - They must have intermediate programming experience. They must have an understanding of the blocks of code running on the microcontroller and how to debug them. This is because they would be filtering the CAN Frames for the ones they want directly on the microcontroller and send it exactly where they want without needing to use the software.
    - A background in electrical and / or computer engineering is not needed but is recommended.
    - Knowledge of the structure of CAN Frames as well as an idea of the data being sent in each CAN Frame must be understood.

3.3.4 Operating Environment

The product(s) would most likely be used in delivery service headquarters / automobile repair shops / garages. The microcontroller would be connected to the automobile and a web application would be used to interact with the data on a computer or an Android Device.

3.3.5 Design and Implementation Constraints

- Hardware Limitations: The processing power of the microcontroller may limit how much data can be sent between the automobile and the mobile device / personal computer per unit time.
- Language Requirements: The products as well as their documentation would be written in English so users would need to be able to understand English.
- Database Management: Because of possible data overflow, the CAN Frame data may need to be filtered before it is sent to a database either directly from the microcontroller or from the mobile device / personal computer running the web application.

3.3.6 Assumptions and Dependencies

- It is assumed that any user who adopts this solution has an understanding of the data being sent in an automobile's Controller Area Network and understand that different automobile manufacturers may use different parameter IDs for different data.

3.4 Specific Requirements

Microcontroller System Features

1. Requirement: The microcontroller would need to be able to work at a clock rate higher than the frequency of transmission to prevent aliasing.

   Input: Highs and Lows from CAN Bus.

   Output: CAN Frame Data

   Sampling / Translation of Physical Layer data from CAN Bus to CAN Frame Data.

   Priority: High

   The microcontroller would be able sample the data coming in from the CAN bus High and Low lines and translate this to CAN Frame Data.

2. Requirement: The microcontroller should be able to send data to and receive requests/instructions from the analysis device.

   Input: CAN Frames from microcontroller and instructions / requests from Analysis Device.

   Output: Data received on opposite end.

   Establish Connection between Microcontroller and Analysis Device

   Priority: High

   The microcontroller should be able to connect with the Analysis Device through a wireless communication protocol like Wi-Fi / Bluetooth / Cellular.

3. Requirement: The microcontroller should either have GPS Functionality or interface reliably with a GPS module.

   Input: GPS data from microcontroller or GPS module.

   Output: GPS data received and shown on the web application. Provide GPS Data

   Priority: High

   The microcontroller should be able to send GPS data over the chosen communication protocol.

Software Features

1. Requirement: Graphical Representation of CAN Frames with known IDs.

   Input: Button Tap or Press on Known ID / Description, Choosing Graphical Representation from a list box.

   Output: Chosen Graphical Representation is shown.

   Priority: High

   The CAN IDs would be shown in a list box. A few graphical representations would be available for CAN Frame Data that have known IDs. Some of these Graphical Representations are Cartesian Graphs against Time and / or against other ID values and Tabulation.

2. Requirement 1: The range of IDs shown during the capturing process should be customizable.

   Requirement 2: Values that change during the sniffing process should change the color of their highlighting.

   Input: Button Taps or Clicks

Output: List of Frames Show, List of Frames Saved (Captured), Changed values Highlighted.

Sniffing Mode Option

Priority: High

This mode would be used to figure out the CAN IDs of data that changes. This helps identify what a particular CAN ID is used for as different manufacturers may have different uses for different IDs. This would be an optional mode accessible by the click or tap of a button. A button would then be pressed to capture the data. A customizable range of IDs would be shown on the screen and anytime their value changes, the changed byte would be highlighted in a different color. If the value should revert to what it was when it was captured, the value would be unhighlighted.

User Interfaces

- The User Interface for the Software would be presented graphically instead of in a command line format to alleviate users' thoughts of complexity.
- The User Interface would be presented in English Language.
- The User Interface would have very conspicuous text for better ease of access.
- CAN Frames would be presented in a List box when sniffing.
- The User Interface would be created as a web application.
- The User Interface would allow logged data to be saved locally on the device.

Hardware Interfaces

- The microcontroller should have a wireless communication interface to communicate with the analysis device and / or other IoT devices.
- The microcontroller should have a 5V Power supply.

- The microcontroller should have a higher sampling / baud rate than the vehicle's CAN Bus.

Communications Interfaces

- The wireless communication protocol used to send data from the microcontroller to the analysis device and vice versa should allow for a range of about 50m.
- The wireless communication protocol used should have decent data transfer rates (around 5 MB/s).

3.5 Non-Functional Requirements

Performance Requirements

- The device / solution should be energy efficient. The embedded system connected to the car and the software interface should not use too much power.

Safety Requirements

- The embedded system node should be insulated to prevent liquid spills from short circuiting the board.
- Very high frequencies should be avoided since over exposure to them could lead to some health risks.
- There should be no loose wire connections.

