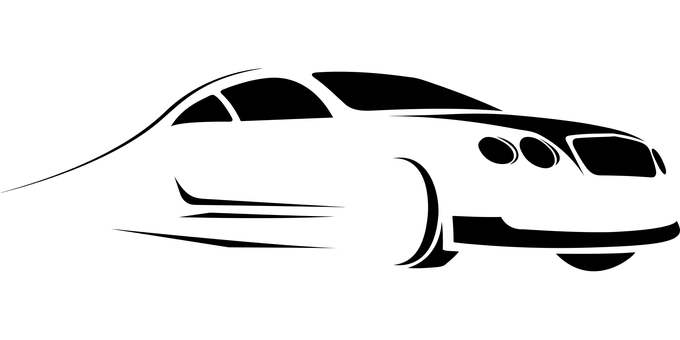
Carputer – All in One Head-Unit

Project Report

By: Rishi Nauth



Subject: Technical Project – TPJ655

Program: Computer Engineering Technology

Professor: Mahesh Aravamudan

Date: August 14th 2020

Table of Contents

**[Table of Figures](#_Toc48301885)** [3](#_Toc48301885)

[Executive Summary 1](#_Toc48301886)

[Introduction 1](#_Toc48301887)

[Functional Features 1](#_Toc48301888)

[Car section 2](#_Toc48301889)

[Audio section 2](#_Toc48301890)

[Project Specifications 3](#_Toc48301891)

[Operating Instructions 3](#_Toc48301892)

[Installation 3](#_Toc48301893)

[Startup and initialization 4](#_Toc48301894)

[Once Deployed 4](#_Toc48301895)

[Product Design, Implementation and Operation of the system 4](#_Toc48301896)

[Bluetooth 4](#_Toc48301897)

[GPIO 4](#_Toc48301898)

[General Coding 5](#_Toc48301899)

[Buttons 6](#_Toc48301900)

[Problem Handling 6](#_Toc48301901)

[Software and Hardware flow-chart 7](#_Toc48301902)

[Components and Captures GUIs 8](#_Toc48301903)

[Photos of GUI’s: 10](#_Toc48301904)

[Theory of operation 11](#_Toc48301905)

[Car section 11](#_Toc48301906)

[Code 11](#_Toc48301907)

[GUI 12](#_Toc48301908)

[Circuit 12](#_Toc48301909)

[Audio section 12](#_Toc48301910)

[Code 12](#_Toc48301911)

[GUI 13](#_Toc48301912)

[Maintenance 13](#_Toc48301913)

[Conclusion 13](#_Toc48301914)

[Further Developments 14](#_Toc48301915)

[Appendix 15](#_Toc48301916)

[1) Electrical Schematics 15](#_Toc48301917)

[2) Parts List 16](#_Toc48301918)

[Bibliography/References/ Citations, and Links to major component manufacturer specifications 17](#_Toc48301919)

[Contact Information 17](#_Toc48301920)

# **Table of Figures**

Figure 1: Variable Declaration 5

Figure 2: Flow-chart 7

Figure 3: Components 9

Figure 4: Audio GUI 10

Figure 5: Car GUI 10

Figure 6: OBDII Data Retrieval 11

Figure 7: Gear Indicator Calculation 11

Figure 8: Audio Folder Loop 12

Figure 9: Song Folder Path 13

Figure 10: Track Name Formatting 13

Figure 11: Electrical Schematic 15

Figure 12: Parts List 16

# Executive Summary

The system will interface with a vehicle to; display real-time vehicle parameters and current gear, provide a shift indicator light, provide audio playback features, as well as include user interface on a screen with 2 separate displays controllable by a toggle switch.

# Introduction

One of the many issues in the automotive industry involves the electronics that the users interact with. Car manufacturers have spent decades perfecting a vehicle’s fuel efficiency, horsepower and styling, yet, fall short when it comes to the head unit/infotainment system, and usually provide nothing in terms of updates over the lifetime of the vehicle. This is problematic due to the fact that vehicles are large purchases and most drivers keep their vehicles for 5+ years. Due to the rapid innovation in technology this is simply unacceptable. This project designed for the TPJ655 course at Seneca College, with the intention of incorporating all the knowledge gained through the 6 semesters of the ECT (Computer Engineering Technology) program, aims to create an all in one head unit. The Carputer is designed to be cost effective yet versatile, with minimal lag that provides modern and high-end features that most desire, yet are limited to only luxury or performance vehicles. With the current design the “car section” is intended to be a performance/race track oriented. It will pull information from the car to display what gear the car is in, show values not normally seen on the gauge cluster and includes a shift light indicator. However, the project is scalable and features that may be more desirable to the average driver, such as navigation can be added in the future. There is also the “audio section” where the system will provide users with a simple and easy to use GUI (graphical user interface) which will show what song is currently playing as well as display the current volume percentage.

