Shift Buddy

Matthew Bichay

California State University, Monterey Bay

CST 499: Directed Capstone

Advisor: David Wisneski

June 3, 2016

Executive Summary

The Shift Buddy is a project primarily designed for one core essential purpose, *speed*. The Shift Buddy is a tool designed to help drivers learn how to drive their motor vehicles to the maximum potential of its mechanical abilities. The Shift Buddy's main essential function is to notify the driver of the optimum time in which they should shift the vehicle's transmission to achieve fastest possible acceleration.

The Shift Buddy provides a convenient solution to tackling this problem. Through the entry of some simple key information, the Shift Buddy backend can calculate the optimum shift point for your vehicle's transmission and finally export it as a database or lookup table for information in which a microcontroller can do some simple post processing to figure out which shift point corresponds with the current gear and notify the driver on when to actually shift. The Shift Buddy can be dissected into two core problems. The "optimum shift point" can be easily defined as the point on a vehicle's RPM band in which the vehicle incurs the smallest loss in torque resulting from an upshift. How can we calculate the optimum shift point? And the final not-so-obvious challenge is, how can we figure out what gear the vehicle is in real-time? The diagnostics port on a car offers a plethora of telematics information which directly interfaces with the engine control unit (ECU), but this also means that the ECU has no knowledge on what gear the vehicle is actually in. Shift Buddy plans to solve this problem by providing an answer to these major challenges; Shift Buddy contains an effective algorithm for looking at a small set of basic information doing some mathematical processing for estimating the optimum shift point. Likewise, Shift Buddy also takes this information and prepares it for post processing where

onboard algorithms can do real-time processing to figure out what gear the vehicle is in and determine the optimum shift point based on how the driver is currently driving.

The basic design for a off-the-shelf shift light is a small microcontroller with a set of LED lights. Some also provide a small digital display for displaying the RPM the vehicle is currently in. These lights connect to the vehicle's OBD-II diagnostics bus or can be wired directly to the ignition connection for older vehicles which don't support OBD-II connections. These lights then quickly poll the diagnostics bus for the vehicle's RPM value and when the value reaches the user-selected RPM value, the light will flash to the driver.

When driving a manual or semi-automatic transmission, being able to shift gears for racing is a totally separate skillset than shifting a transmission on a daily commute. Many drivers think they possess the skill to do it correctly because they've driven manual transmissions their entire life; this is not the case, it takes practice. The best way to train yourself to shift correctly is to use dynamometer, review the torque curve charts for the vehicle, , and then finally do some math to figure out the shift points for your vehicle based on the transmission's gear ratios. Once calculated, it is important to train to them and re-do this process for every modification or change in power [7][9][10].

There are a few primary goals of this project. First and foremost, the primary goal is to create a software application to provide a solution to the technical question on how to best help a driver learn the optimum way to control the car to produce the highest performance regulated by its transmission. The second goal is to produce a tool which is completely customizable, modifiable, and expandable across multiple platforms and vehicles. The third and final goal is to provide an open source and affordable solution which rivals similar available products in not only price, but customizability and flexibility.

The car enthusiast community is an extremely tight knit and diverse community. Many of these people know very well that being a car enthusiast is far from the cheapest hobby one could be interested in, performance modifications come with a hefty price tag, and for a good reason, engineering is not a cheap effort. Car enthusiasts come from many different ethnic, financial, and educational backgrounds, yet are brought together by the similar vested interest in cars. Whether you're an enthusiast which prefers to keep a vehicle stock or modified, this project attempts to reach out to all of these sub communities and offer a solution to a interesting and significant problem; which is, after purchasing a new car, modifying an existing car, etc, how do you know where on the RPM band you should shift? A common misconception is that the red-line on a vehicle's tachometer is simply how you achieve best performance; this is a common misconception and since vehicles vary from one to another in how they behave, it isn't necessarily true for all vehicles [6][11].

Table of Contents

Table of Figures

Introduction

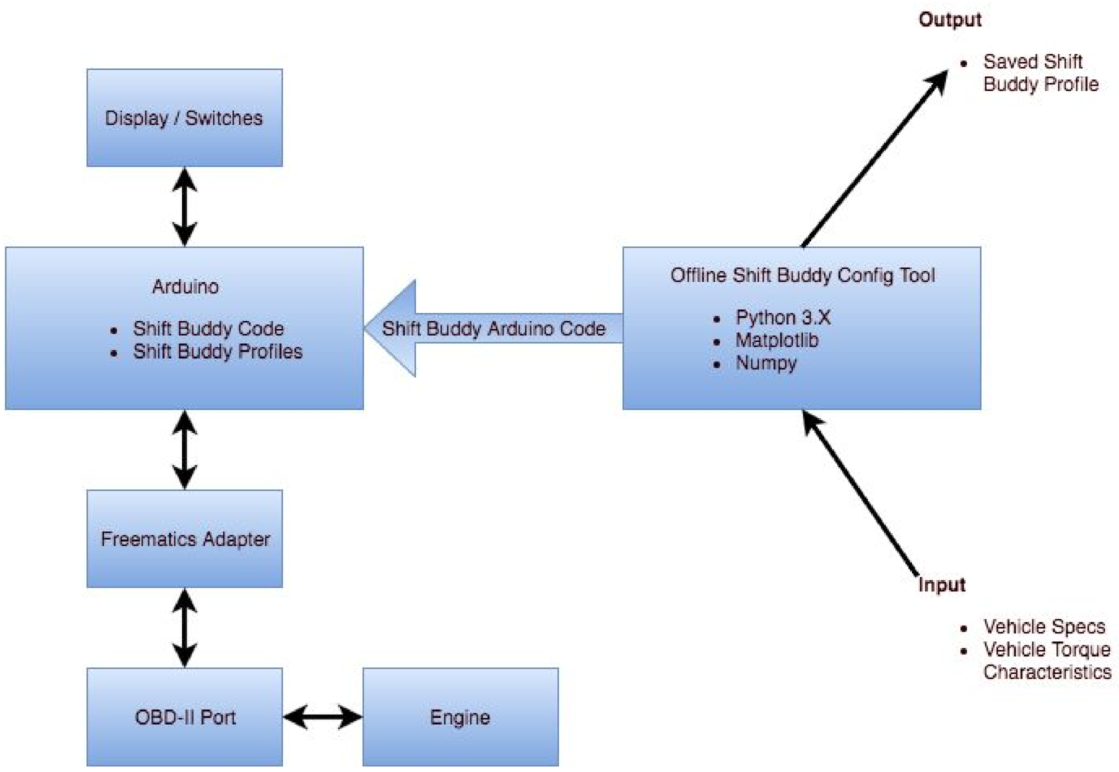
This product is named, “The Shift Buddy” to emphasize the primary feature which it offers; it is designed to be a *friend* or helper tool for training drivers when to shift the vehicle’s manual or semi-automatic transmission for maximum performance. This product is comprised of four major components which all communicate together to provide a full system for the driver to integrate into *any* of their vehicles.

1. Microcontroller for processing
2. Communication bus to the vehicle’s OBD-II diagnostic/telematics port
3. Display for relaying information to the user
4. Backend software suite for configuring/calibrating the tool properly

With these four components, this product has the ability to positively affect the car enthusiast community. This product plans to make an impact for that community by providing a powerful tool to keep up with, and allow drivers to get the most out of their vehicle’s performance before and after modification.

Problem Breakdown

The Shift Buddy accomplishes what it does using the following schema seen in the diagram below.



Apdx-A

First input and interaction is done with the offline configuration tool. The resulting output is a profile(s) which corresponds to a specific vehicle. The profiles along with the Arduino code is then compiled and uploaded to the Arduino microcontroller. The Arduino Shift Buddy code contains the main control loop which interfaces with the freematics adapter. This adapter sends queries to the OBD-II port to receive engine information about the vehicle. Using the information retrieved from the vehicle, the Shift Buddy can then interface with the uploaded profiles and update the display with real-time information. The Shift Buddy display also contains hardware switches for switching between profiles and different information which the user would like to conveniently display on the seven segment display (RPM, speed, boost pressure, etc).

Goals and Objectives

The following table contains the list of goals and the objectives used to tackle these goals.