Security Requirements

- The connection protocol used should prevent unauthorized parties from accessing the vehicle's data and sending instructions to it the car.
- Encryption may be used to obfuscate the data being transferred.

Chapter 4 - Design

4.1 Introduction

The project's solution is in two parts: The Hardware Section and the Software Section. In this chapter, the various components needed to fulfill this project's standards and the criteria by which decisions were made as to which hardware components would be the best choice are discussed. A circuit design using the chosen components is also shown below. The use case diagram for the software as well as the whole system would also be shown.

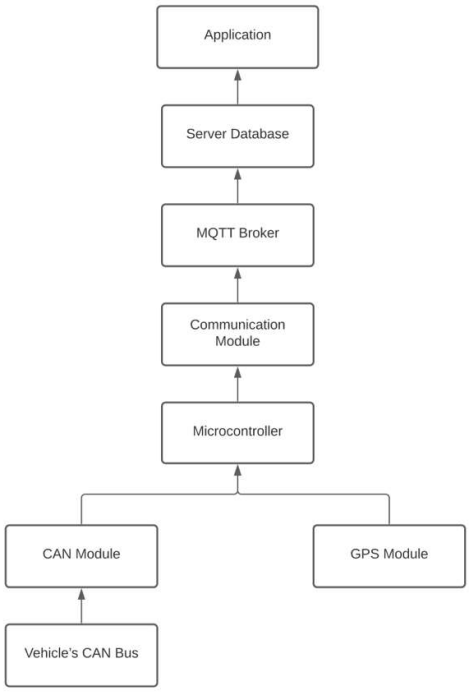


Figure 4.1: Diagram showing the components of the solution.

4.2 Hardware Component Selection

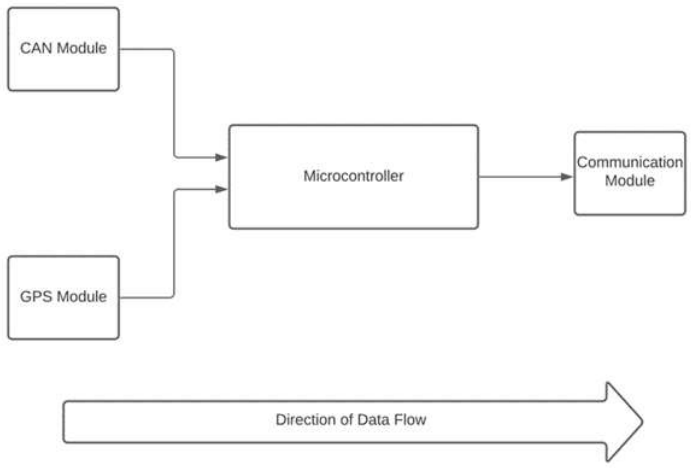


Figure 4.2: Diagram showing the components of the hardware system.

For the solution's hardware, the most important components are:

- The microcontroller: - This would be the unit that would be accumulating the data coming from the other hardware modules connected to it and sending it to the communication module.
- The communication module: - This is the unit that would be used to connect the hardware solution to the software solution.
- The CAN module: - This is the unit that would communicate with the vehicle's CAN Bus.
- The GPS module: - This module sends the data about the embedded system's location to the microcontroller.

A Pugh Chart was used to choose the microcontroller / development board to be used in this project. An ATMega328p, a NXP KL25Z Development Board and a DOIT ESP32 Devkit board. The boards were compared in the following categories: Cost, Accessibility, Ease of Code Modification, Processing Ability and Peripheral Interfaces. The ATMega328p is the

best selection according to the criteria. It is relatively low in cost and easily accessible as they can usually be found as the choice microcontroller for the Arduino Uno development board. It is very easy to modify the code on it thanks to the Arduino IDE and although it has lower processing power than the other choices, it is enough for the purposes outline in this project. It does have one UART interface, however, using the "SoftwareSerial" library included with the Arduino IDE package,

Another Pugh Chart was used to select a communication module that would be used to connect the microcontroller to the web application. A SIM800 GSM/GPRS module, a HC-05 Bluetooth Module and an ESP8266 Wi-Fi Module were compared against the following criteria: Cost, Ease of Use, Transmission speed and Communication Range.

The SIM800L is the best selection according to the criteria. Its very distinguishing feature is its wider range compared to the other the communication modules as it uses a cellular data solution. A wider range is desired as it would make it easier to for the system to be used while on the move. If the range is too small, there may be no need for the GPS module at all. The SIM800L may be lacking in transmission speed compared to the other modules as it is only capable of 2G cellular network interfacing, but a higher range is much more valuable for a fleet management system. The ESP8266 supported with a Wi-Fi Hotspot from a Cellular Device that supports 3G or 4G would be an excellent alternative to the SIM800L module however, this depends on the mobile device that it hotspots from.

