Embedded Senior Project Report (Draft)

Zak Rowland

Oregon Institute of Technology

Author Notes:

Submitted in partial fulfillment of the requirements of Embedded Senior Project

(CST 471)

Revision History			
Revision	**Date**	**Reason**	**Name**
0.1	November 19th, 2020	Published draft	Zak Rowland

Abstract

The project is a Bluetooth diagnostic tool that interfaces with the OBD-II port in a modern vehicle. This system will consist of two major modules: the Bluetooth transceiver module that interfaces with the port, and the handheld module that includes a computer, Bluetooth transceiver, touchscreen, and rechargeable battery. Features of the system will be limited to the essentials to keep operation simple and costs low. The user will be able to read a list of diagnostic (trouble) codes and their descriptions as well as clear the codes. There will be another mode that will read sensor data and display it to the user including speed, oil pressure, and coolant temperature. A simple user interface will utilize touchscreen input for actions like selection, scrolling up and down, and changing modes or settings.

Table of Contents

Project Management

Schedule

Trello is the software used to schedule and manage progress on the project. The board is divided into five sections: Fall, Winter, Spring, In Progress, and Done, and each term has its own color to differentiate tasks in the In Progress and Done lists. Each task for a term has a due date set to maintain steady progress. If a task has multiple parts, a checklist can be added underneath the main task. The Trello board can be seen in Figure 1 below.
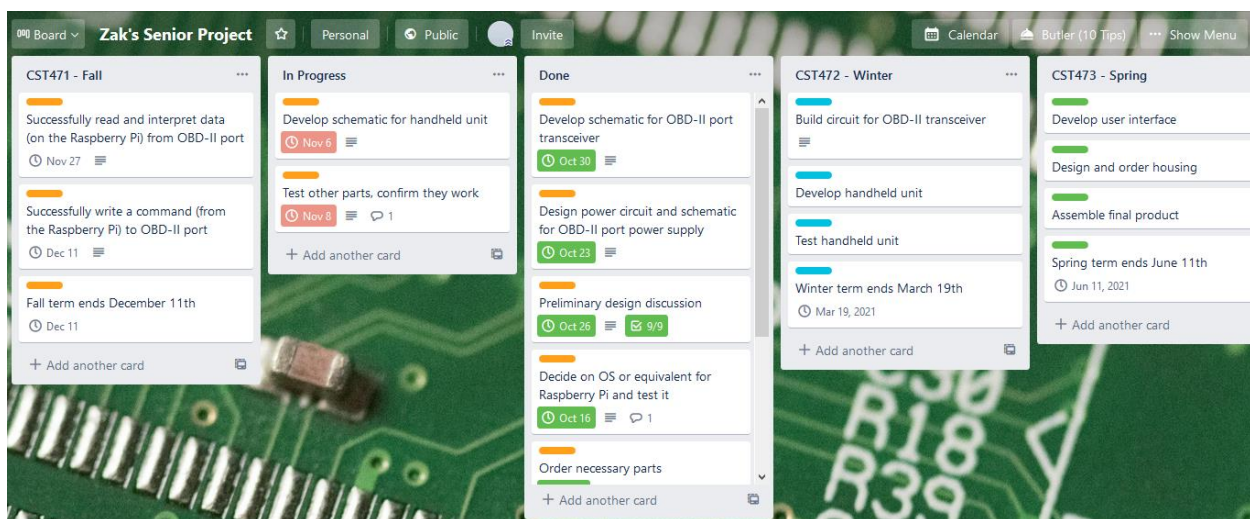

Figure 1. A screenshot of the _Trello_ board used to track progress on the project [1].

Estimated Non-Recurring Engineering Costs

Description	Cost	Total
Entry level computer engineer salary	$70,000	9-months for $52,500
(2) Breadboards	$2.50 each	$5.00
5V/3A+ USB-C power supply	$10.00	$10.00

Multimeter (Fluke 117)	$180.00	$180.00
Oscilloscope (Rigol DS1054Z)	$349.00	$349.00
Micro HDMI to HDMI cable	$9.00	$9.00
		$53,053

Table 1. Estimated non-recurring engineering costs.