Goals	Objectives
Working shift light programmed for making timed upshifts.	• Optimum Shift Point Algorithm ○ Design an algorithm for calculating the best shift point for any car. • Current Gear Model ○ Design an algorithm for estimating what gear the vehicle is currently in. • Display Controller ○ Design code for communicating and interfacing with the user via some display system. • Vehicle / Microcontroller Messaging Interface ○ Design code for polling the vehicle for real-time diagnostic and telematics information.
Customizable Apdx-B	• Configurable ○ Design code for handling multiple profiles at once for on-the-fly switching (useful for multiple cars). ○ Design code for handling multiple options for displaying purposes (Speed, RPM, boost pressure, etc). • Manual overridable settings ○ Designing an Optimum Shift Point Algorithm bypass for users who want to experiment with manual entry of shift points (Useful for experimenting with profiles such as a set of shift points for achieving goals other than optimum acceleration).
Modifiable	• Functionality Flexibility ○ Design a architecture which can be easily modified by using specific hook functions which modularize much of the output and interface to the user. Programmers can customize to their likings. • Hardware Flexibility ○ Design an architecture which isn’t tightly coupled with any specific hardware component. Programmers can easily switch displays or other components and with minor code modifications, adapt the current code to take advantage of the difference in hardware.
Open, Easy, and Interesting	• Open Source ○ Leave codebase fully open-sourced. ○ Use fully open-sourced hardware.

	○ Use fully open-sourced libraries. ● Well Documented ○ Documentation for wiring guide. ○ Documentation for installation. ○ Documentation for configuration. ● Analyzable ○ Use of different mathematical techniques (interpolation schemes) to represent a torque curve. ○ Built-in plotting function for viewing the generated torque curve.

Community and Evidence of Usefulness

The car enthusiasts community, specifically the vehicle modders who consistently invest and upgrade their vehicle's performance have all faced the dilemma of having to do cost-benefit analysis when deciding to upgrade between two parts. Car modifications are expensive and people often sacrifice safety for performance, performance for safety, performance for convenience, etc. One of the biggest things to look at is the amount of performance someone would gain from something like newer tires, a better exhaust, or a shift light. Yes, the shift light would train you to utilize your existing hardware to its fullest potential, but when the cost of a new shift light is $250 and the cost of a brand new exhaust or downpipe is $500, it can be easy to see why drivers would prefer the downpipe. Everywhere there are people who are making these modifications to their cars but not truly seeing the maximum potential of their upgrades and investments. The Shift Buddy offers what the $250 off-the-shelf shift light provides and more at an estimated cost of $65 dollars or less [5][7]. The hope is that at this lower price point along with some other attractive features, Shift Buddy can encourage people to put forth a better effort in learning how to drive their vehicle and provide a tool which is enjoyable to tinker with and tweak to one's own likings. Drivers are knowledgeable of the benefits of having a optimized shift light, it is one of the most common motifs of Formula 1 and GT racing, most of these newer performance vehicles have a shift light and diagnostics information displayed on the steering wheel or dashboard and it is necessary for these drivers to eek the most potential out of their vehicles to gain a competitive edge during their races [7][10]. The reason why it is not a more common modification for cars isn't because it isn't useful, it's because aftermarket options are few, expensive, and lack some of the flexibility which would drive that high price point, this is where the Shift Buddy's usefulness truly shines.

Feasibility

There are many options for shift lights on the market. The performance and vehicle modification community is a large community with many willing buyers willing to pay the money for performance upgrades. After conducting an environmental scan, I've found two issues with the currently available products. First, there are many products out there which claim to be proper shift lights, however simply function on a constant single value input by the user; this is not only an inflexible design, it is also deceiving considering that shift points can possibly be different for *every gear* in a transmission's gear set. Second, the product does nothing to *help* the users decide upon efficient shift points for their vehicles.

These lights can cost anywhere from $70 for the most basic designs and over $250 for more complex designs which contain digital displays and maybe the possibility to set separate shift points for different gears. In this element, this is where Shift Buddy is quite competitive in terms of price and feature set. With careful purchasing, a Shift Buddy can cost less than the high estimate of $65 and offers more features than a shift light with the price of $250 [3][8]. The Shift Buddy uses fairly inexpensive open source hardware. The main controller is an Arduino which is required to run the C code compiled for the device. It does however work on the cheapest and smallest Arduino microcontroller, the Arduino Uno, which can be purchased anywhere from $15 - $25. The display which Shift Buddy supports natively is the TM1638 display which has 8x digital segments for displaying data, 8x LED bulbs, and 8x programmable buttons, this can be purchased at an extremely affordable price of $8 [1][2][4][5]. Finally, Shift Buddy natively supports the Freematics OBD-II Arduino adapter for connecting to the vehicle's OBD-II bus for data polling *and* power. The architecture which was chosen for the Arduino code was done in a way that the hardware isn't tightly coupled with any of the components, meaning if a programmer wanted to support a different display or OBD-II adapter, this is all completely possible with minor modifications. Finally, the configuration portion of the Shift Buddy can be used on any computer operating system or platform, the only requirements is the newest version of Python and the numpy and matplotlib libraries compatible with that Python version.

With the use of open source hardware, mainly the Arduino, and Python as a back-end configuration tool, Shift Buddy was able to implement flexibility which no off-the-shelf shift light can offer. This allows shift buddy to receive continuous optimizations and updates as seen fit and the powerful Arduino platform allows for endless customization and interface with sensors which go far beyond a simple connection to the OBD-II bus. Users can customize not only *when* the shift light notifies them, but how. The Python tool is broken into different functional pieces in a way that users can program their own tools or interfaces and even automate creating profiles further. The profile system implemented in the Python backend also allows users to *share* their profiles with other people, essentially allowing people to collaborate on the best way to configure their cars.

Functional Decomposition

The Shift Buddy can be decomposed into two main functional areas. The Shift Buddy config tool used for configuring and doing analysis on the profiles which a user would want to upload into the Shift Buddy. The second functional area is the C/C++ Arduino Shift Buddy control software which uses the exported header file created by the Python backend. The Arduino codebase contains the main control loop used for interfacing with the vehicle and the display, it contains the main logic for the shift light and interfacing with the user using the integrated buttons in the display. See Apdx-C for more details.

Final Deliverables

The final deliverable for this project is essentially a Github link. The Github link will contain three main components. The first component is the ConfigurationTool folder containing all of the Python codebase used for configuring the Shift Buddy. The second component is the ShiftBuddy folder which contains all of the Ardunio codebase used to upload to the Arduino and interface with the hardware / run the product. The third deliverable will be the docs folder which contains all pertinent docs relating to this product as well as a README file which contains explanations on requirements, dependencies, installation and wiring guides. Finally, the last deliverable is the full Shift Buddy hardware which is a working model using the software created in development.

Approach / Methodology

The approach and methodology used for creating this product was an iterative approach towards completing the objective. Once the hardware was tested, Matlab was used for making a quick initial prototype of all the major algorithms. The “optimum shift point algorithm” and “gear shift model” algorithm was originally implemented by pulling a large collection of driving data from two different vehicles and using the data to plot and analyze the two major algorithms. See Apdx-D for more details.

Legal and Ethical Considerations

Legal concerns revolving the Shift Buddy pertain to potentially commercializing the product. Shift Buddy uses a variety of open source libraries, mainly the Freematics OBD-II library and the TM1638 library used for communicating with the display. These libraries are distributed under the Berkley Software Distribution (BSD) license and Gnu Public License (GPL). Another thing to think about is the Arduino license. Arduino is free to use and distribute as long as credit or homage is given to the original Arduino source. In order to distribute this product as more than a DIY concept, Shift Buddy would have to develop its own hardware and interface/libraries to the hardware, otherwise the code would have to be left open sourced and adhere to all of its hardware and software’s license agreements in order to avoid copyright issues between Shift Buddy and the involved parties which might occur.

The major ethical concerns of this project would be involved in the testing and using the final product of the capstone. The purpose of the project is to give drivers the ability to practice shifting their vehicle's gearbox at the most optimum position for minimum power loss so they can harness the fullest potential of their specific vehicle; this entails ethical problems in itself because driving fast could endanger others and the user. There is also the potential problem of the testers or users testing the functionality of the shift light on public roads, which could be illegal and even dangerous to the people on the roads.