For the CAN module selection process, it seemed that the most popular and easily accessible CAN Controller of choice was the MCP2515 CAN Controller. A CAN Transceiver is needed to bridge the connection between the vehicle and the CAN Controller. The CAN controller and transceiver can usually be found packaged together in an all-in-one module.

There is hardly any difference between the functionality of the TJA1050 and the MCP2551 CAN Transceiver modules other than the fact that it is easier to find the TJA1050 modules than it is to find the MCP2551 modules.

Not much of a selection process was done for the GPS Module. The UBlox Neo-6m GPS module was chosen because it was the easiest to find and there was enough documentation and guides online to help get it working correctly.

From the decisions made in the Pugh Charts, the selected components are:

- ATMega328p Microcontroller
- SIM800L GSM/GPRS Module
- MCP2515 + TJA1050 CAN Module
- UBlox Neo-6M GPS Module

4.3 Circuit Design

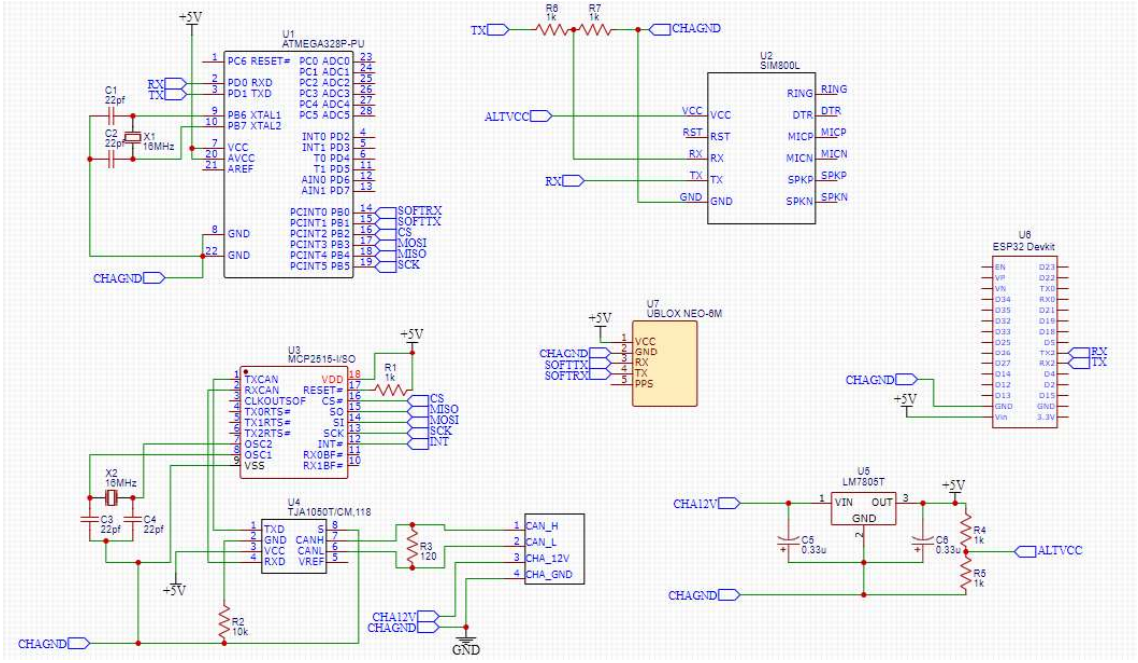


Figure 4.3: Circuit Schematic of Hardware Solution

A voltage regulator in the form of an LM7805 (located bottom-right) has also been added to the schematic to ensure voltage levels are at an acceptable level of 5V. The ATMega328P (top-left) connects to the MCP2515+TJA1050 CAN Module (bottom-left) through its SPI interface. The ATMega328P connects to the SIM800L module through its UART interface. Because a maximum of 4.4V and a minimum of 3.4V can be applied to the SIM800L, potential divider circuits have been implemented to cut down its operating voltage and the voltage going through its RXD pin to acceptable levels. The GPS Module (located in the middle) is connected to the ATMega328P through pins PD2 and PD3. They have been configured as "SoftwareSerial" pins with a baud rate of 9600. This is sufficient for the GPS Module to send data effectively. The DOIT ESP32 Devkit board (located at extreme right) can be used as a substitute if the SIM800L module does not work properly. It is connected to the ATMega328p with the same UART interface that the SIM800L module uses, however, the SIM800L and the DOIT ESP 32 Devkit board would never be connected at the same time as program conflicts may occur.