Estimated Production Costs

Description	**Cost**	**Total**
Raspberry Pi 4 Model B 2GB	$35.00	$35.00
ELM Electronics ELM327 IC	$22.13	$22.13
OSOYOO 5" DSI display	$43.88	$43.88
Samsung 32GB MicroSD card	$7.99	$7.99
ESP32-DevKitC-32D microcontroller	$9.99	$9.99
MCP2551 (CAN transceiver)	$1.09	$1.09
4MHz crystal oscillator	$0.50	$0.50
L7805 regulator	$0.50	$0.50
317L regulator	$1.00	$1.00
Male J1962 Type A connector	$2.00	$2.00
Rechargeable battery	$29.50	$29.50
Battery charger and protection	$5.95	$5.95
Assorted resistors	$3.00	$3.00
Assorted capacitors	$3.00	$3.00
Assorted diodes	$2.00	$2.00

Assorted transistors	$4.00	$4.00
		$171.53

Table 2. Estimated production costs.

Change Management Procedures

If a change must happen in the requirements or design of the system, engineering change requests will be submitted to the project supervisor as well as a memo if necessary.

Status Reports

The status of the project will be updated to management or the customer with memos, demonstrations, or meetings.

Skills and Knowledge Areas

This project will require knowledge of many hardware components and protocols. Information about the OBD-II port and the CAN Bus protocol is necessary for communicating with the vehicle. Bluetooth will be used to send and receive data to and from the OBD-II port, so knowledge of the protocol is needed. The handheld module utilizes a touchscreen user interface which means GUI development is required as well. Finally, schematic capture experience is needed to develop schematics for the two modules, including power circuitry. PCB layout and 3D printing experience may also be required.

Conceptual Overview

Introduction

This project is an open source, handheld Bluetooth OBD-II diagnostic tool for diagnosing and repairing vehicles. The system consists of two primary modules: the OBD-II port module and the handheld touchscreen module, both of which contain Bluetooth transceivers for communication. Costs will be kept low to make the project accessible to as many people as possible.

Problem Statement

There is no standalone, Bluetooth, and handheld system with the required basic features commercially available at a reasonable cost (<$250.) The existing projects that are similar do not utilize a handheld touchscreen device or a custom Bluetooth OBD-II transceiver circuit.

Intended Audience

This project is targeted primarily towards home mechanics who wish to perform their own automotive repairs. However, paid technicians at a professional shop, or even electronics hobbyists and engineers, could benefit from a low cost and open source tool to interface with a vehicle wirelessly.

Proposed Project

The project is a Bluetooth OBD-II diagnostic tool for diagnosing and repairing vehicles. The system consists of two primary modules: a Bluetooth transceiver that interfaces with the OBD-II port, and a handheld rechargeable unit

that displays information on a touchscreen and receives input from the user. The

handheld unit's default mode will be scanning for and clearing diagnostic (trouble)

codes. However, there will be an option to switch modes and read data such as

speed, coolant, temperature, and oil pressure.

Relation of Proposed System to Existing Systems

While commercial products similar to this do already exist, there is no

reasonably priced solution (<$250) that provides the required features, such as

having a rechargeable handheld unit. The solutions that do exist, like the one

pictured in Figure 2 below, can easily cost over $500, and are bloated with

unnecessary features.

Figure 2. An example of an existing solution that is high cost from Amazon [2].

These products are targeted more towards professional mechanics and

technicians who work on multiple vehicles a day, making the investment justified.

The low-cost options, like the one pictured in Figure 3 below, all connect to a

proprietary phone app which does not meet the requirements of having a

standalone handheld device.

Figure 3. An example of an existing solution that is low cost from Amazon [3].

Many people do not like installing apps on their phone from small, unknown

companies. Phones may also be needed for other things while working on a vehicle

such as a making phone calls or texts, playing music, looking up information and

manuals, or even using the flash for a flashlight. Another problem with these cheap

products is the high probability that the OBD-II interpreter integrated circuit used

is a clone of the real one. The cloned chips are prone to malfunctions and sometimes

have a limited command set compared to the genuine chips. This project is different

from these existing commercial solutions because it will have a dedicated

touchscreen handheld device that is rechargeable, a genuine OBD-II interpreter

integrated circuit from the manufacturer will be used, all documentation and code

will be open source, and costs will be kept as low as possible while retaining the

necessary functionality.