Fatal car accidents cause problems every year, many of the cases involve reckless driving and driving above the speed limits, resulting in deaths of all parties involved in driving on the street at the time. Reckless driving accidents also cost the taxpayers dollars in cleanup costs as well as impede other motorists continue when accidents cause traffic jams. These are all potential concerns when looking at a project which is designed to help drivers put the most power they can to the road, drivers which use the project to drive on public roads endanger others, themselves, and cause all of these potential problems.

Other than a disclaimer stating that this device *shouldn't* be used on public roads, there is a concern that vehicles driven at their maximums repeatedly will cause unusual wear and tear on the transmission and motor. A disclaimer stating that the project is in no way responsible for the extra stress and wear on your vehicle will be placed on the front page of the product's Github repository; this is just as important as a disclaimer stating that the project will not be used outside of a safe testing track.

Dangers of the testing phase in designing this project can be mitigated by purchasing or creating an OBD-II simulator. OBD-II is the main source of data output from the vehicle's diagnostics, which feeds directly into the proposed project's logic and control functionality. This means that with a simulation, testers can avoid having to test the product using an actual vehicle, everything can be tested from a computer and a USB port. An OBD-II simulator is a device used to simulate what the OBD-II port on a vehicle outputs during vehicle operation. In the picture below, the knobs each correspond to a specific vehicle output such as RPM, Speed, etc; the red switch is an on/off switch which simulates engine ignition, and the OBD-II port is on the side for direct connection to the simulated engine.

Apdx-E

As this device is meant to be open source and hackable, there is no way to avoid drivers from using the end result on public roads. Products are already available which do the same thing, just at a higher cost; it will be up to the driver's discretion whether they want to use the final product like it's meant to be used, on a race track, or if it will be used (in compliance with traffic laws) on public roads.

This project is freely available to the public, it will require only the necessary parts to build the system as well as a vehicle in which the transmission can be shifted manually by the driver. This project doesn't marginalize anyone and will be primarily noticed by people who follow car specific forums as well as car enthusiast cultures. One consideration might be made for the companies which are already spending the serious money and engineering to create and package similar products; this project is expected to steal some business from these companies as well as create some fair competition in the performance parts industry.

Timeline and Budget

Budget breakdown [1][2][4][5]	● Arduino Uno - $25 ● 8x Seven Segment, 8x LED, 8x Switch display $8 ● Wires and misc. - $10 ● Freematics OBDII V4 - $25 ● Freematics OBDII Emulator - $240 (Not required)

Milestone # / Date	**Objectives**
#1 5/10	● Pull data from OBD-II Simulator as well as vehicle (preliminary testing and logging of equipment) ● Creating test cases using the OBD-II simulator and supporting websites ● Testing of shift lights and external Arduino sensors
#2 5/22	● Complete creation of basic upshift algorithms ● User programmable setup fully implemented

#3 6/8	● Fully tested algorithms and user input ● Documented and source controlled codebase ● Final package for Arduino and accessories

Usability Testing and Evaluation

With every engineering project, there must be some form of validation and verification of the design. In order to validate the design, I have done a small field study of different public algorithms and websites which offer the calculation of shift points. I used a grouping of these websites to average the calculated results of my test cases and scenarios in order to create a "truth set". This truth set was used to test and evaluate my design based on how close or how much of an improvement my newer algorithms make. Plotting the results over iterations of design and testing helped me come up with what I feel would be the best possible design for real-life usability on the drag strip or race track.

To conduct final testing, shift points was be tested on open roads. Gears were tested under non WOT (wide-open throttle) positions on the freeway in order to test and evaluate. All open road testing was done within the legal speed limits as well as tested using multiple OBD-II compatible cars. See Apdx-F and Apdx-G for more information.

Final Implementation and Validation of Design

There are only two major functions which need some explanation. The rest of the codebase is simple implementation and design decisions for interfacing with hardware or communicating with the user.

The first major important function was to implement the “optimum shift points algorithm”. This algorithm was designed to take two inputs, an array of overall gear ratios, 2D array representation of a torque curve.

Overall Gear Ratios	● (Current Transmission Gear Ratio * Final Gear Ratio) ○ Final gear ratio is the collective ratio of the wheel to transmission connecting hardware (differentials, transfer cases, etc).’
Torque Curve	The torque curve is an array containing an RPM/torque value pairs. In the Shift Buddy’s implementation, it allows users to enter in a minimum and maximum RPM. After entering a minimum and maximum RPM, for every 500 RPMs between the min and max, the user will enter a torque value (reading from a dynamometer chart). Using these values, Shift Buddy will do a linear or lagrange interpolation (user selectable). In most cases, linear interpolation works fine; however sometimes the lagrange interpolation allows the user to account for some of the unique rounded curves in the torque curve.

The outputs for the “optimum shift point algorithm” is an array containing the calculated shift point for *every* gear ratio entered by the user. These values identify the RPM in which the user *should* shift in order to reduce the amount of torque lost between each shift during wide-open-throttle (WOT) acceleration.

Optimum Shift Point Algorithm

Explanation

Apdx-H

```
Do the following for every gear ratio and RPM:
        for i in every gearRatio (excluding the final gearRatio):
             minTorqueDrop = 9999999.0
             optimumShiftPt = The maximum RPM
             for x in every RPM from minRPM to maxRPM:
```

The optimum shift point algorithm works by looking at the torque produced at the current gear and RPM.

- currentGearTQ = torqueCurve[x][1] * gearRatios[i]

It then looks at what the torque would be if the user shifted to the next gear from the current RPM.

- nextGearTQ = torqueCurve[nextGearRPM][1] * gearRatios[i+1]

After the currentGearTQ and nextGearTQ have been calculated, it calculates the total torque drop.

- torqueDrop = currentGearTQ - nextGearTQ

Use this information to calculate a rolling minimum torqueDrop over all RPMs for the current gear. Every time a new minimum torqueDrop is found, keep track of the current RPM and torqueDrop

```
        if (abs(torqueDrop) < minTorqueDrop):
                minTorqueDrop = abs(torqueDrop)
                optimumShiftPt = torqueCurve[x][0]
```

If the torqueDrop is greater than zero, than it means that the optimum shift point has been found and there will be no other RPMs which will produce a smaller torqueDrop.

```
        if (torqueDrop > 0):
           break

     shiftPoints.append(optimumShiftPt)
```

The other main algorithm developed for Shift Buddy was the “Gear Shift Model” function. Shift Buddy will calculate the optimum shift points for your specific vehicle but there was one major issue, although Shift Buddy knows the shift point for gear N, how do you know if the vehicle is in gear N? The Shift Buddy directly communicates with the ECU (Engine Control Unit) through the OBD-II bus, however it doesn’t know any information about the transmission because the bus doesn’t directly communicate with the TCU (Transmission Control Unit). Shift Buddy solves this problem with the “Gear Shift Model”, when the users enter in gear ratio information for their transmission/differentials and the tire diameter information for their tires, the Shift Buddy can analyze the current RPMs the vehicle is traveling in correlation with the speed of the vehicle in order to do some backwards math and make an intelligent estimation of the current gear in real-time.

Gear Shift Model Explanation Apdx-I	● If the vehicle’s RPM is larger than zero but the car is moving, Shift Buddy will assume the vehicle is in 1st gear. This is due to being in park with the car started or having your foot on the break. ○ Vehicles generally idle at about 800-1200 RPMs ● The Shift Buddy now looks at the current gear ratio using the following equation: ○ PI * Tire Diameter(inches) / * Inches Per Minute * RPM ● This ratio will represent a value of what Shift Buddy considers to be the current gearing ratio dependent on your tire size, and how your vehicle is in motion. ● The algorithm now serially loops through all stored Overall Gear Ratio values and does a diff between the calculated ratio and each overall ratio in order to correlate to the value with the lowest delta. ● This correlation then allows Shift Buddy to assume what gear you’re in correlated with that specific value and index into the database to find the correct shift point.