4.4 Software Solution Design

A web application has been settled on for the software application as it would be very easy to deploy on either a mobile device or a dedicated / personal computer. The web application would be created using tools in the MERN stack of web application development tools. For database management, MongoDB would be use. For server routing and the implementation of a simple RESTful API, Express.js would be used in conjunction with Node.js. The front end of the web application would be written with React. To receive data from the hardware / embedded system, an MQTT broker service would be needed. This is where the embedded system publishes its data initially. The web application server subscribes

to this MQTT broker and saves any data that happens to be published in its database. The data is then shown on the web application user interface.

A use case diagram showing how a user would interact with the web application and the system is shown below.

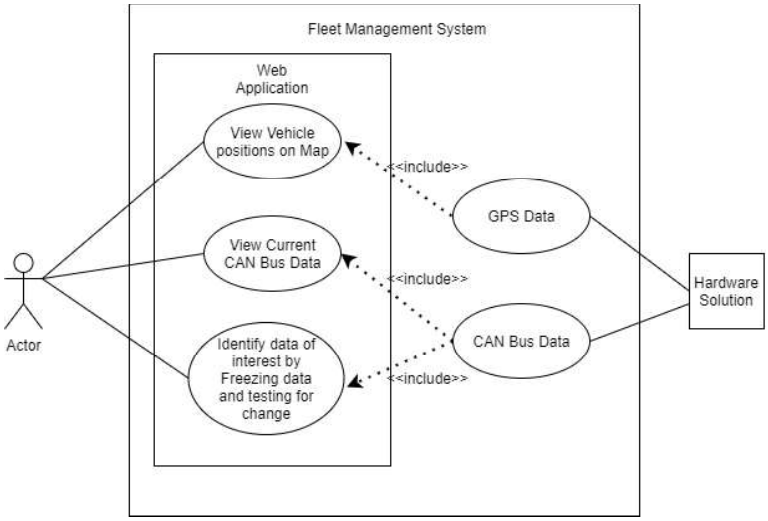


Figure 4.4: Use case diagram showing how a user would interact with the Fleet Management System.

Below is a simple flow chart showing the different components of the project and how they would work together.

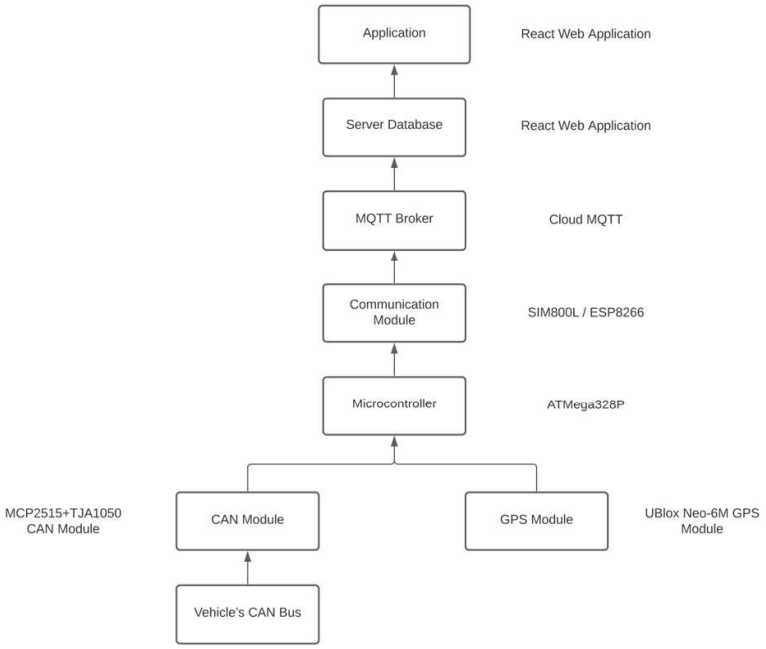


Figure 4.5 Flowchart of the Fleet Management System Solution

The Web application gets all its data from the MongoDB server database. The server database receives this data when the ATMega328P module sends an MQTT packet to the MQTT Broker with the help of the SIM800L module. The CAN Module is constantly collecting data from the vehicle's CAN Bus and presenting it to the ATMega328P to send. The GPS module also sends its latitudinal and longitudinal data to ATMega328P to parse and send to the server database.

Chapter 5 - Implementation

5.1 Testing Modules

A prototype was built to test each of the modules in isolation and together. The breadboard prototype is shown below.