There are projects completed by other students and engineers that are

similar to this one. One example, called "OBD-Pi," was done in 2014. However, this

solution used a prebuilt OBD-II Bluetooth reader and an aftermarket stereo head

unit to display the data. [4] The proposed project will include a custom OBD-II

interfacing circuit and a handheld touchscreen device. Another example is a project

made for a 1997 BMW. Like this first example, this solution uses a prebuilt OBD-II

reader that also utilizes a different interpreter chip (STN1110) [5]. A Raspberry Pi

is also used in this project, but the touchscreen interface and the Pi itself are

embedded permanently into the dash of the car. The proposed solution differs from

the existing projects because the OBD-II circuit will be designed from scratch using

a genuine ELM327 IC, the touchscreen interface is portable and rechargeable, and

all documentation and code will be open source.

Deliverables

All source code and documentation will be delivered to program director

Kevin Pintong as well as made open source. The components used to build the

project will remain in Zak Rowland's possession.
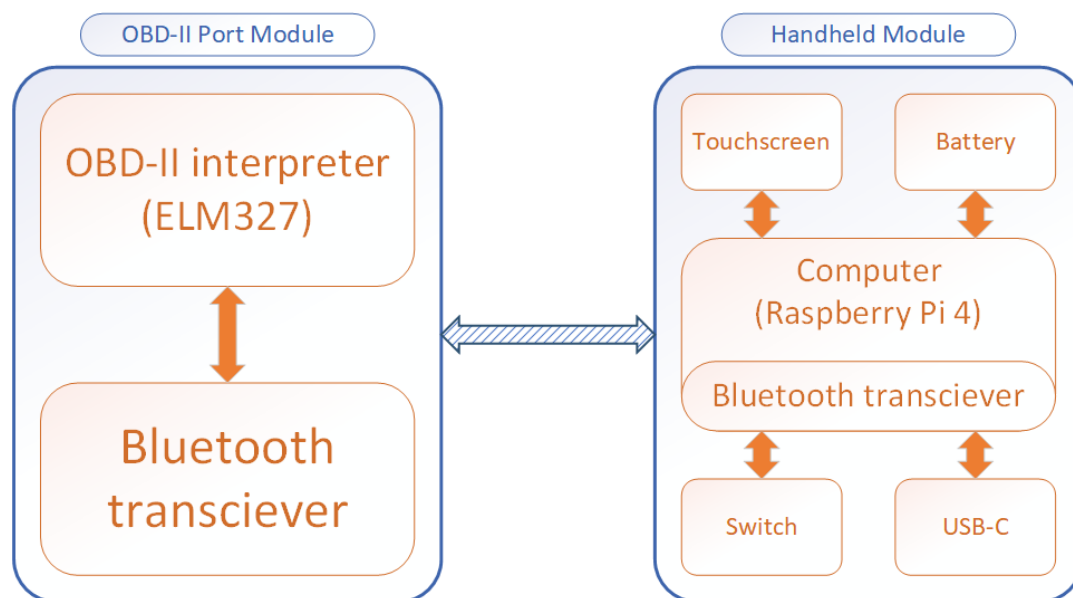
System Description

System Block Diagram

Figure 4. A block diagram of the system.

System General Description

Upon power up, the system's default mode is to scan diagnostic codes. To accomplish this, the circuit connected to the OBD-II must read all the trouble codes and send that information to the handheld unit through a Bluetooth connection. The computer inside the handheld device must process this information and drive a display for the user to see. Input from the user will occur through the touchscreen. A list of codes, or a message indicating the lack thereof, will be displayed to the user and they will be able to tap on a code to view the description and/or the possible cause. The user will have the option to clear all the diagnostic codes which involves sending a command over the Bluetooth connection so the OBD-II can respond accordingly. There will also be an alternate mode for viewing sensor information live (at least 30 times per second,) and a settings menu to adjust various options like brightness or units.

Major Subsystems

The major subsystems of the project are the power circuitry for the OBD-II module, the Bluetooth circuitry for the OBD-II module, the rechargeable battery circuitry for the handheld module, and the user interface for the handheld module.