# Functional Features

Although a single unit, the Carputer has 2 distinct sections, one being the car section and the other is the audio section. The car section interfaces with the car and performs related functions whereas the audio section acts as a media player and while it is used for audio playback in a vehicle it can be deployed as a standalone device.

The features are as follows:

## Car section

* The system will pull information from the vehicle’s ECU (engine control unit).
* An LED based Shift indicator light. At configurable RPM values, 3 sections of 4 LEDs with various colors will turn on at separate RPMs and all LEDs will eventually flash when the rpm reaches the highest configured RPM to indicate it is time to shift. This is applicable to both manual transmissions as well as automatic transmissions equipped with manual shifting modes. In non-manual shifting automatics, or while in automatic mode, the LEDs will still be lit at the set RPM levels but gives no relevant information to the driver as the car will shift itself into gear.
* A gear indicator using a 7seg display. Due to most cars not having an onboard gear value stored on the ECU, a calculation is done using speed and RPM values from the car to determine the current gear and it is then displayed on the 7seg display as the number 1-6. This is scalable as most manual transmission vehicles have 5-7 gears while some automatics can have 10+ gears.
* GUI designed using Pygame to show current rpm, car speed, and gear on screen and when the GUI switch is flipped, the system will switch to another GUI specified in the audio section

## Audio section

* The program will scan a specific folder for .mp3 files, the location will be configurable through changing the path within the .py file
* Play .mp3 files using the OMXPlayer application present on the Raspberry PI
* Output to either the car’s aux port to play audio through the internal speakers, or output to the HDMI display and use the built-in speakers.
* 2 buttons will be used for song control, one will be used to play next audio file in the folder while the other button will play the previous track. When at either end of the folder, the program will loop to the other end, i.e. pressing the next button while on the last song will loop back to the first song.
* Volume adjustment, 2 buttons are used to increase or decrease volume, the volume when the program starts is set to 0.35 (or 35%), and by default the volume buttons will increase the audio volume by 5% and the volume will not go above 100% or below 0%
* GUI to display current volume and song title
* The reason physical buttons are used as opposed to using a touch screen display is so that way, the driver’s eyes will not have to leave the road to see where on the screen they need to press in order to not miss and tap elsewhere on the screen.

# Project Specifications

The microcontroller used is a Raspberry Pi 3b+

The overall size of the project is 24cm x 17cm x 4cm

The current draw is under 2A to allow for functionality with a vehicles 12v port or if available, USB port, which normally won’t supply the Raspberry Pi’s default cable and adapter of 3A

The project is coded within a single python script

Storage is in the form of a 32 GB micro SD card, and can be increased if the end user intends to load more songs onto the system, the audio files make up the majority of the space required due to the other functions of the Carputer being related to a single python file roughly 11kb of space

Operating system: Noobs

Screen: 7-inch LCD with a resolution of 1024x600 connected through HDMI

# Operating Instructions

## Installation

The screen, LED’s and 7 seg display need to be placed within view of the driver. An ideal location involves; Being within the driver’s field of view so that the driver does not need to shift attention away from the road, not directly in the way of the windshield so as to not limit the driver’s vision, and not directly exposed to sunlight as it may push the Raspberry Pi past it’s temperature limit during high temperature days. To connect the Carputer to the car, the ELM327 module must be plugged into the OBDII port on the vehicle, usually located under the steering column on one of either side. The adapter’s red led will turn on when the vehicle is on to confirm the unit is receiving power. Next, power the raspberry pi by connected to a USB source, 2A recommended, this can be through a USB port present on the vehicle (read the owner's manual to see the current output of the port) or through the 12v port (commonly known as the cigarette lighter port) with an adapter to USB.