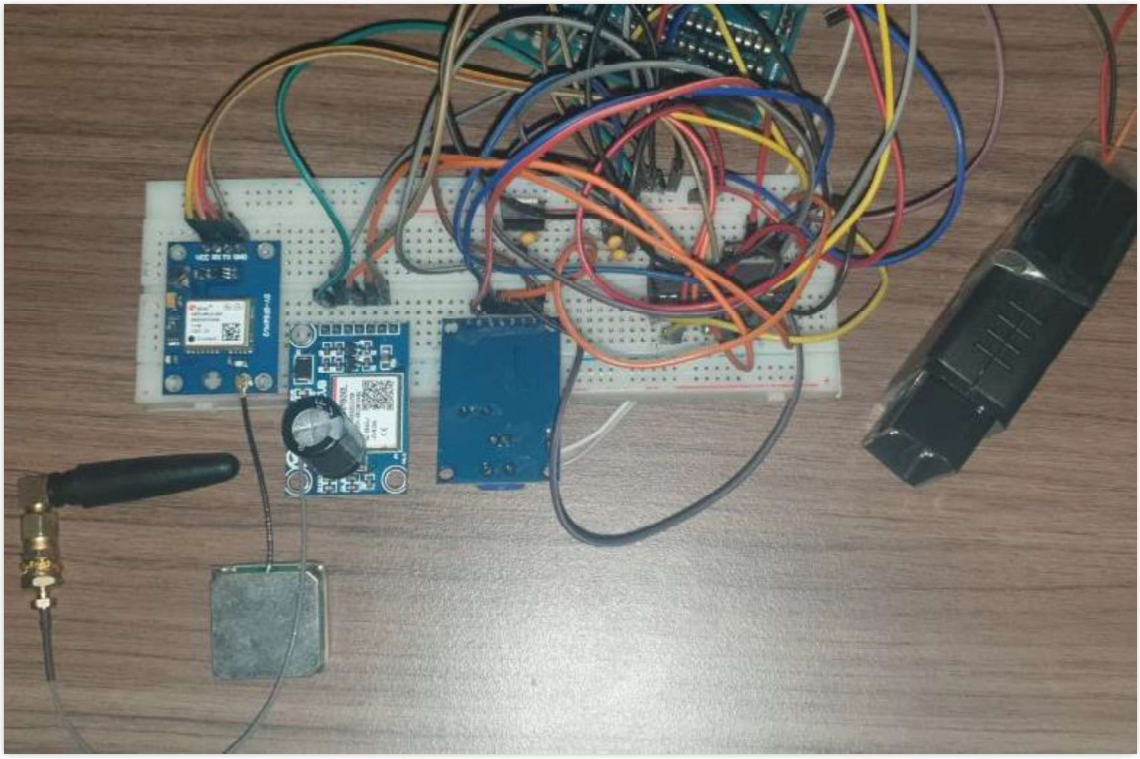


Figure 5.1: Hardware Prototype

The code for the individual modules was implemented in different cases of a switch statement to ensure that they are tested in isolation. Below are the findings for each of the modules.