Hardware Platform Description

A Raspberry Pi 4 will be used as the computer inside the handheld unit. The Pi is a good choice for this project for several reasons. It can run popular operating systems which makes user interface development much easier. There is plenty of memory, storage, I/O, and even documentation, and the Pi 4 has a DSI connection

that makes adding a touchscreen trivial. An ESP32 microcontroller will act as the

Bluetooth transceiver for the OBD-II module. This microcontroller supports

Bluetooth version 4.2, has several GPIO pins, and will make debugging the OBD-II

module much easier.

Software Platform Description

For the Raspberry Pi, scripts will be written in Python and any other

software, if necessary, will be written in C. The code for the ESP32 microcontroller

will be written in C. Python scripts will be written in an IDE or text editor such as

Spyder or Notepad++, and C code will be written in an IDE such as Visual Studio or

Atmel Studio.

Component Comparison and Selection

This project was initially intended to be as low cost as possible. However, the

COVID-19 pandemic has caused limited access to lab equipment. To keep the

project achievable during such uncertain times, some parts may be chosen primarily

due to their popularity and documentation. Table 3 below compares different single

board computer options for the project. The Raspberry Pi 4 Model B was chosen due

to its popularity, ease of use, and built in DSI connection for a display. However, the

Raspberry Pi Zero W would be the ideal computer for the project due to its smaller

size and lower power consumption.

Computer for Handheld Unit

Table 3 below compares different single board computer options for the

project. The Raspberry Pi 4 Model B was chosen due to its popularity, ease of use,

and built in DSI connection for a display. However, the Raspberry Pi Zero W would

be the ideal computer for the project due to its smaller size and lower power

consumption.

Name	Raspberry Pi 4 Model B	Raspberry Pi Zero W	BananaPi-M3
Price	$35	$10	$43
Processor	Broadcom BCM2711 (Quad core Cortex-A72 ARM v8), 64-bit, 1.5GHz	Broadcom BCM2835 (Single core ARM1176JZFS), 1GHz	Realtek RTD1395 (Quad core Cortex-A53), 64-bit
Storage	MicroSD	MicroSD	8GB eMMC + MicroSD
Memory	2GB LPDDR4-3200 SDRAM	512MB	1GB DDR4
Wireless	Wi-Fi and Bluetooth 5.0	Wi-Fi and Bluetooth 4.1	Wi-Fi and Bluetooth 4.2
I/O	(2) USB 3.0, (2) USB 2.0, (2) Micro HDMI, DSI, CSI, 3.5mm A/V, Ethernet	Mini HDMI, USB OTG, CSI, Composite video	(4) USB 2.0, M.2, 3.5mm, HDMI, Ethernet
GPIO	40-pin	40-pin	28-pin
USB/Power	USB-C, 5V/3A+	Micro-USB, 5V/2A+	USB-C, 5V/2A+
Other	OpenGL ES 3.0 graphics		OpenGL ES 1.1/2.0 graphics

Table 3. A comparison table for single board computers.

Table 4 below compares two different OBD-II interpreter integrated circuits,

the ELM327 and the STN1110. The ELM327 was chosen for the project. This chip is

uncontestably the most popular when making OBD-II readers. The documentation

is superb and there are many usage examples to reference, and it was chosen

primarily for this reason. The STN1110 is almost an exact replica of the ELM327, it

is even completely compatible with all commands used for the ELM chip. However,

the STN1110 is half the price and includes an extended command set for even more

available features. Since the STN1110 is not documented as well as the ELM327

and access to lab equipment is restricted, it is more sensible to use the ELM chip.

OBD-II to UART Interpreter

Name	ELM327	STN1110
Price	$21	$10
Operating voltage	4.5 – 5.5VDC	3 – 3.6VDC
Operating current	12mA	63mA
Power saver mode	0.15mA	<2mA
Operating power	60mW at 5V	207.9mW at 3.3V
Power saving power	0.75mW at 5V	6.6mW at 3.3V
Features	- Power control with standby mode - Serial interface - Automatically searches for correct protocol - Fully configurable with AT commands - Low power CMOS design - Very popular and well documented	- Fully compatible with the ELM327 (AT) command set - Extended (ST) command set - UART interface - Automatically searches for correct protocol - Large memory buffer (more RAM) - Voltage input for battery monitoring
Cons	- Expensive - Many buggy fakes/clones if not purchased from manufacturer	- Less documentation and examples - Uses more power

Table 4. A comparison table for OBD-II interpreters.