Conclusion

This project was an exciting learning experience for me. It was a solo project in which I created an idea from scratch, did my own research, and iteratively turned an idea into something which I'm proud of. The Shift Buddy was one of the only projects in which I thoroughly enjoyed every part of development. Placing hours and hours into development and testing of the final project was brought about some awesome learning experiences. I learned quite a lot about the physics of vehicles in motion in order to create the "Gear Shift Model" and implement the "Optimum Shift Point Algorithm". I also learned quite a lot about interpolation and different mathematical techniques for assessing errors within an interpolation. Finally, I was able to integrate software and hardware and learned quite a lot of electrical engineering concepts when testing and programming the hardware used in the Shift Buddy. I've definitely planned out a list of greater, better ideas for the Shift Buddy which could not make it within the 7 week development timeframe, the Shift Buddy seems to be the first software application which I plan to properly maintain and develop because I've had such a great time creating it.

References

1. 8X Seven Segments Display. (n.d.). Retrieved on April 10, 2016 from the website: http://www.dx.com/p/8x-digital-tube-8x-key-8x-double-color-led-module-81873

2. Arduino Uno Rev 3. (n.d.). Retrieved on April 10, 2016 from the website: http://store-usa.arduino.cc/products/a000066

3. AutoMeter Level 1 External Pro Shift Lights. (n.d.). Retrieved on April 15, 2016 from the website: http://www.summitracing.com/parts/atm-5343/overview/

4. Freematics OBD-II Emulator MK2. (n.d.). Retrieved on April 10, 2016 from the website: http://freematics.com/store/index.php?route=product/product&product_id=71

5. Freematics Vehicle Data Logger V4. (n.d.). Retrieved on April 15, 2016 from the website: http://freematics.com/store/index.php?route=product/product&product_id=82

6. Kim, Stephen. (Oct 1, 2007). Learn About Dyno Testing - How Dynos Work. Retrieved on April 13, 2016 from the website: http://www.superchevy.com/how-to/0710ch-how-dynos-work/

7. Messersmith, Glenn. (2012). Optimal Shift Point. Retrieved on April 13, 2016 from the website: http://glennmessersmith.com/shiftpt.html

8. On/Off Programmable Digital Shift Light. (n.d.). Retrieved on April 14, 2016 from the website: http://www.autozone.com/1/products/44894-programmable-digital-shift-light-89631-msd-89631.html

9. Power under the curve. (July 25, 2015). Retrieved on April 14, 2016 from the website: http://oppositelock.kinja.com/power-under-the-curve-1714778421

10. Rev Matching & Down Shifting. (n.d.). Retrieved on April 17, 2016 from the website:

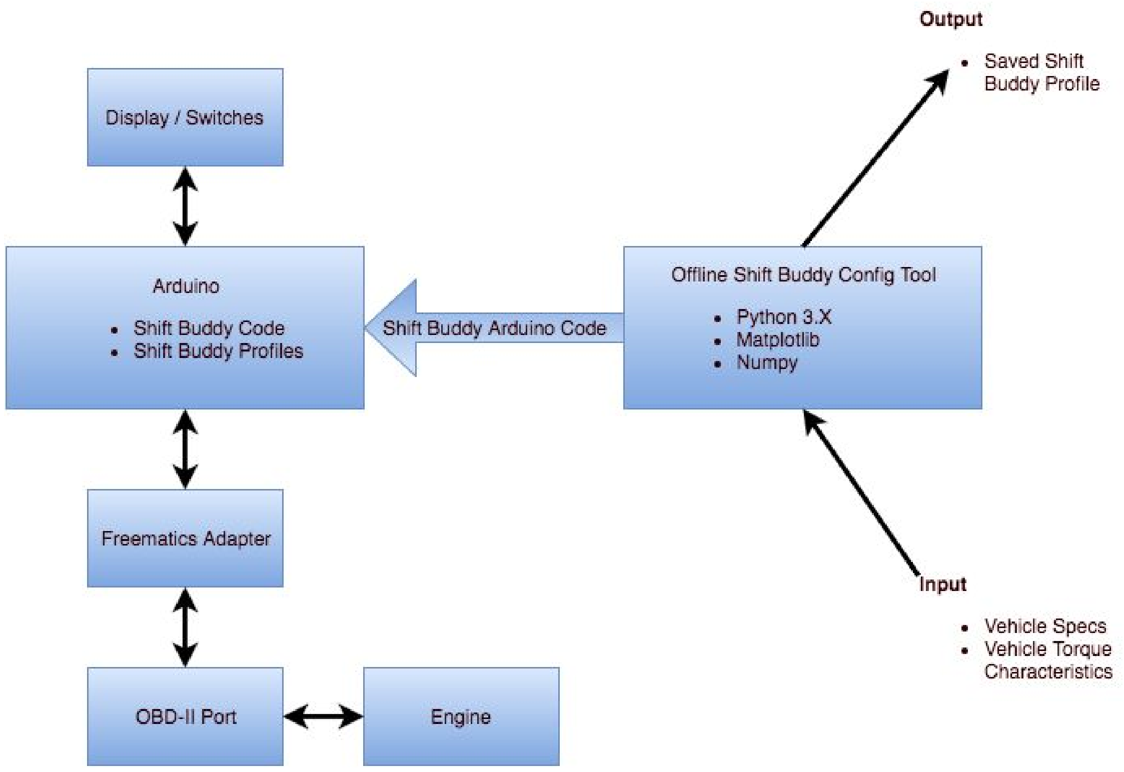
http://www.drivingfast.net/car-control/rev-matching.htm

11. What is a shift light. (n.d.). Retrieved on April 17, 2016 from the website:

http://www.wisegeek.com/what-is-a-shift-light.htm

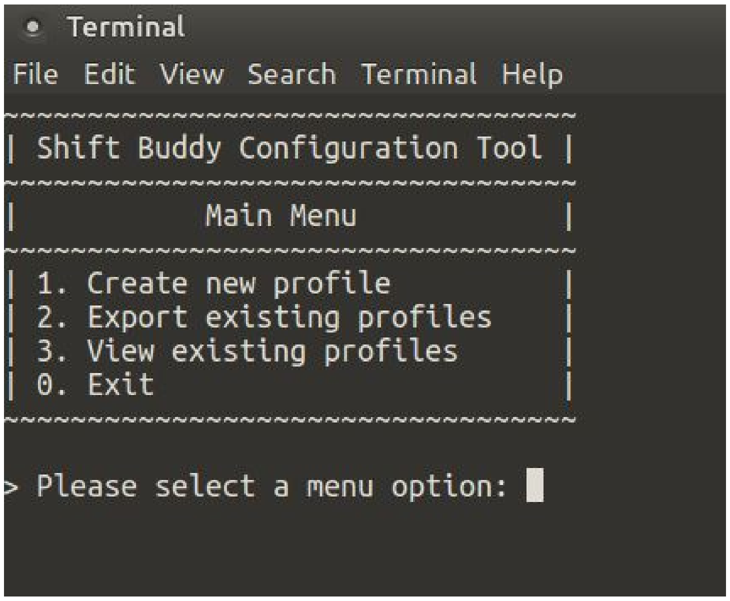
Appendix

Apdx-A

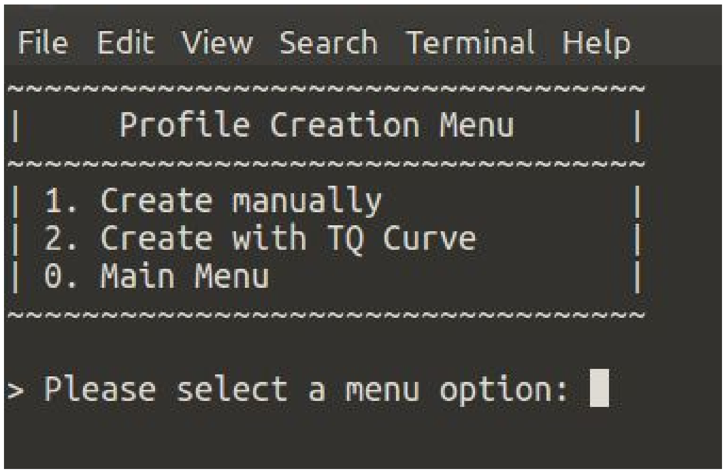


Functional breakdown of all Shift Buddy components (hardware and software).

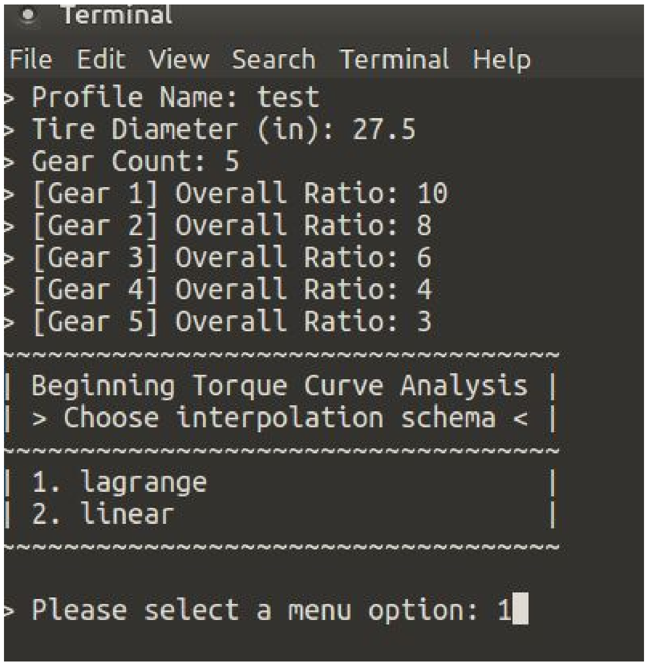
Apdx-B



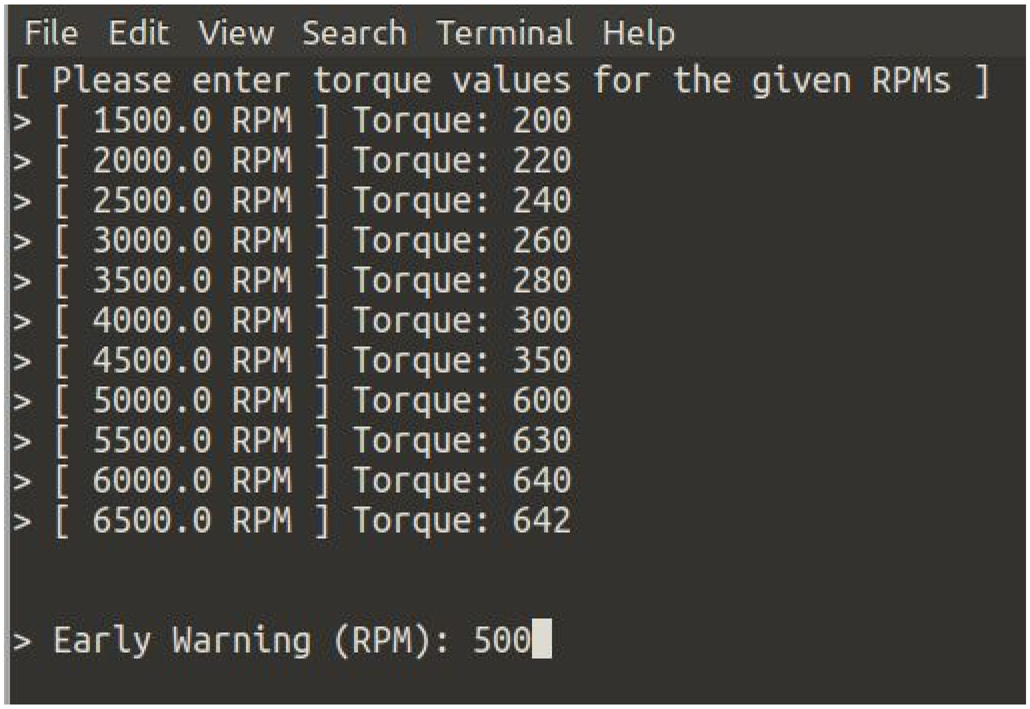
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| Shift Buddy Configuration Tool |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|           Main Menu            |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| 1. Create new profile          |
| 2. Export existing profiles    |
| 3. View existing profiles      |
| 0. Exit                        |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

> Please select a menu option:
```



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|     Profile Creation Menu      |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| 1. Create manually             |
| 2. Create with TQ Curve        |
| 0. Main Menu                   |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

> Please select a menu option:
```



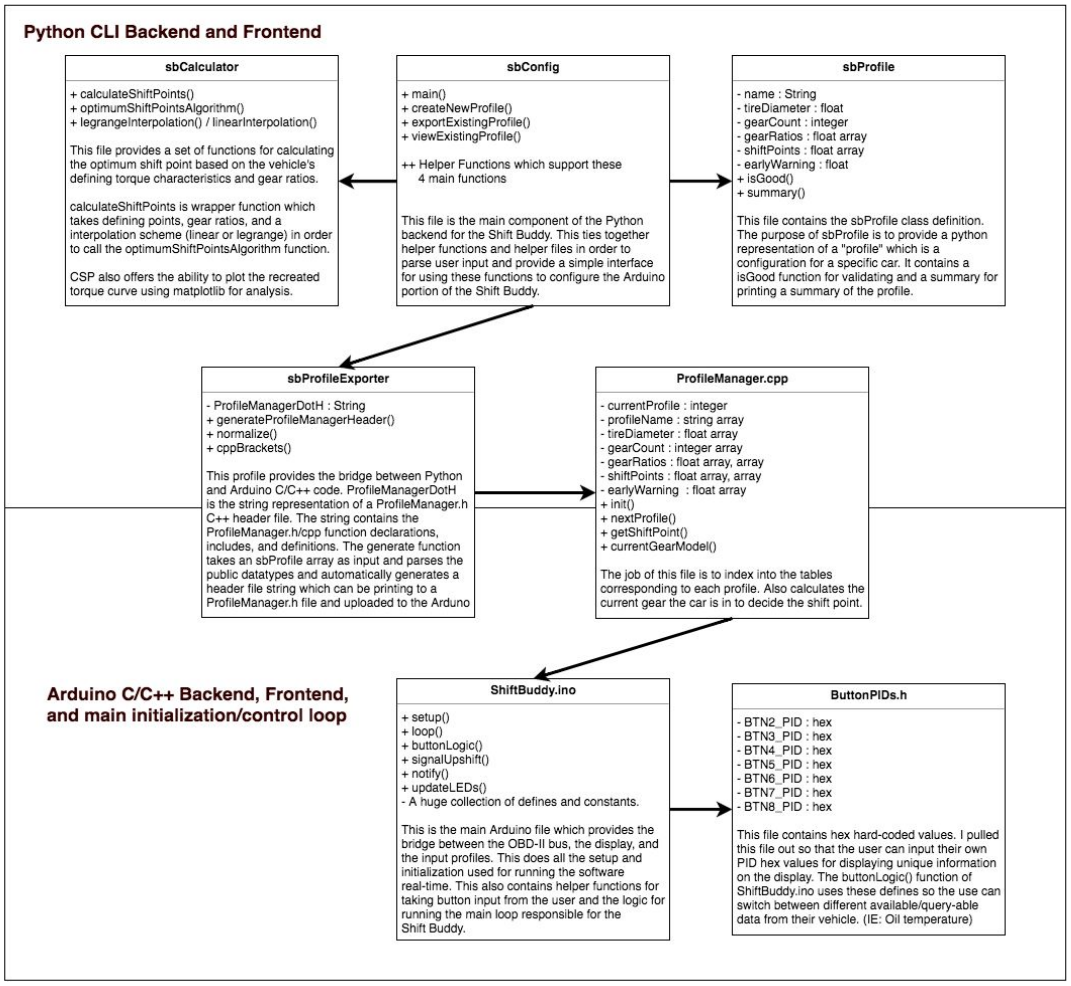
```
> Profile Name: test
> Tire Diameter (in): 27.5
> Gear Count: 5
> [Gear 1] Overall Ratio: 10
> [Gear 2] Overall Ratio: 8
> [Gear 3] Overall Ratio: 6
> [Gear 4] Overall Ratio: 4
> [Gear 5] Overall Ratio: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| Beginning Torque Curve Analysis |
| > Choose interpolation schema < |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
| 1. lagrange                     |
| 2. linear                       |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

> Please select a menu option: 1
```



```
[ Please enter torque values for the given RPMs ]
> [ 1500.0 RPM ] Torque: 200
> [ 2000.0 RPM ] Torque: 220
> [ 2500.0 RPM ] Torque: 240
> [ 3000.0 RPM ] Torque: 260
> [ 3500.0 RPM ] Torque: 280
> [ 4000.0 RPM ] Torque: 300
> [ 4500.0 RPM ] Torque: 350
> [ 5000.0 RPM ] Torque: 600
> [ 5500.0 RPM ] Torque: 630
> [ 6000.0 RPM ] Torque: 640
> [ 6500.0 RPM ] Torque: 642


> Early Warning (RPM): 500
```

Screenshots from the Shift Buddy Config Tool’s main menu and profile creation page.

Apdx-C


Python CLI Backend and Frontend
Arduino C/C++ Backend, Frontend, and main initialization/control loop

sbCalculator

+ calculateShiftPoints()
+ optimumShiftPointsAlgorithm()
+ legrangeInterpolation() / linearInterpolation()

This file provides a set of functions for calculating the optimum shift point based on the vehicle's defining torque characteristics and gear ratios.

calculateShiftPoints is wrapper function which takes defining points, gear ratios, and a interpolation scheme (linear or legrange) in order to call the optimumShiftPointsAlgorithm function.

CSP also offers the ability to plot the recreated torque curve using matplotlib for analysis.

sbConfig

+ main()
+ createNewProfile()
+ exportExistingProfile()
+ viewExistingProfile()

++ Helper Functions which support these
4 main functions

This file is the main component of the Python backend for the Shift Buddy. This ties together helper functions and helper files in order to parse user input and provide a simple interface for using these functions to configure the Arduino portion of the Shift Buddy.

sbProfile

- name : String
- tireDiameter : float
- gearCount : integer
- gearRatios : float array
- shiftPoints : float array
- earlyWarning : float
+ isGood()
+ summary()

This file contains the sbProfile class definition. The purpose of sbProfile is to provide a python representation of a "profile" which is a configuration for a specific car. It contains a isGood function for validating and a summary for printing a summary of the profile.

sbProfileExporter

- ProfileManagerDotH : String
+ generateProfileManagerHeader()
+ normalize()
+ cppBrackets()

This profile provides the bridge between Python and Arduino C/C++ code. ProfileManagerDotH is the string representation of a ProfileManager.h C++ header file. The string contains the ProfileManager.h/cpp function declarations, includes, and definitions. The generate function takes an sbProfile array as input and parses the public datatypes and automatically generates a header file string which can be printing to a ProfileManager.h file and uploaded to the Arduno

ProfileManager.cpp

- currentProfile : integer
- profileName : string array
- tireDiameter : float array
- gearCount : integer array
- gearRatios : float array, array
- shiftPoints : float array, array
- earlyWarning : float array
+ init()
+ nextProfile()
+ getShiftPoint()
+ currentGearModel()

The job of this file is to index into the tables corresponding to each profile. Also calculates the current gear the car is in to decide the shift point.

ShiftBuddy.ino

+ setup()
+ loop()
+ buttonLogic()
+ signalUpshift()
+ notify()
+ updateLEDs()
- A huge collection of defines and constants.

This is the main Arduino file which provides the bridge between the OBD-II bus, the display, and the input profiles. This does all the setup and initialization used for running the software real-time. This also contains helper functions for taking button input from the user and the logic for running the main loop responsible for the Shift Buddy.

ButtonPIDs.h

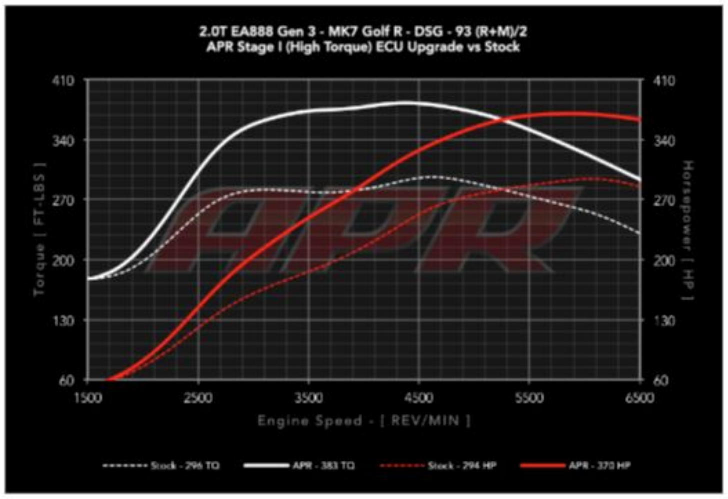
- BTN2_PID : hex
- BTN3_PID : hex
- BTN4_PID : hex
- BTN5_PID : hex
- BTN6_PID : hex
- BTN7_PID : hex
- BTN8_PID : hex

This file contains hex hard-coded values. I pulled this file out so that the user can input their own PID hex values for displaying unique information on the display. The buttonLogic() function of ShiftBuddy.ino uses these defines so the use can switch between different available/query-able data from their vehicle. (IE: Oil temperature)

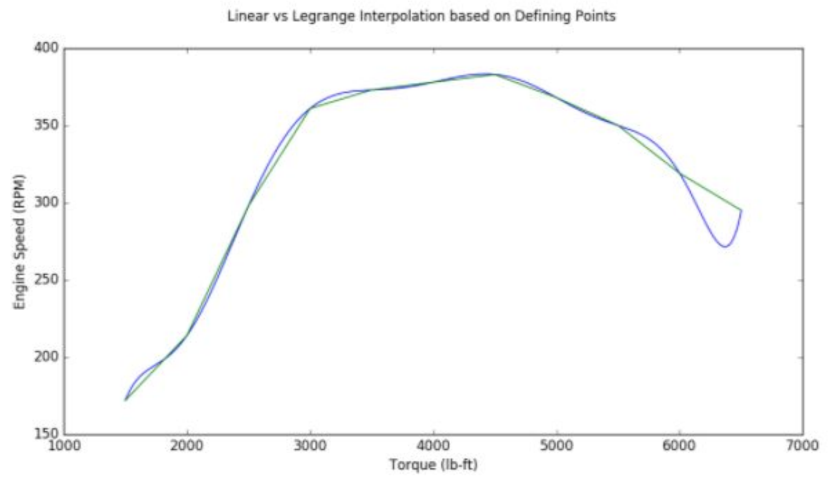
Software breakdown and flow chart of relational files. ButtonPIDs corrispond to the hexidecimal value which is sent to the OBD-II port to query for information.