GPS Module

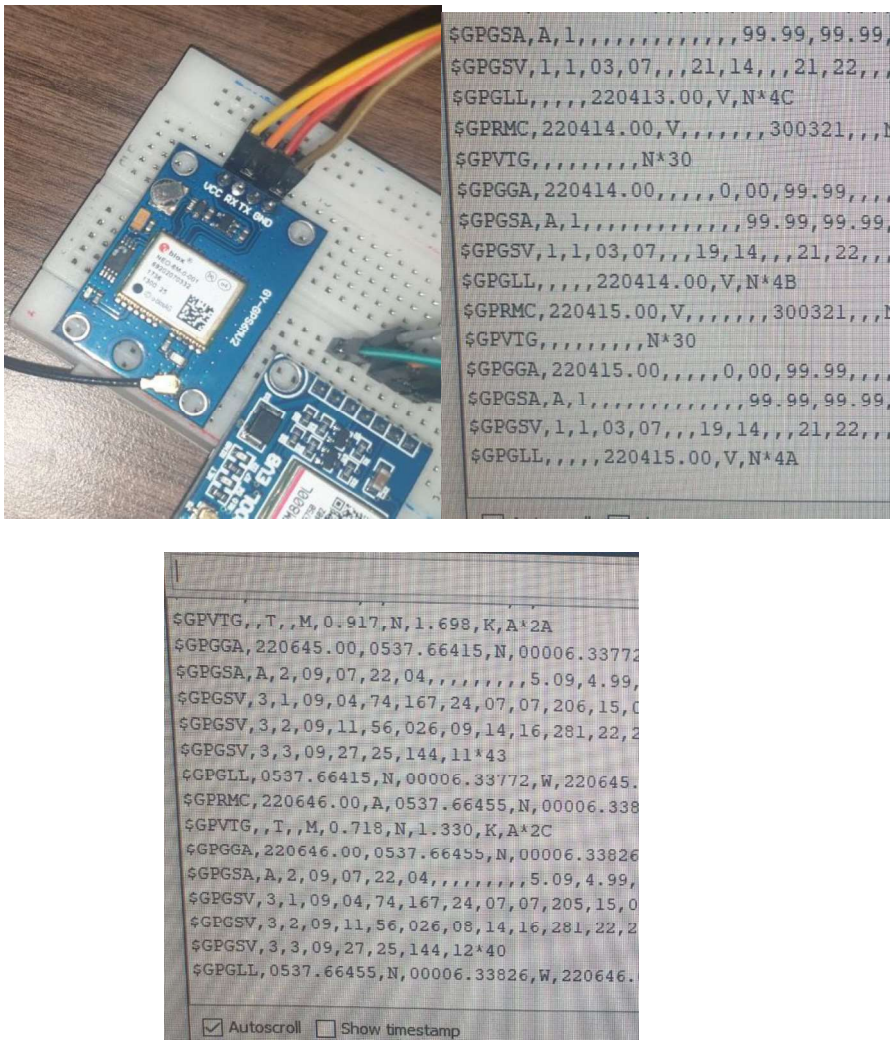


Figure 5.2: UBlox Neo-6m GPS Module (top-left), data at startup (top-right), data after startup (bottom).

- The GPS Module takes a while to find the nearest satellites and finish its triangulation process but when it is done, it works fine.

CAN Module

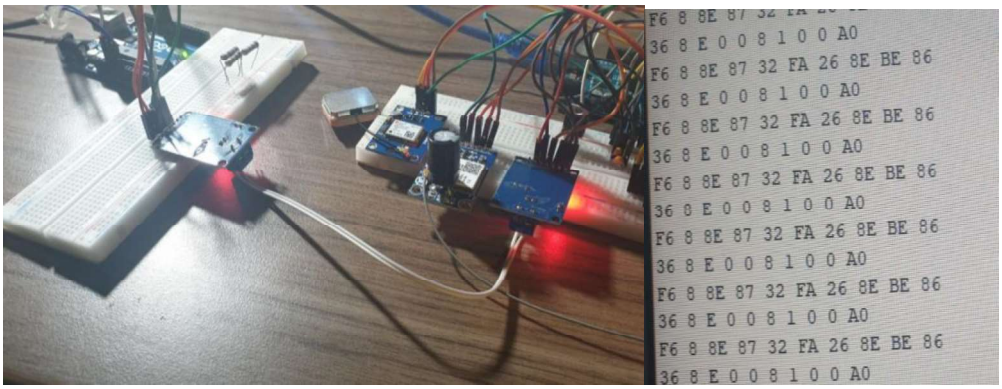


Figure 5.3: Two CAN modules sending data between each other (left), dummy CAN data being received (right).

- The CAN Module worked fine when it was connected to another CAN module which sent dummy data to it. It successfully received the dummy data.
- Tests were also carried out in a car. The car sent data for a few seconds and stopped right after. This occurred on many attempts. An error flag that is not being monitored may have been raised on the module.

GSM/GPRS Module

Figure 5.4: SIM800L Module before changing the capacitor (left) and after changing the capacitor (right)

- The SIM800L Module has a faulty problem of resetting every few seconds. Solutions found online have alluded to the fact that the problem occurs because the onboard capacitor that aids the antenna has too low of a capacitance needed for proper operation.
- Upon changing the 100uf capacitor on-board into a 1000uf capacitor, the problem does not seem to have been solved, however, the resets are less frequent. The image below shows AT commands that were being sent to try and test that it was working fine.

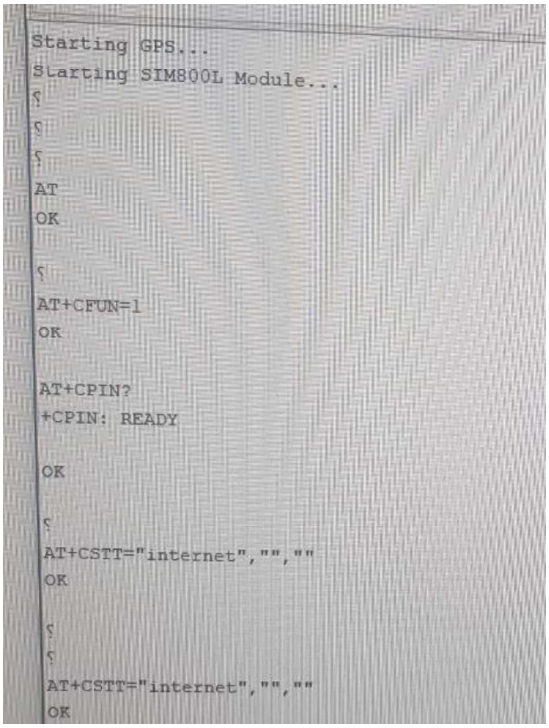


Figure 13: AT Commands being sent to SIM800L Module. Every backwards question mark symbol shows when the module resets itself.