Bluetooth Transceiver

Table 5 below compares different Bluetooth modules for potential use in the project. The HM-19 or HM-10 module was going to be used originally, however the ESP32 microcontroller will be used instead. The ESP32-DevKitC-32D uses Bluetooth version 4.2, costs about the same as the modules below, and provides more functionality for debugging.

Name	HC-05	HC-06	HM-19
Price	$7.99	$7.39	$9.99
Bluetooth version	2.0	2.0	5.0
Operating voltage	3.6 – 6VDC	3.3VDC	3.6 – 6VDC
Logic voltage	3.3V	3.3V	3.3V
Modes	Master/slave	Slave only	Master/slave

Table 5. A comparison table for Bluetooth transceiver modules.

Fall Term Progress

Work completed so far has primarily been research and documentation. Notes, calculations, drawings, and time spent are all tracked in a spiral notebook as seen in Figure 5 below.

DATE	DESCRIPTION	TIME
9/27/20	Revising documents	3 hours
10/1/20	Researching parts	2 hours
10/2/20	Meeting 1 with Kevin	15 minutes
10/3/20	Building parts list	3 hours
10/3/20	Revising work deliverables	1 hour
10/9/20	Researching and ordering a screen, ELM327	2 hours

10/9/2020 continued:

I am going to wait until I start working on schematics before I decide on a Bluetooth module. This will be fine because I still need more parts (various circuit components.) Working on memo 1 now.

Figure 5. An example of the notes and time logs from the notebook.

The first difficult decision was choosing a Bluetooth module. The class was recommended to use the HC-05 module. However, that module uses Bluetooth 2.0, and the project requirements require at least 4.0. While deciding on a Bluetooth module, parts that were already chosen including the ELM327 chip, a 32GB MicroSD card, and a touchscreen were purchased.

Still undecided on Bluetooth parts, the Raspberry Pi was loaded with an operating system, Raspberry Pi OS, to test it. This operating system was chosen because it is the one recommended by the manufacturer, but it would be simple to change to a different one if necessary. An undervoltage warning appeared

intermittently while the Pi was in use, and it was discovered that the Pi prefers

closer to 5.1V/3A instead of 5V/3A. The undervoltage will not be an issue because

the rechargeable battery circuit will use a regulator that can supply 5.1V. After

testing the various applications included with the operating system, a very simple

Python GUI tutorial was completed. The tutorial was completed to get a feel for the