## Startup and initialization

Through the Raspberry Pi OS or through terminal, pair the Pi to the ELM327 module. Place all music in the default folder (/home/pi/Music) or change the path to a more preferred location if desired. Once completed the python file may be started through terminal

## Once Deployed

The audio player starts with no music playing, press the previous or next button to start playing a track. Audio up and down buttons are used to increase or decrease volume (the default button delay is set to 900ms and can be adjusted to allow faster follow up presses but if adjusted to low may register multiple inputs when only 1 press was intended). Flip toggle switch to switch between car GUI and audio GUI. The escape key on a keyboard acts as a backup exit, it allows for a clean exit of the file by setting all components in an off state without turning the vehicle off. When the vehicle is turned off the Raspberry Pi will shut down with it unless the port from which it’s powered allows for current to be drawn while the vehicle is turned off.

# Product Design, Implementation and Operation of the system

## Bluetooth

Bluetooth was decided as the connection method between the Raspberry Pi and vehicle, due to the abundance of ELM327 adapters as well as the fact that wireless capability would mean the unit may be placed in a more ideal location and can potentially lead to a cleaner install with minimal exposed wiring. The inherent disadvantage of this however, is that the Bluetooth connection has a minor, yet not insignificant, delay in transmitting data. The slight delay in when the gear indicator displays the correct gear or in the audio playback may not be problematic, however, for those looking for the fastest lap times, the shift indicator LED array may not have an ideal response time, and this can be remedied by using a physical OBDII to USB adapter.

## GPIO

Certain GPIO (General Purpose Input Output) pins are reserved for special functions on the raspberry pi, as such, where possible, standard GPIO pins are used to leave the special pins available for future usage.

These include:

* Pulse-Width Modulation (PWM), however software PWM is available on all GPIO pins.
* The PWM pins were used during this project due to the availability of software PWM and the limited number of available standard GPIO pins.
* SPI, the communication protocol between the Raspberry Pi and peripherals such as sensors.
* I2C, similar to SPI with similar applications.
* RX and TX, the Raspberry Pi serial communication pins.

## General Coding

During initialization, a connection is made to the car using the OBD library, the music file folder is scanned, as well as default parameters are set.

Values that may need to be changed are stored as variables, i.e. the rpm at which the LED’s light up are stored as four variables; rpmLow, rpmMid, rpmHigh, and rpmFlash. This allows reconfiguration to remain simple, if one would like to have the LEDs flash at, say, 5,000 rpm, all that needs to be done is to modify the rpmFlash assignment to “5000”.

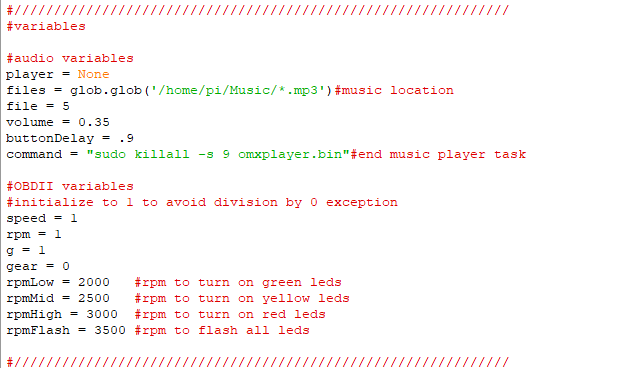


Figure : Variable Declaration

Most code contained within a while loop based on the variable “running” when the escape key is pressed the loop breaks and the program will run the last few lines of code to set all GPIO pins to low, shut off the audio player, as well as stopping the connection to the car.

## Buttons

The four buttons are connected to their respective GPIO pin from the same 3v3 source, and when each button is pressed it shorts the 3v3 to its corresponding GPIO pin at which point it is registered as logical high and its function will be processed.

The buttons and switch are all internally pulled down using the Raspberry Pi GPIO pull-down function to prevent floating, without the use of an external resistor

An internal delay is programed in the form of a variable, buttonDelay, and is set to .9 seconds to prevent multiple presses being registered if the user cannot release the button fast enough

The Raspberry Pi internal software, OMXPlayer, handles the playing of the audio files

## Problem Handling

Due to the internal current limitation of the Raspberry Pi, the LED array cannot be powered directly from the GPIO pins at the same time as the 7 Seg display. To alleviate this, the LEDs are powered by an external and replaceable 9v battery and are turned on using a level shifter in the form of 3 2N3904 NPN transistors.

The default value for rpm and speed is set to 1 due to a division calculation in the while loop to determine gear which will throw a division by 0 exception, if initialized to 0

# Software and Hardware flow-chart

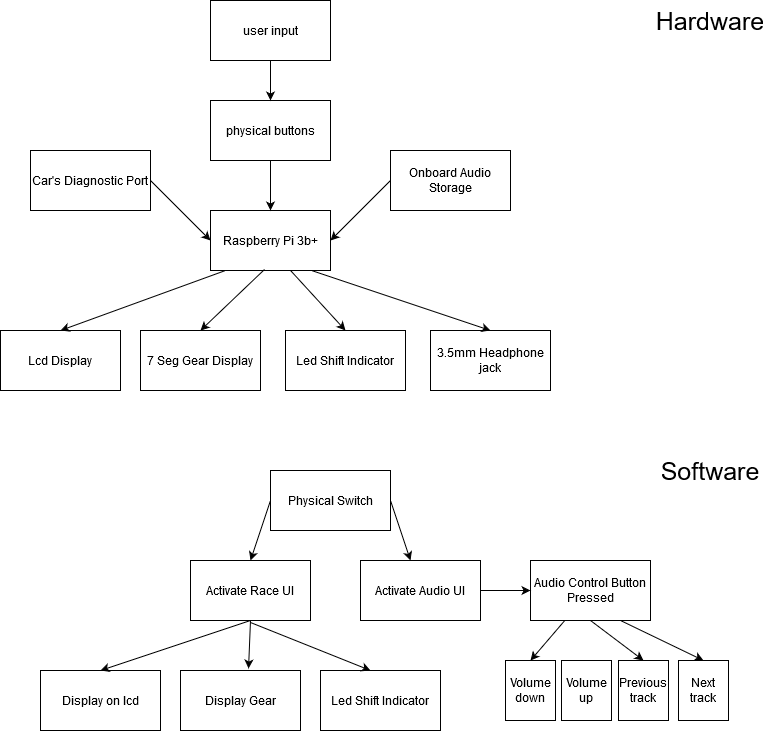


Figure : Flow-chart

# Components and Captures GUIs

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Component Picture** | **Component Description (from Manufacturer)** |
| Raspberry Pi 3b+ | https://images-na.ssl-images-amazon.com/images/I/810pJSU2UEL._AC_SL1500_.jpg | Programmable Microcontroller   * Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz * 1GB LPDDR2 SDRAM * 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE * Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) * Extended 40-pin GPIO header * Full-size HDMI * 4 USB 2.0 ports * CSI camera port for connecting a Raspberry Pi camera * DSI display port for connecting a Raspberry Pi touchscreen display * 4-pole stereo output and composite video port * Micro SD port for loading your operating system and storing data * 5V/2.5A DC power input * Power-over-Ethernet (PoE) support (requires separate PoE HAT)   <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/> |
| UNIROI 7 inch HD LCD Screen for Raspberry Pi |  | HDMI display   * UNIROI 7 inch LCD Screen for Raspberry Pi, fashionable and ultra-slim, compatible with Raspberry Pi 3 2 Model B+ 3B 2B B+ A+ A. * High definition screen with standard HDMI input signal interface. Default 1024\*600 resolution. * Universal USB 5V power supply interface design, support power bank, PC and 5V 1A power adapter power supply. Working current: 400mA.   <https://www.amazon.ca/gp/product/B07D6P2R5S/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&th=1> |
| 7 segment display |  | 7 Segment Display   * Common cathode * Includes 2 pieces Red 14.20mm (0.56”) Single digit numeric display series. * LED color Red * Forward Voltage: 2.2V Forward current: 50mA   <https://www.amazon.ca/gp/product/B07PLBL7FB/ref=ppx_yo_dt_b_asin_title_o05_s01?ie=UTF8&psc=1> |
| ELM327 adapter |  | * Read diagnostic trouble codes, both nonspecific and manufacturer-specific, and display their meaning (over 3000 nonspecific code definitions in the database). * Clear trouble codes and turn off the MIL ("Check Engine" light). * Super Compact PCB Board Design offers the latest chip as well as a CSR high quality Bluetooth module. * Works with the torque android app. * Works with all OBD-II compliant vehicles. * <https://www.amazon.ca/Bluetooth-Diagnostic-Scanner-Torque-Android/dp/B01DCZ3F7Y/ref=sr_1_4_sspa?dchild=1&keywords=elm327+bluetooth&qid=1597417507&sr=8-4-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzVDhBOUE4TlpTNVc0JmVuY3J5cHRlZElkPUEwMzM3NTgyQlowRzFINkFNME9XJmVuY3J5cHRlZEFkSWQ9QTAwOTU0NTJPUDU3MFJaNU80WDMmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl> |

Figure : Components

## Photos of GUI’s:

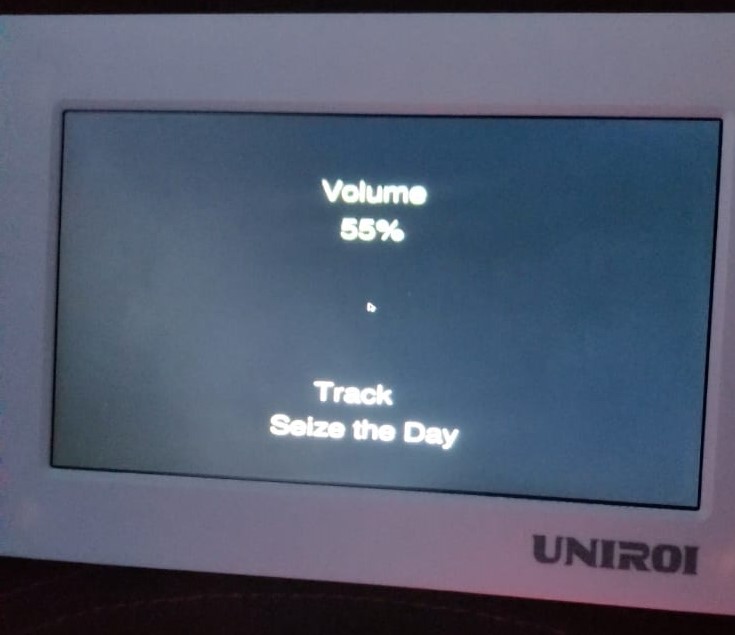


Figure : Audio GUI



Figure : Car GUI

# Theory of operation

## Car section

### Code

The car is connected to the system through an ELM327 Bluetooth module to be plugged into the vehicle’s OBDII port and paired on the PI and once the Python program runs, the data is set to constant retrieval and The rpm and speed are pulled through the usage of the getRpm(); and getSpeed(); functions constantly



Figure : OBDII Data Retrieval

While there are only 2 parameters pulled from the vehicle, rpm and speed, there are 3 displayed on the screen; rpm, speed, and gear. Most cars do not have a gear value stored on the ECU, so a gear is calculation is required. By dividing the speed by the rpm the gear ratio of the transmission may be obtained, based on which, a gear number is assigned. The system assumes a manual transmission for the gear indicator and as such, an upper and lower threshold is set for the gear ratio to account for minor differences in clutch mating as well as clutch slippage, however with an automatic transmission the gear indicator will still work as intended but the upper and lower thresholds are unnecessary.

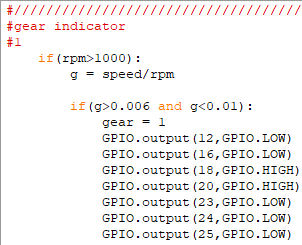


Figure : Gear Indicator Calculation

### GUI

If the GUI switch, S1, is closed, the carHud(); function is called, drawing a black screen with the labels for each parameter to be displayed and underneath, showing said parameters.

### Circuit

Once a gear ratio is determined the gear variable is set to a number between 1 and 6, and the relevant GPIO pins are turned on to display that number on the 7 seg gear indicator. The led array is in 3 sections. The green, yellow, and red LEDs are grouped together and when the corresponding rpm for rpmLow, rpmMid, rpmHigh, and rpmFlash are reached, the related functions happen, green LEDs turn on, yellow LEDs turn on, red LEDs turn on, and all LEDs flash on and off every 200ms.

Optical isolation was initially theorized for the usage of the level shifters to protect the MCU, however ideally, there is no inductance in the circuit. As such, while it may be a good preventative measure, it is not required for the current design

## Audio section

### Code

Once the next or previous button is pressed, the playFile(); function kills the current task before starting the next song to prevent issues of multiple songs playing at once. The volume level is adjusted in increments of 5 but can be configured manually within the script if one desires finer or coarser adjustment and once an audio file at either end of the folder is played, the program will loop to the other end of the file to keep a continuous loop of audio files

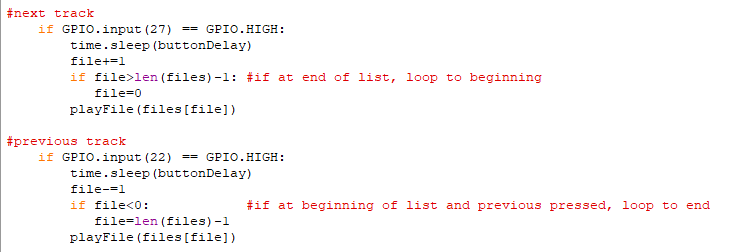


Figure : Audio Folder Loop

### GUI

If the GUI switch is open, the audioHud(); function is called, and will display the current song playing as well as the current volume which is converted from 0-1 to 0-100 before being called to the display, to show a more relevant audio level to the user, i.e. Percent rather than a decimal. The current song parameter is formatted to show just the song name as opposed to the path and extension that is gained through figure 9.



Figure : Song Folder Path

The formatting removes the first 15 characters and the last 4. The first 15 characters are part of the path which includes folder names and the forward slash between them and the last 4 is for the extension which in this case is .mp3. If a longer path is selected then this formatting will need to be adjusted so as to not display irrelevant information as all songs must be located within the same folder anyways.



Figure : Track Name Formatting

# Maintenance

The 9v battery that powers the led array will need to be replaced when it dies. This however won't affect the other portions of the project such as the gear indicator or any of the audio section as the 9v is only electrically connected to the led array.

# Conclusion

To conclude the project was successful, all features have been met, and are fully functional before the specified deadline. It offers an affordable way to have reliable features implemented in a vehicle while allowing customization to the end user’s preference. Due to the current pandemic, the initial idea of designing a PCB (printed circuit board) to contain the entire circuit had not been reached, but it still may be implemented at a future date although it is no longer within the scope of this project.

# Further Developments

Due to the maximum current limit that the Raspberry Pi can output, greater efficiency or switching the internal power source to an external one is a welcome addition. One possibility is using a common anode 7Seg display instead of the common cathode one used in this project as MCU’s can generally sink more current than they can source. For the audio section one can add a random number generator (RNG) so as when the user would like to shuffle all songs, the RNG will provide a number and play the indexed song associated with that number. The RNG number will have to be limited to the length of the file folder however.

The possibilities of further development of this project is practically endless, one can look towards auto manufacturers and implement designs that are already available as options or standard on higher-end vehicles to gain these features on lower end vehicles for a fraction of the price or to customize these features to one's own liking. Some of these features could include a backup camera with guide lines, parking sensors and navigation. More invasive features can be implemented but will require major enhancements to achieve, as Carputer is meant to just retrieve and modify data from the vehicle rather than being used as an input to the ECU. These features may include, collision mitigating braking, lane keep assist, and vehicle stability control. Due to the flexibility of an MCU such as the Raspberry Pi, the main limitation to modifications is just getting the idea.

# Appendix

## 1) Electrical Schematics

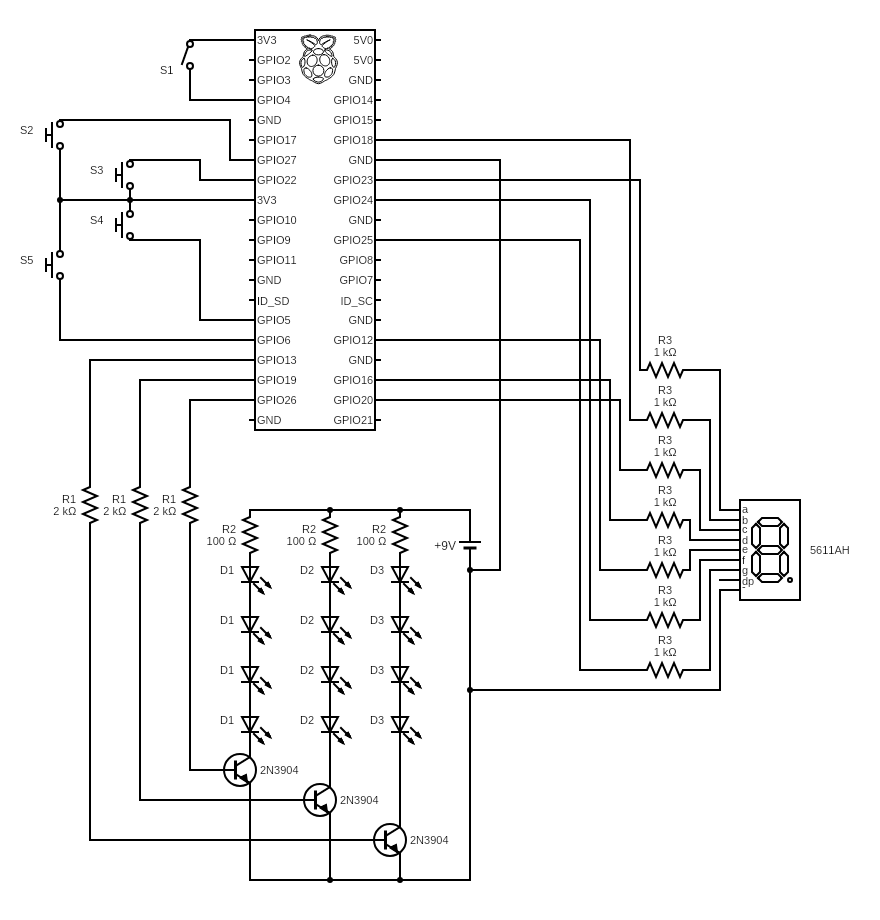


Figure : Electrical Schematic

## 2) Parts List

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part#** | **Model #** | **Reference Designation** | **Part** | **Unit Cost** | **Quantity** | **Subtotal** |
| 1 | Raspberry Pi 3 Model B+ | N/A | Raspberry Pi | $99.99 | 1 | $99.99 |
| 2 | N/A | D1 | Red LEDs | $0.375 | 4 | $1.50 |
| 3 | N/A | D3 | Green LEDs | $0.375 | 4 | $1.50 |
| 4 | N/A | D2 | Yellow LEDs | $0.375 | 4 | $1.50 |
| 5 | 2n3904 | Q1 | NPN Transistors 2n3904 | $2.50/25 | 3 | $0.30 |
| 6 | KS0219 | N/A | GPIO Breakout | $13.99 | 1 | $13.99 |
| 7 | N/A | BT1 | 9v Battery | $1.00 | 1 | $1.00 |
| 8 | 57-902-0 | N/A | Battery Clip | $1.50/2 | 1 | $0.75 |
| 9 | 7200022 | S1 | Toggle Switch 1P1T | $3.95 | 1 | $3.95 |
| 10 | LS-00003 | S2 | Push Buttons MOM | $1.50 | 4 | $6.00 |
| 11 | RDC-2K-10 | R1 | 2K Ohm Resistor | $1.50/10 | 3 | $0.45 |
| 12 | RDC-100R-10 | R2 | 100 Ohm Resistor | $1.50/10 | 3 | $0.45 |
| 13 | 5161AS | N/A | 7 Segment Display | $8.70/2 | 1 | $4.35 |
| 14 | RDC-1K-10 | R3 | 1K Ohm Resistor | $3.00/10 | 7 | $2.10 |
| 15 | UR071-US-FBA | N/A | HDMI Screen | $65.99 | 1 | $65.99 |
| 16 | BTOBD-Black | N/A | OBDII Bluetooth Adapter | $15.99 | 1 | $15.99 |
|  |  |  |  |  | Total: | $219.81 |

Figure : Parts List

## Bibliography/References/ Citations, and Links to major component manufacturer specifications

<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>

<https://www.amazon.ca/gp/product/B07D6P2R5S/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&th=1>

<https://www.amazon.ca/gp/product/B07PLBL7FB/ref=ppx_yo_dt_b_asin_title_o05_s01?ie=UTF8&psc=1>

https://www.amazon.ca/Bluetooth-Diagnostic-Scanner-Torque-Android/dp/B01DCZ3F7Y/ref=sr\_1\_4\_sspa?dchild=1&keywords=elm327+bluetooth&qid=1597417507&sr=8-4-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzVDhBOUE4TlpTNVc0JmVuY3J5cHRlZElkPUEwMzM3NTgyQlowRzFINkFNME9XJmVuY3J5cHRlZEFkSWQ9QTAwOTU0NTJPUDU3MFJaNU80WDMmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl

7seg fig <http://www.xlitx.com/Products/7-segment-led-dot-matrix/5611ah.html>

## Contact Information

Rishi Nauth

[Rnauth2@myseneca.ca](mailto:Rnauth2@myseneca.ca)

(647) 461-4565