5.2 Prototype Creation

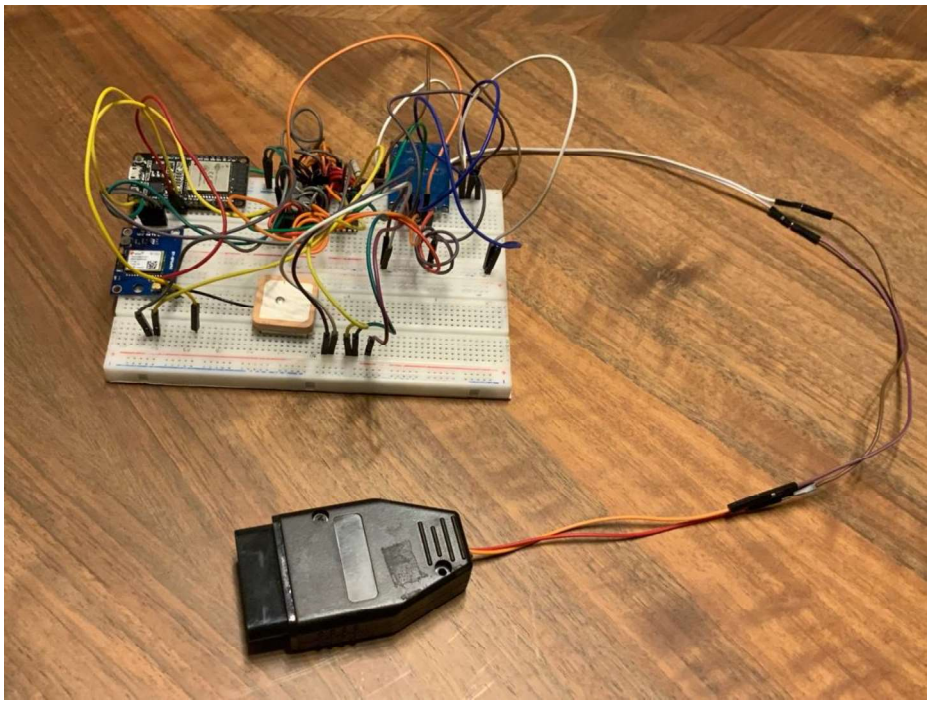


Figure 5.5: Image of prototype

The prototype design is shown above. A fix could not be found for all the SIM800L modules tested so the communication module was switched to a Wi-Fi module. The ESP8266 would have been a fine candidate for the prototype but the process of flashing it to make it operate properly was not successful. Because of that, A DOIT ESP32 Devkit board was used as the Wi-Fi module as they both share similar functionality. With this change, it became easier to send the GPS and CAN Data over to the MQTT Broker. The Wi-Fi module could not be implemented as a substitute communication solution because the ATMega328P does not have enough UART interfaces to efficiently support this. Changing another set of pins with the “SoftwareSerial” library to emulate the UART functionality tends to shift all the “SoftwareSerial” pins out of sync.

The prototype runs for a few seconds when connected to a vehicle's CAN bus and stops abruptly. The reason for this is still unknown. Most of the CAN data obtained from the vehicle were in extended CAN format which is out of this project's scope.

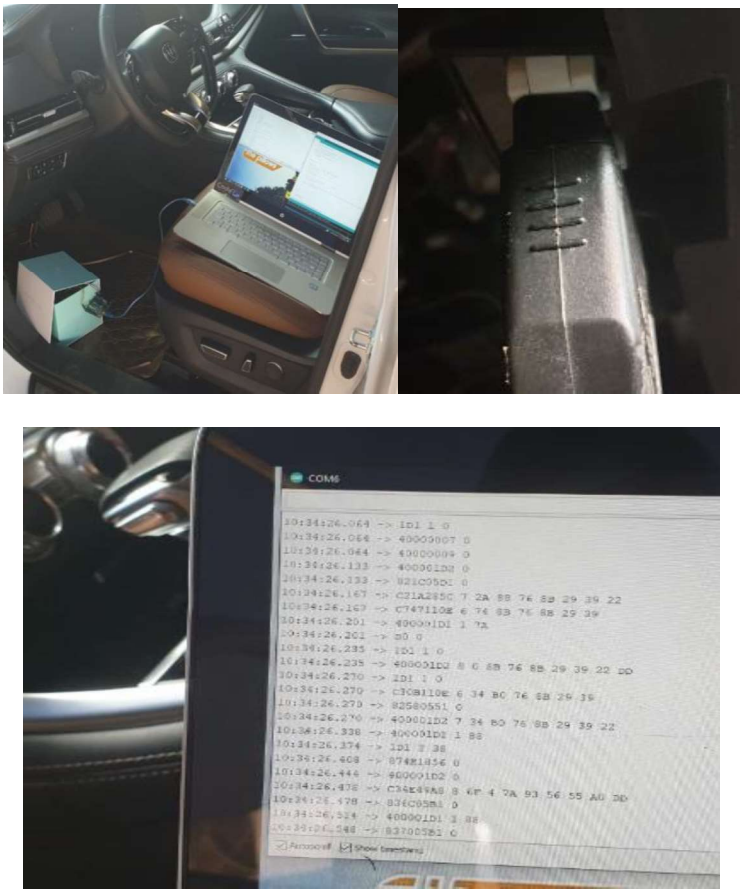


Figure 5.6: Reading CAN Values from Vehicle (top-left), OBD-2 Connector connected to Vehicle's OBD-2 Port (top-right), CAN being received (bottom).