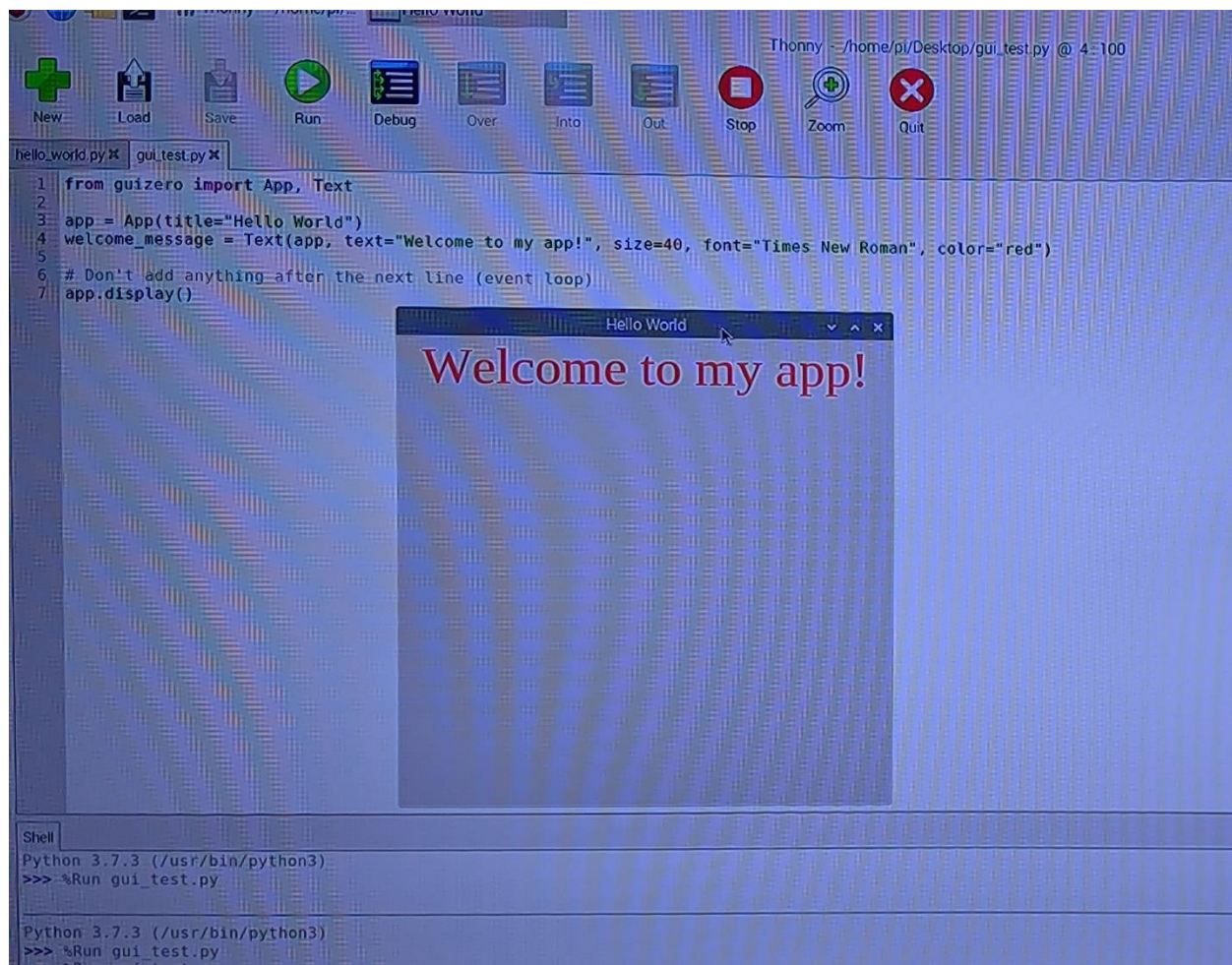process, and the result can be seen in Figure 6 below.

Figure 6. A screenshot of the basic GUI created with Python script.