Apdx-D

APR Tuning Plots - MK7 Golf R



Shift Buddy Interpolated Plots - MK7 Golf R

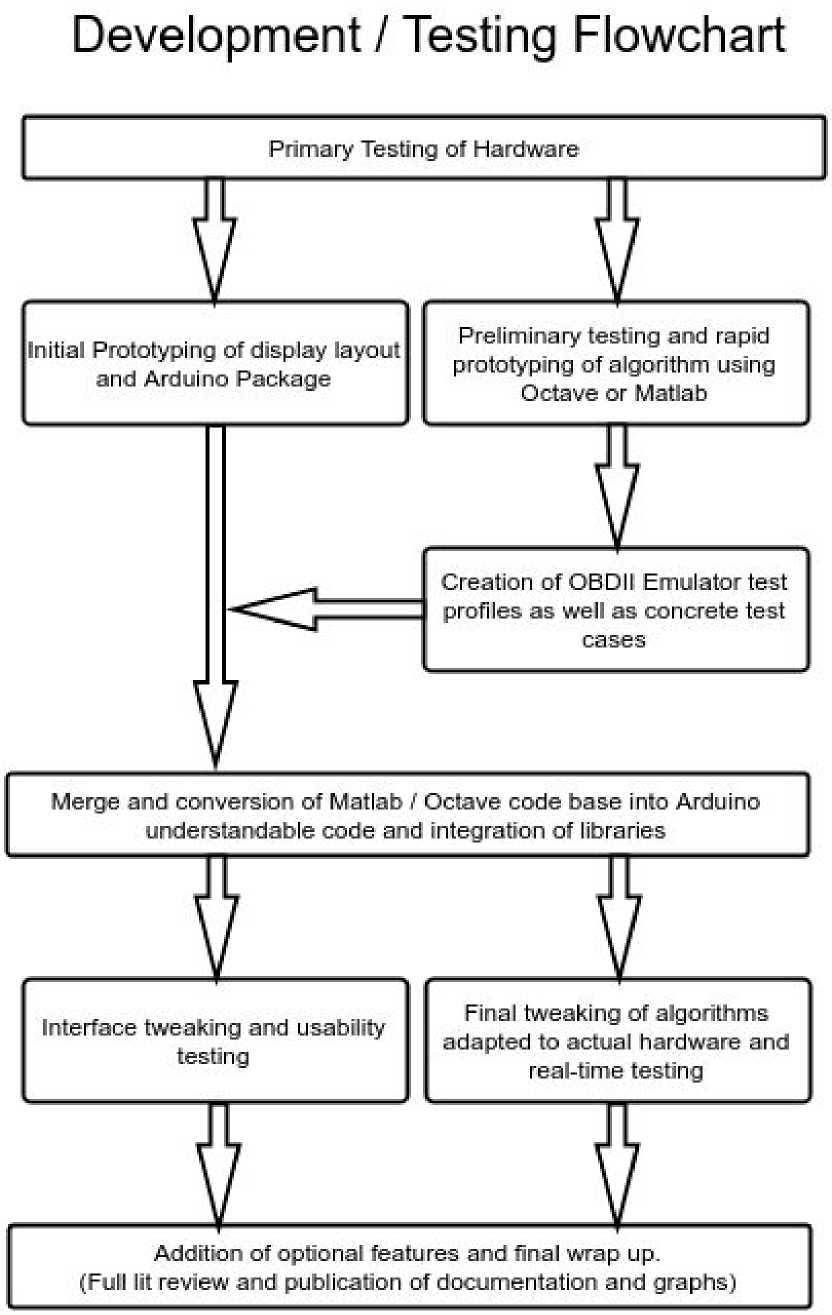


With the APR dynamometer charts as truth, the chart below represents how close of a representation the two interpolation schemes (linear and lagrange) were able to get in approximation of the real torque curve.

Apdx-E

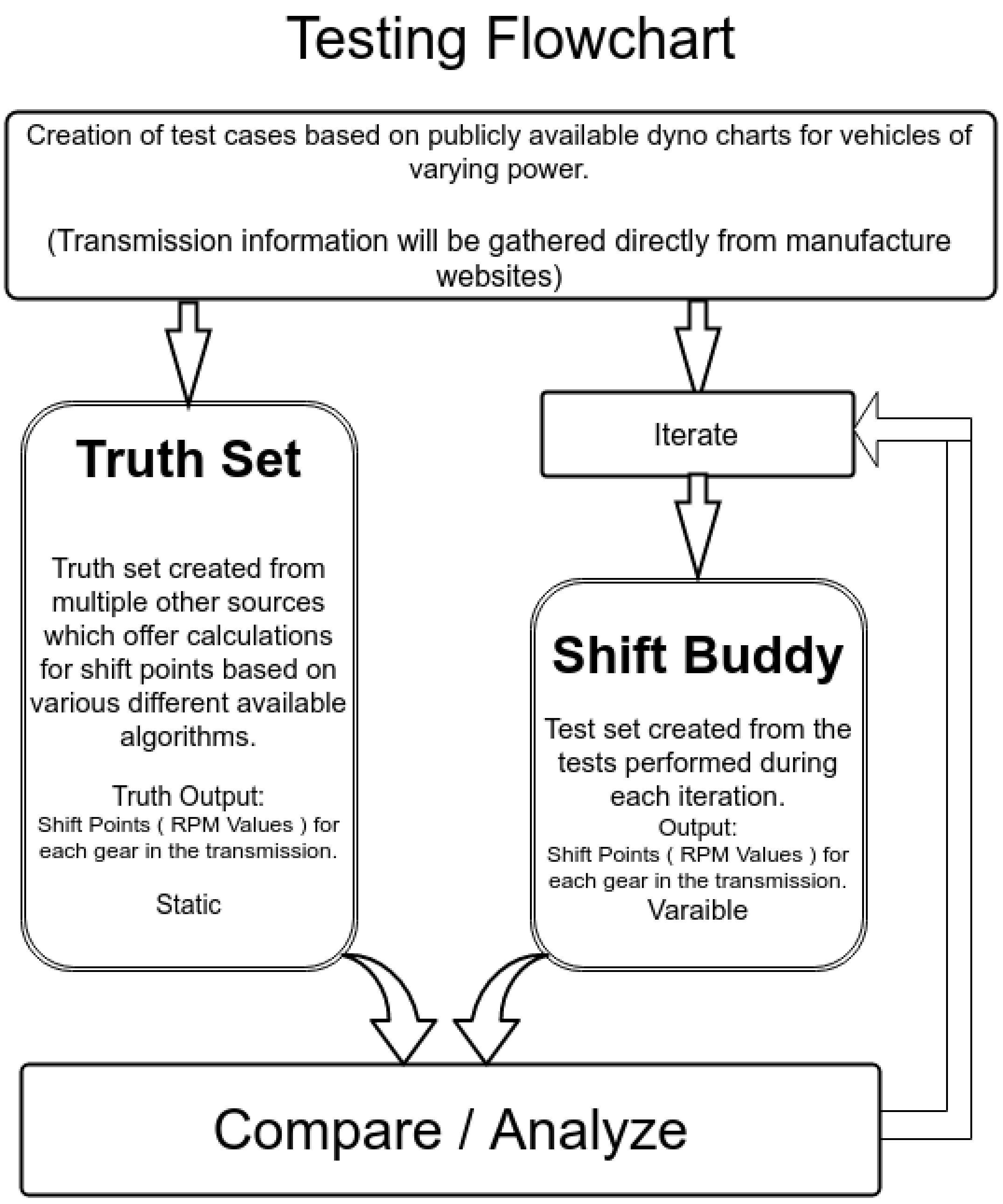
Testing unit (OBD-II Simulator) which simulates the OBD-II bus that talks to the onboard ECU. Knobs control different PIDs, red switch simulates engine ignition, and switches control message length and baud rate.

Apdx-F


Development / Testing Flowchart
Primary Testing of Hardware
Initial Prototyping of display layout and Arduino Package
Preliminary testing and rapid prototyping of algorithm using Octave or Matlab
Creation of OBDII Emulator test profiles as well as concrete test cases
Merge and conversion of Matlab / Octave code base into Arduino understandable code and integration of libraries
Interface tweaking and usability testing
Final tweaking of algorithms adapted to actual hardware and real-time testing
Addition of optional features and final wrap up. (Full lit review and publication of documentation and graphs)

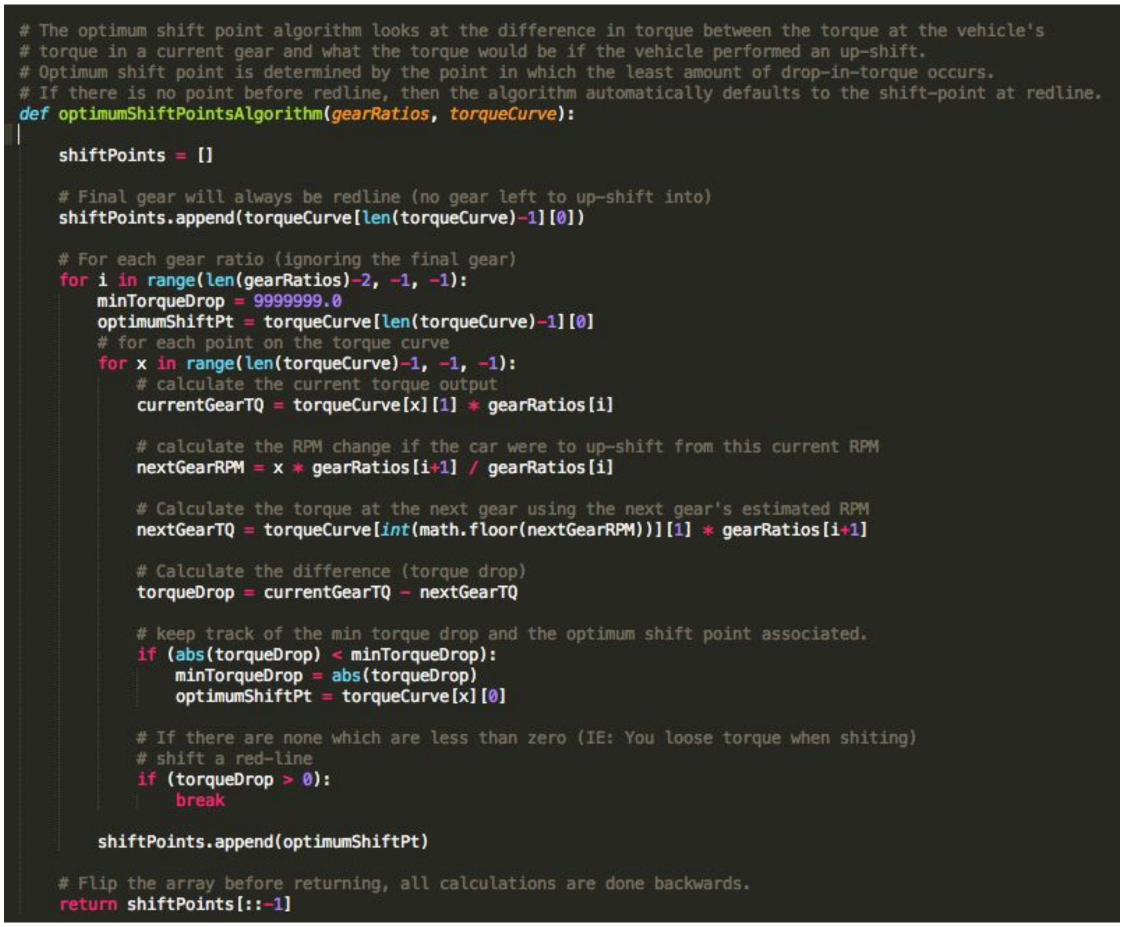
Developmental and testing flow chart / timeline for how the project started from the beginning of development towards the end.

Apdx-G


Testing Flowchart
Creation of test cases based on publicly available dyno charts for vehicles of varying power.
(Transmission information will be gathered directly from manufacture websites)
Truth Set
Truth set created from multiple other sources which offer calculations for shift points based on various different available algorithms.
Truth Output:
Shift Points (RPM Values) for each gear in the transmission.
Static
Iterate
Shift Buddy
Test set created from the tests performed during each iteration.
Output:
Shift Points (RPM Values) for each gear in the transmission.
Varaible
Compare / Analyze

Flow chart for how all testing and iterative work was done during the developmental process.

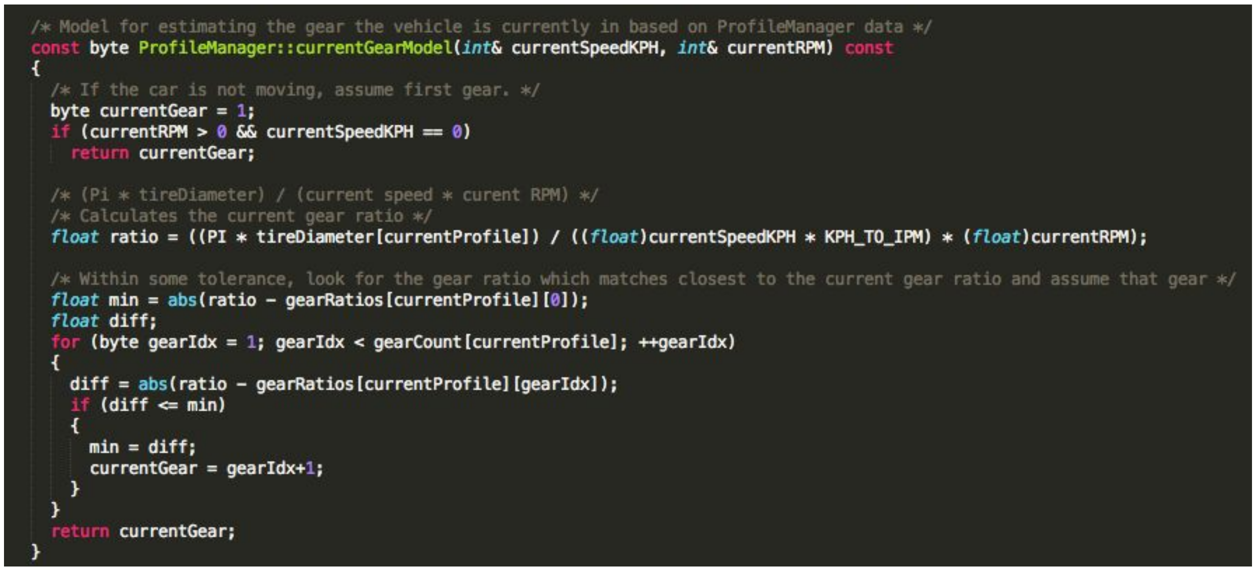
Apdx-H



```python
# The optimum shift point algorithm looks at the difference in torque between the torque at the vehicle's
# torque in a current gear and what the torque would be if the vehicle performed an up-shift.
# Optimum shift point is determined by the point in which the least amount of drop-in-torque occurs.
# If there is no point before redline, then the algorithm automatically defaults to the shift-point at redline.
def optimumShiftPointsAlgorithm(gearRatios, torqueCurve):

    shiftPoints = []

    # Final gear will always be redline (no gear left to up-shift into)
    shiftPoints.append(torqueCurve[len(torqueCurve)-1][0])

    # For each gear ratio (ignoring the final gear)
    for i in range(len(gearRatios)-2, -1, -1):
        minTorqueDrop = 9999999.0
        optimumShiftPt = torqueCurve[len(torqueCurve)-1][0]
        # for each point on the torque curve
        for x in range(len(torqueCurve)-1, -1, -1):
            # calculate the current torque output
            currentGearTQ = torqueCurve[x][1] * gearRatios[i]

            # calculate the RPM change if the car were to up-shift from this current RPM
            nextGearRPM = x * gearRatios[i+1] / gearRatios[i]

            # Calculate the torque at the next gear using the next gear's estimated RPM
            nextGearTQ = torqueCurve[int(math.floor(nextGearRPM))][1] * gearRatios[i+1]

            # Calculate the difference (torque drop)
            torqueDrop = currentGearTQ - nextGearTQ

            # keep track of the min torque drop and the optimum shift point associated.
            if (abs(torqueDrop) < minTorqueDrop):
                minTorqueDrop = abs(torqueDrop)
                optimumShiftPt = torqueCurve[x][0]

            # If there are none which are less than zero (IE: You loose torque when shiting)
            # shift a red-line
            if (torqueDrop > 0):
                break

        shiftPoints.append(optimumShiftPt)

    # Flip the array before returning, all calculations are done backwards.
    return shiftPoints[::-1]
```

sbCalculator.py: optimumShiftPointAlgorithm

Apdx-I



```
/* Model for estimating the gear the vehicle is currently in based on ProfileManager data */
const byte ProfileManager::currentGearModel(int& currentSpeedKPH, int& currentRPM) const
{
  /* If the car is not moving, assume first gear. */
  byte currentGear = 1;
  if (currentRPM > 0 && currentSpeedKPH == 0)
    return currentGear;

  /* (Pi * tireDiameter) / (current speed * curent RPM) */
  /* Calculates the current gear ratio */
  float ratio = ((PI * tireDiameter[currentProfile]) / ((float)currentSpeedKPH * KPH_TO_IPM) * (float)currentRPM);

  /* Within some tolerance, look for the gear ratio which matches closest to the current gear ratio and assume that gear */
  float min = abs(ratio - gearRatios[currentProfile][0]);
  float diff;
  for (byte gearIdx = 1; gearIdx < gearCount[currentProfile]; ++gearIdx)
  {
    diff = abs(ratio - gearRatios[currentProfile][gearIdx]);
    if (diff <= min)
    {
      min = diff;
      currentGear = gearIdx+1;
    }
  }
  return currentGear;
}
```

ProfileManager.cpp: currentGearModel