Because of this, dependence on an Arduino board simulating a car's CAN Bus has been used to test the data flow. The figure below shows the substitute CAN bus data being received on the CloudMQTT broker user interface.

Received messages

Topic	Message
CAN/072A	0E 00 00 00 00 00 00 00
CAN/00F6	8E 87 32 FA 26 8E BE 86
CAN/0036	0E 00 00 08 01 00 00 A0
CAN/00F6	8E 87 32 FA 26 8E BE 86
CAN/0036	0E 00 00 08 01 00 00 A0
CAN/00F6	8E 87 32 FA 26 8E BE 86
CAN/0036	0E 00 00 08 01 00 00 A0

Figure 5.7: Showing Substitute Data being received on CloudMQTT's user interface

The picture below shows the demo web application user interface. The GPS data from the MQTT broker updates the position of the marker on the map shown in the image. The CAN data is shown listed on the web application interface as well.

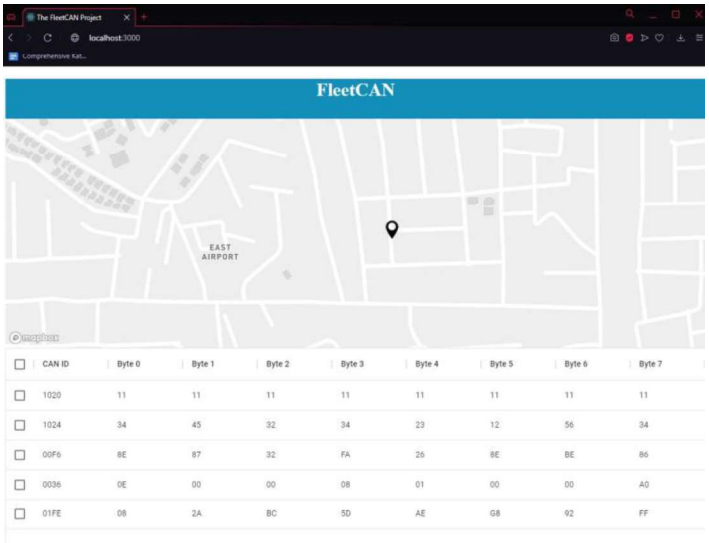


Figure 5.8: Web application showing the CAN data in a listbox and the GPS location on a map.

Chapter 6 - Conclusion

The prototype is unfinished. It does not meet some of the requirements defined in the requirements chapter. The embedded system hardware can connect to vehicles however, it freezes after collecting a few values. It is suspected that this is due to some miscalculation in the configuration process of the CAN module. The web application cannot freeze the CAN data which would help identify what particular IDs are responsible for. In retrospect, most of the time in this project was spent debugging the communication module (SIM800L) as it was not working as expected. Time would have been better spent if limits were placed on time sinks like the debugging process. If that were the case, other requirements for this project could have been met.

Despite these hindrances to the project, the transfer of simulated data from the embedded system to the web application was successful. If the car successfully connects to the embedded system and the web application is fully developed, this would prove to be a very cost-effective fleet management solution.

References

[1] W. Yan, “A two-year survey on security challenges in the automotive threat landscape,” in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*, Shenzhen, China, 2015.

[2] M. E. Marin, M. Maricaru, F. Constantinescu and A. G. Gheorghe, “Hardware and software approach for teaching automotive networks.,” in *2017 Electric Vehicles International Conference (EV)*, 2017.

[3] E. Ceuca, A. Tulbure, A. Taut, O. Pop and I. Farkas, “Embedded system for remote monitoring of OBD bus.,” in *Proceedings of the 36th International Spring Seminar on Electronics Technology*, 2013.

[4] J. Suwatthikul, R. McMurran, and R. P. Jones, “Automotive network diagnostic systems,” *International Symposium on Industrial Embedded Systems,* pp. 1-4, 2006.

[5] J. Hu, F. Yan, J. Tian, P. Wang, and K. Cao, “Developing PC-based automobile diagnostic system based on OBD system,” in *2010 Asia-Pacific Power and Energy Engineering Conference*, 2010.

[6] “In-Depth: Interface ublox NEO-6M GPS Module with Arduino,” Last Minute Engineers, 2021. [Online]. Available: https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/. [Accessed 20 April 2021].