While working on the documentation for the preliminary design review, more

research was done on the components required for the OBD-II port module. Some

pins on the ELM327 are not required depending on the protocol used in the vehicle, so the pinout of the OBD-II port in a 2008 Nissan Altima had to be verified by inspecting the port to see which holes had metal contacts for pins. This pinout can be seen in Figure 7 below.
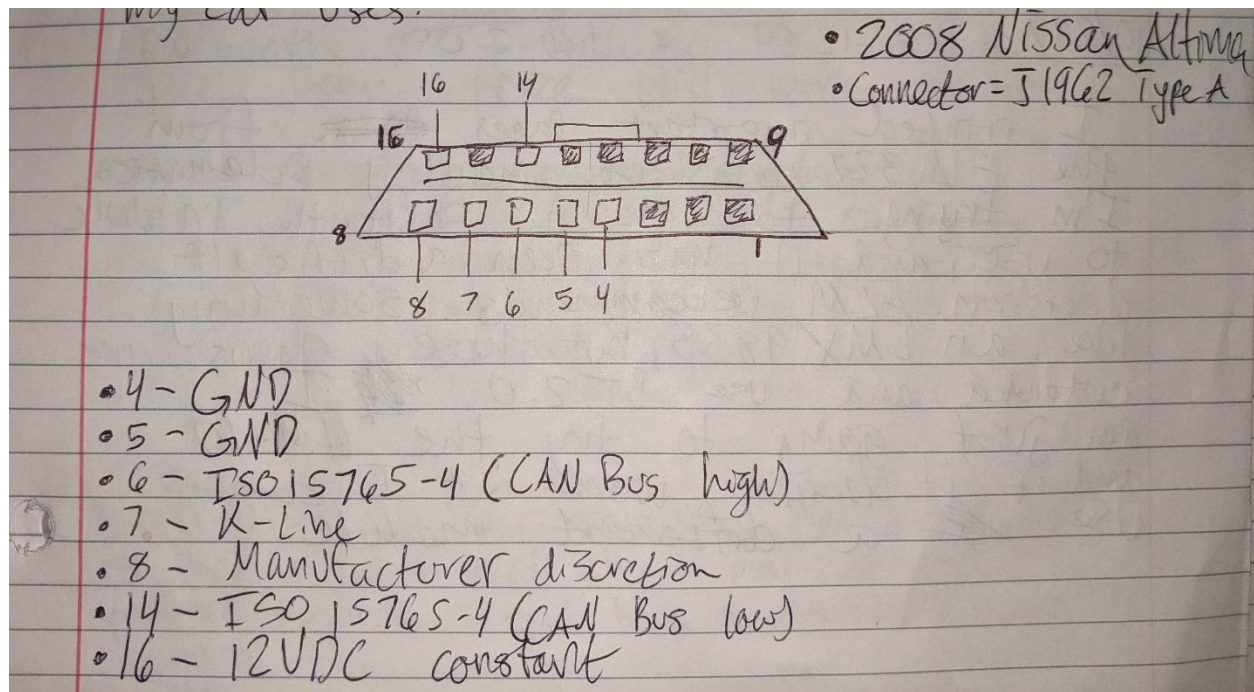

Figure 7. The pinout for the OBD-II port in a 2008 Nissan Altima.

Schematic capture with the KiCAD software began by creating a symbol library for the project, and the first symbol created was the ELM327 chip. An image of the symbol can be seen in Figure 8 below.
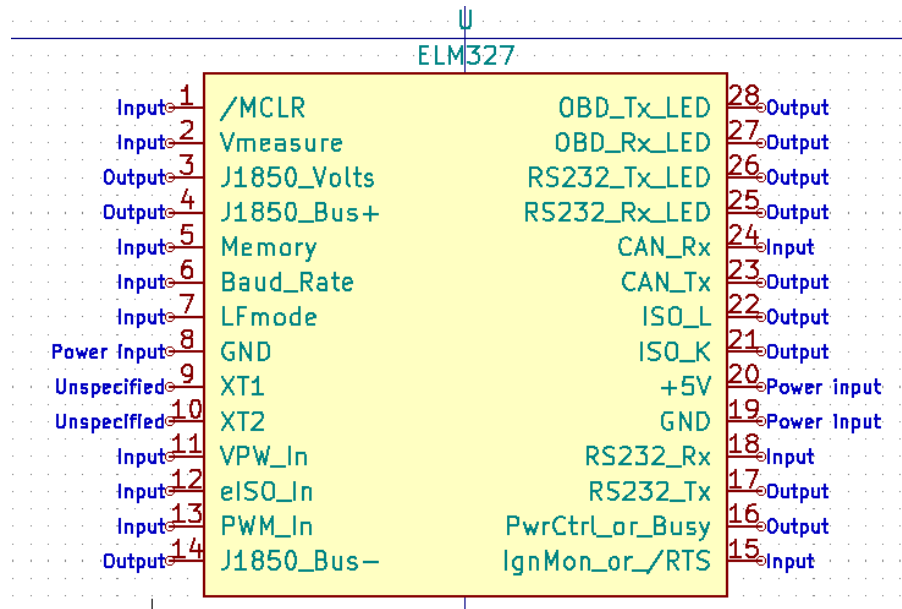
Figure 8. The ELM327 symbol created for use in schematics.

The official ELM327 datasheet includes a diagram for an example application circuit for the chip, so the schematic for the OBD-II port module will include pieces of the circuit recommended in said datasheet [6]. Since progress was being made on the schematic, a Bluetooth module had to be chosen. After more research, the ESP32 was chosen as the best fit for the project given the circumstances. This microcontroller uses Bluetooth version 4.2 so it meets requirements, includes TX and RX pins in the GPIO for serial communication, and will allow for much easier debugging.

While developing the schematic, a table of parts was made to make ordering and creating a bill of materials much faster. After creating more symbols and making the necessary connections, the schematic for the OBD-II port module was completed and can be seen in Appendix A. Development of the next schematic is currently underway and has involved researching rechargeable battery parts.

Requirements

1. The system shall work with the 2008 Nissan Altima OBD-II protocol, ISO15765-4 (CAN-BUS.)

   a. Other vehicles that use the same protocol may work with the system, however it is not required.

2. The system shall be able to read and clear diagnostic (trouble) codes.

   a. The user interface will display the diagnostic codes in list form with buttons to scroll up and down through the list.

      i. The list will display the diagnostic codes (e.g. P0011) only.

      ii. The user must touch one of the diagnostic codes to read the description or possible cause.

   b. The user interface will provide a button to clear all diagnostic codes.

3. The system shall have the ability to read sensor data at minimum 30 times per second including speed, coolant temperature, and oil pressure.

   a. The user interface will display the data in decimal format.

      i. The option for digital gauges may be implemented.

   b. The data will be displayed by default in units of miles per hour for speed, Fahrenheit for temperature, and pounds per square inch for pressure.

      i. The option for metric units shall be implemented.

4. The system shall use Bluetooth 4.0 or greater for data transfer.

    a. The Bluetooth version shall be 4.0 or greater because previous versions do not support Bluetooth Low Energy (BLE.) The range of the connection is also improved with newer versions.

5. The Bluetooth transceiver circuit that interfaces with the OBD-II port shall be powered by the OBD-II port.

    a. A power circuit will be designed to ensure power from the port is reliable.

6. The handheld unit shall use a touchscreen to display information and receive user input.

    a. The touchscreen of the handheld unit shall be 3.5" or greater diagonally.

    b. The touchscreen of the handheld unit shall operate at a resolution of at least 320 x 480 pixels.

7. The user interface shall not require multi-touch or swiping.

    a. Operation with single presses simplifies the user interface and is more accessible to those with disabilities.

    b. The option to scroll through a list by swiping up or down may be implemented.

8. The rechargeable battery shall be recharged via a USB-C port.

9. The rechargeable battery shall power for the handheld unit at full load for at least 2 hours.

a. The rechargeable battery must include protection and charging circuitry.

10. The charge level of the rechargeable battery shall be displayed on the user interface.

   a. The charge level will be displayed as a percentage, 100% being fully charged.

11. The brightness level of the touchscreen shall be adjustable in the user interface in 10% increments, 100% being maximum brightness.

12. The housing of the handheld unit shall include a magnet.

   a. The magnet will allow users to easily mount the device on the various magnetic dash mounts available (or any other magnetic surface.)

13. The handheld unit shall include a physical ON/OFF power switch.

14. The system shall use a single board computer or microcontroller.

15. The code and documentation developed shall be open source.

Glossary

Term	Definition
OBD-II	OBD-II stands for On-Board Diagnostics (version 2) and is the interface used in all vehicles model year 1996 or newer.
CAN Bus	CAN stands for Controller Area Network and it is the bus protocol used in vehicles and other embedded systems to send and receive data to and from all the various devices (nodes) connected to it. The OBD-II port has connections to the CAN Bus.

Appendix A

References

[1] Trello, Zak's Senior Project, https://trello.com/b/57kp2fr0/zaks-senior-project

[2] Amazon, Autel MaxiCOM MK808BT Diagnostic Scan Tool,

  https://www.amazon.com/dp/B07MBSZH48/

[3] Amazon, OBD2 Scanner Bluetooth 4.0 OBDII Scan Tool for Android Device &

  iOS, https://www.amazon.com/dp/B07MB8XH6Q

[4] Instructables, OBD-Pi, https://www.instructables.com/OBD-Pi/

[5] Raspberry Projects, M3 Pi: Raspberry Pi OBD-II Touchscreen Car Computer,

  https://projects-raspberry.com/m3-pi-raspberry-pi-obd-ii-touchscreen-car-

  computer/

[6] Elm Electronics, ELM327 OBD to RS232 Interpreter,

  https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf