EFI Retrofit for Small Engines

A Baccalaureate thesis submitted to the
Department of Mechanical and Materials Engineering
College of Engineering and Applied Science
University of Cincinnati

in partial fulfillment of the
requirements for the degree of

Bachelor of Science

in Mechanical Engineering Technology

by

Ian Lind

April 2021

Thesis Advisor:

Professor Amir Salehpour

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

For my senior design project, I decided to build an electronic fuel injection (EFI) system to retrofit on older small engines to help increase their performance and efficiency. The project started with my desire to do something automotive related as a senior design project with the goal of applying that knowledge later on in life. I already had a healthy understanding of most automotive related systems so with that I dedicated that an EFI system best fit the requirement of an automation or robotic project associated with the Robotics and Automation Minor. The design of the EFI system followed what I understood about engine electronics and what sensors are required for an engine to run combined with an immense amount of research as to how to integrate that with a hobbyist microcontroller.

From the start of building the EFI system it was clear that the amount of time spent on research paid dividends as most of the coding and construction went relatively easily if a bit time consuming. Until the first attempt to start the engine everything was working as expected. That is when the troubleshooting began to take an enormous amount of time. With this project being something that does not have well documented history I had a lot of trial and error work to ensure it ran. After the many problems were solved with design alterations or part changes the engine ran very well and tested impressively compared to its carbureted counterpart.

PROBLEM DEFINITION AND RESEARCH

PROBLEM STATEMENT

I will be addressing an issue plaguing many old engines, the lack of efficiency from their fuel delivery. Almost every four-stroke gasoline engine, except those made for automobile use in the last 40 years, uses a carburetor to atomize fuel and control the ratio of fuel to air entering the engine. I intend to replace the carburetor with a closed-loop electronically controlled fuel injection system in an effort to increase efficiency and reduce emissions in a broad range of conditions.

RESEARCH

BACKGROUND AND SCOPE OF THE PROBLEM

Currently most engines in uses other than automotive, utilize a carburetor rather than electronic fuel injection [EFI] to atomize fuel and control the amount of air entering the engine. Carburetors are used because they are inexpensive and rather simple mechanical devices, but one issue inherent to carburetors is that they are inefficient. As a carburetor is an open loop system it is incapable of making adjustments and thus is only capable of being tuned for efficiency in a very narrow operating window. This lack of efficiency results in higher emissions and higher fuel usage. Under perfect stoichiometric operating conditions, an internal combustion takes gasoline and air and emits exhaust consisting of equal parts carbon dioxide and water at 13% each, with nitrogen from air comprising most of the 73% left (1), however in the real world engines can't run at stoic for a variety of reasons which leads to emissions of carbon (soot), carbon monoxide and other greenhouse gasses.

People impacted by the use of carburetors instead of EFI would be the anyone using or who is near small equipment powered by a gasoline engine. This would be anyone from construction worker to yard workers to people mowing their lawns or power washing their sidewalk. Poor emissions due to the use of carburetors not only effects people but also impacts every animal and plant in the immediate area and to a much smaller degree, all over the earth. The widespread use of carburetors also contributes to an increased use of gasoline as carburetors are not as efficient as EFI. According to Miller a welder manufacturer, the use of EFI in one their generators resulted in 150% longer runtime given the same conditions as their carbureted version (2). As the air fuel ratio diverges from stoichiometric to rich (excess fuel) hydrocarbon and carbon monoxide emissions greatly increase, as much as 4 times with only a 20% change (3).

Of the main emissions from an internal combustion engine carbon monoxide is the most harmful to life. Carbon monoxide is a colorless odorless gas that is harmful to humans if inhaled. According to the NHS "Prolonged exposure to carbon monoxide can cause memory problems and difficulty concentrating." (4). Internal combustion engines can be a real danger when in enclosed environments as the CDC states that "… small gasoline-powered engines and tools present a serious health hazard. They produce high concentrations of [carbon

monoxide] …" (5).

Currently the problem of poor emissions from carburetors is being addressed by the EPA and other environmental agencies around the world who are pushing for tighter regulations on new engines, as in 2011 and 2012 the EPA introduced their "Phase 3 exhaust emissions standard" (6) for small equipment and tools. This makes manufactures either improve the efficiency of their carburetors or use an EFI system, but unfortunately it doesn't do much for older equipment. There are currently a few companies such as Edelbrock, Holly, and MSD that make aftermarket EFI and engine control unit [ECU] solutions aimed at older engines however these are largely meant for automotive application and thus have much more functionality than would be needed leading to increased cost.

CURRENT STATE OF THE ART

The problem of poor efficiency and emissions from carburetors is currently being tackled by a few companies, such as Holly who make an aftermarket EFI and engine control unit [ECU] solution which is aimed at older engines. However, these are largely meant for automotive applications and specifically target the older American V8 market. The Terminator X system from Holly allows the user to add many sensors to increase the closed loop performance benefits. This system has the ability for the computer to use the closed loop sensors to provide a self-tuning function. These automotive grade solutions would be extraordinarily expensive for the application of small engines removing most of the cost benefit, as even Holley's least expensive universal kit is over $1100 (7). These systems have the potential to be adapted to small engines but could possibly be expecting a number of sensors that may be unnecessary or unavailable for smaller engines.

There currently aren't any products being offered to retrofit EFI onto older gasoline small engine equipment, however there are a few concepts being developed. Walbro is one of the largest carburetor and OEM EFI system manufactures, and at the GIE+EXPO in 2015 they demonstrated a prototype version of their carburetor conversion system called Electronic Engine Management. This system is relatively easy to setup as it has most of the necessary sensors contained within the one unit. The Walbro Electronic Engine Management also offers onboard diagnostics to make repairs easier. Unfortunately as of Q3 2020 there is no information on when this product will debut, how expensive it will be, how easy it will be to install, or whether or not it actually works. As of now it is also not known how much the system would cost but based on market value it would be reasonable to assume around $1000 or higher. (8)

Another product aimed at providing EFI to older small engines is the NanoEFI. This is a crowdfunded enthusiast headed attempt at creating an aftermarket EFI system to replace carburetors. The NanoEFI has a proprietary enthusiast-style PCB meaning the developer can include any features they deem important. This system has a built in 802.11 access point to allow for wireless tuning as well as data display and logging. The NanoEFI is also designed to bring EFI to people at an affordable price which is in contrast to most other systems which

are not very budget minded. However, the development of the NanoEFI is focused on motorcycles which means that it may have a tough time adapting to power-equipment small engines as their operating conditions are much different. Though there are many benefits to a crowdfunded project, this one still in the Alpha stage of development so there is no guarantee that it can deliver on any of its promises. (9)

END USER

The end user of my EFI Retrofit kit would be someone looking to improve the efficiency or emissions of their small engine equipment. More specifically this product would target two distinct groups, professional lawncare companies with a fleet of older machinery and lawn care enthusiasts interested in improving one or relatively few engines. The professional lawncare companies would have a fleet of carbureted equipment where an improvement in emissions would create a safer work environment, and an increase in efficiency could provide a real monetary value before they replace their equipment. The lawn care enthusiasts would likely either be interested in getting more performance from their current equipment or concerned with the safety or environmental impact of their equipment's exhaust emissions.

CONCLUSIONS AND SUMMARY OF RESEARCH

Based on my research, it seems that the market has not yet realized the benefit of creating an EFI system for small engines. Though there are systems in development for small engine EFI, they appear to be years away from production if not likely to never make it to market. Currently the systems offered to replace a carburetor with EFI are all either too expensive to monetarily justify, too complex to reasonably be fitted, or too difficult to install onto small engines. This leaves a large group of people who operate power-equipment powered by gasoline engines without any way to upgrade, forcing them to either buy something new or live with the drawbacks of using a carburetor.

QUALITY FUNCTION DEPLOYMENT

CUSTOMER FEATURES

The features that I chose for my survey were fuel efficiency, engine power, exhaust emissions, ease of installation or maintenance, and ease of use. When making my survey I made sure to include intended use case as a way to better understand needs of populations based on use case. I found single customers, as in people who only mow their own lawn but may have a large lawn, to be interested in power and ease of installation and their interest would be contingent on the product being less than about $500 relegating my product as a niche project for an enthusiast but not really marketable to a wide audience. I found the professional lawn care audience to be a lot more interested in fuel economy and power than anything else and their interest in my product extended to the $1000 range and but dependent on a reasonable return on investment timetable. The fleet mechanics, as could be expected

were largely concerned with the ease of use and ease of installation.

ENGINEERING CHARACTERISTICS

The engineering characteristics that I chose were, fuel flow at maximum RPM to measure fuel efficiency which while not giving a definite answer can be evaluated relative to the carburetor to give a percentage change. Power in horsepower will also be an engineering characteristic to be judged relative to the carburetor. Time required to install both a carburetor and the EFI system will be evaluated to give a reasonable understanding of the complexity to the end user. O2 levels measured after combustion will be used to measure the exhaust emissions. Finally, time the engine takes to start form cold will be used as it is good measurement of ease of use as that's something you have to do every time you use your engine.

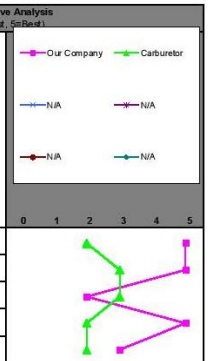
HOUSE OF QUALITY

Title: EFI Retrofit for Small Engines
Author: Ian Lind
Date: 10/2/2020
Notes:

Legend
- Strong Relationship 9
- Moderate Relationship 3
- Weak Relationship 1
- Strong Positive Correlation
- Positive Correlation
- Negative Correlation
- Strong Negative Correlation
- Objective Is To Minimize
- Objective Is To Maximize
- Objective Is To Hit Target

Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Fuel Flow at Max RPM (liters/minute)	Power (horsepower)	Time Required to Install (hours)	O2 Ratio after Combustion (lambda)	Time to Start (seconds)	Our Company	Carburetor	N/A	N/A	N/A	N/A
Column #					1	2	3	4	5						
Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)					▼	▲	▼	x	▼						
1	9	21.5	21.5	Be Fuel Efficient	⊙	O		O		5	2				
2	9	23.5	23.5	Be Powerful	O	⊙		O		5	3				
3	9	20.5	20.5	Easy to Install/Maintain			⊙		▲	2	3				
4	9	13.5	13.5	Emit exhaust safely	O	O		⊙		5	2				
5	9	21.0	21.0	Easy to use			▲		⊙	3	2				
6															
7															
8															
9															
10															
Target or Limit Value					Minimize	Maximize	Minimize	Target	Minimize						
Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)					6	5	8	4	5						
Max Relationship Value in Column					9	9	9	9	9						
Weight / Importance					304.5	316.5	205.5	256.5	209.5						
Relative Weight					23.6	24.5	15.9	19.8	16.2						

Competitive Analysis (0=Worst, 5=Best)



Powered by QFD Online (http://www.QFDOnline.com)

Figure 1 – House of Quality

PRODUCT OBJECTIVES

The weighted product objectives from the house of quality came out to be relatively similar showing no real favorite. The minimization of fuel flow at max RPM (measured in liters per minute) was weighted at 23.6% making it the second most important. For maximization of power (measured in horsepower) it is weighted at 24.5% making it the most important objective. Time required to install minimization was determined be weighted at 15.9%. Hitting the target O2 ratio after combustion will be weighted at 19.8%. And finally minimizing the time to start will be weighted at 16.2%.

DESIGN

DESIGN ALTERNATIVES AND SELECTION

Concept One – Simple Throttle Body

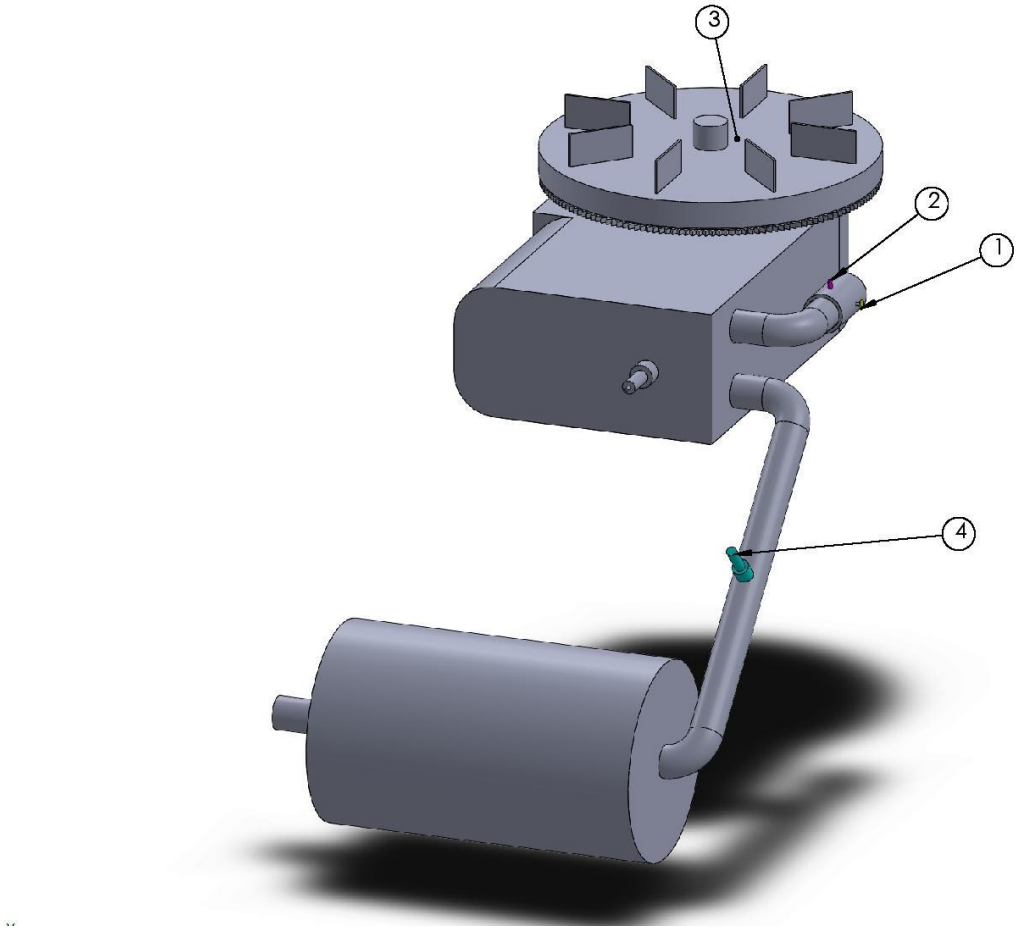


Figure 2 – Concept 1 Simple Throttle Body

#	**Part Description**
1	Throttle Position Sensor (TPS)
2	Fuel Injector
3	Stator (RPM Sensor)
4	O2 Sensor

Table 1 – Concept 1 Part Description

For my first concept I decided to take a more simple approach in an effort to reduce the time required to install. For this concept I will be using a throttle body to meter air flow, with a fuel injector attached to the back of it. With this setup I am using an Arduino Uno as the

brains of my EFI system. To calculate the RPM signal, I will be using information from the stator which should reduce install time. The stator, located underneath the flywheel, is the charging device for the mower and works by waving a magnet past coiled wires, this creates pulses of electricity that increase in frequency in direct correlation with engine RPM. I will also use an encoder to sense throttle position telling the computer very accurately to what degree the throttle is open. The TPS works in conjunction with the RPM sensor to get a base fueling value. To inject fuel, I will be using a pulse width modulated fuel injector. Pulse width modulation is the most common way of controlling how much fuel is injected into an engine, it works by varying the amount of time the injector is spraying rather than the specific amount it sprays. I am using an O2 sensor to provide closed loop feedback to tell how efficient the combustion was.

Concept Two – Complex Throttle Body

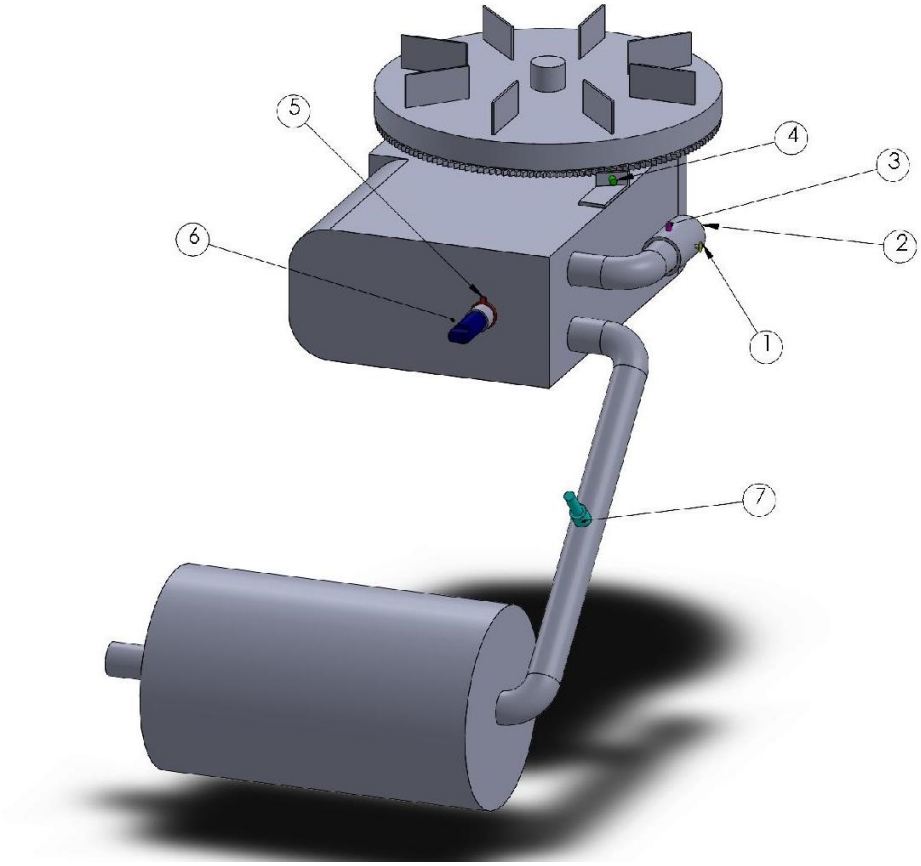


Figure 3 – Concept 2 Complex Throttle Body

#	**Part Description**	#	**Part Description**
1	Throttle Position Sensor (TPS)	5	Temperature Sensor
2	Manifold Absolute Pressure Sensor (MAP)	6	Coil on Plug Ignition (COP)
3	Fuel Injector	7	O2 Sensor
4	Hall Effect Sensor (RPM Sensor)		

Table 2 – Concept 2 Part Description

For this concept I decided to make the EFI system more integrated, almost doubling the amount of information the computer has to work with. Like the first concept, I will be using a throttle body with a TPS to measure the throttle inputs and a PWM fuel injector to control fuel flow. However, there will also be a manifold absolute pressure sensor attached to the throttle body to determine air density which will aid in efficiency. As the computational system of the EFI, I will be using an Arduino Due to control everything. The Arduino Due has considerably more computational power than the Arduino Uno as well as an increased number of inputs and outputs. To find RPM signal I will be using a hall effect sensor. The hall effect sensor uses a magnetic field to sense when each tooth on the flywheel goes by giving a very accurate engine speed measurement. I will also be using a thermocouple placed between the engine head and spark plug to measure the temperature of the engine to increase fuel efficiency and power. This concept also makes use of a coil on plug ignition system which allows for the spark to be computer controlled increasing efficiency and power. This setup will also make use of an O2 sensor for closed loop feedback to increase efficiency.

Concept Three – Carburetor Throttle Control

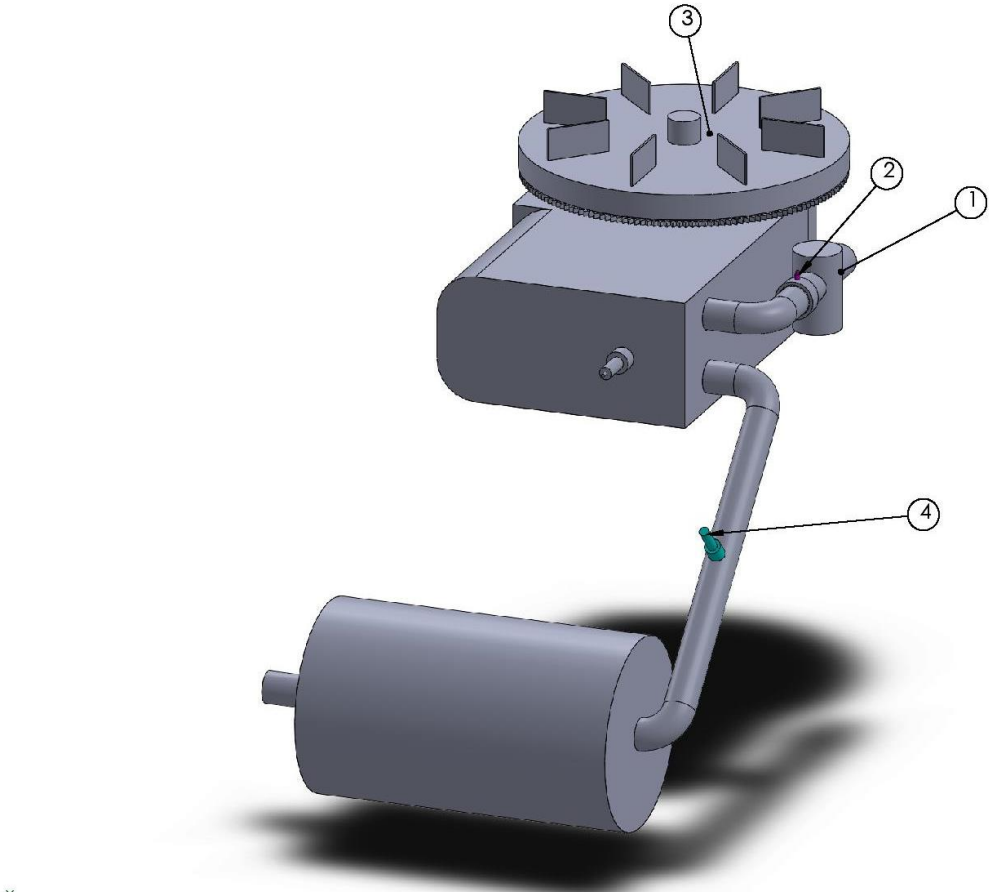


Figure 4 – Concept 3 Carburetor Throttle Control

#	**Part Description**
1	Throttle Position Sensor attached to Carburetor (TPS)
2	Fuel Injector
3	Stator (RPM Sensor)
4	O2 Sensor

Table 3 – Concept 3 Part Description

In this concept much of the setup will remain unchanged from concept one, however I will be using the carburetor rather than a throttle body to control the air entering the engine. While an injector still sprays the fuel, using the carburetor to meter air will reduce the amount of parts that will be needed, hopefully reducing installation time and costs. The throttle position would be sensed by a linear displacement sensor rather than an encoder as the carburetor doesn't have provisions for an encoder. This concept will also make use of an O2 sensor for closed loop feedback and the stator for sensing RPM.

Concept Selection

For the concept selection, I decided to use the same engineering criteria, weighted the same as I used in my House of Quality. Fuel efficiency, power, install time, O2 ratio, and time to start will be the evaluation criteria. From figure 5 it is clear that the second concept outscored both the first and third by quite a bit. The only area where it wasn't the clear favorite was in install time where it was the worst. I believe that even though it has theoretically the longest install time it won't be long enough to severely impact the end user.

		Concepts					
		Option 1		Option 2		Option 3	
Criteria	Importance Weight (%)	Rating	Weighted	Rating	Weighted	Rating	Weighted
Fuel	23.6	2	0.472	3	0.708	1	0.236
Power	24.5	2	0.49	3	0.735	1	0.245
Install Time	15.9	2	0.318	1	0.159	3	0.477
Efficiency	19.8	2	0.396	3	0.594	1	0.198
Time to Start	16.2	2	0.324	3	0.486	1	0.162
	100	Weighted Total	2	Weighted Total	2.682	Weighted Total	1.318

Rating	Value
Worst	1
Middle	2
Best	3

Figure 5 – Concept Selection Matrix

MANUFACTURING DRAWINGS

Design Concept

In figure 6 you can see the concept design for this EFI retrofit system in more detail. From this picture it is easier to discern how the system is to work. With a throttle body to control the air entering the engine and an integrated TPS this concept follows a conventional approach to fuel injection. There are multiple sensors to allow this system to adjust to outside conditions. The MAP sensor allows the system to adjust for altitude as well as provide a mass air flow entering the engine which helps aid in efficiency. This engine also has a cylinder head temperature sensor which allows the system to run closer to its thermal limits while also providing a better experience when the engine is cold. There is also an RPM sensor triggered by the teeth on the flywheel which allows the computer to, in conjunction with the TPS, determine proper initial fueling values. This system has a coil on plug ignition system which allows for the spark timing to be computer controlled increasing efficiency and power. This system will also use an O2 sensor for closed loop combustion efficiency feedback which will increase overall efficiency.

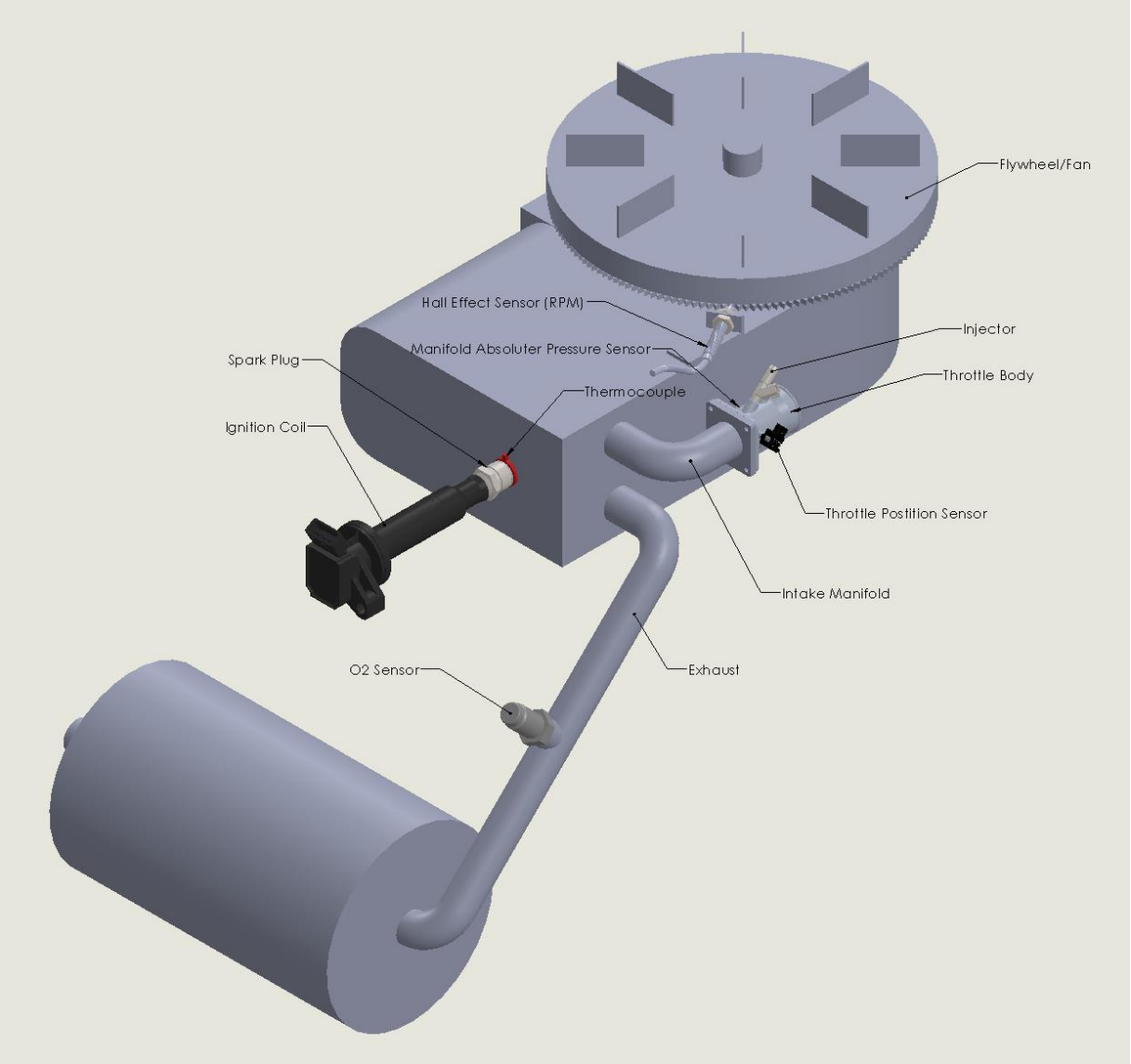


Figure 6 – Concept Design

Wiring Diagram

A large portion of the work in adding EFI to any engine is the wiring that needs to be done. In figure 7 there is a preliminary wiring diagram for all components to the Arduino Due. This wiring diagram makes use of MOSFETs and diodes for triggering the fuel injector as well as the ignition coil as their trigger voltage is higher than the Arduino can output. The Arduino Due is a 3.3V microcontroller meaning that it can only receive an input of up to 3.3 volts, thus I have included a logic level converter which can take digital signals from 5 volts down to 3.3 volts protecting the Arduino Due. The O2 sensor makes use of a heater element and therefore needs 12 volt power from the battery to make readings.

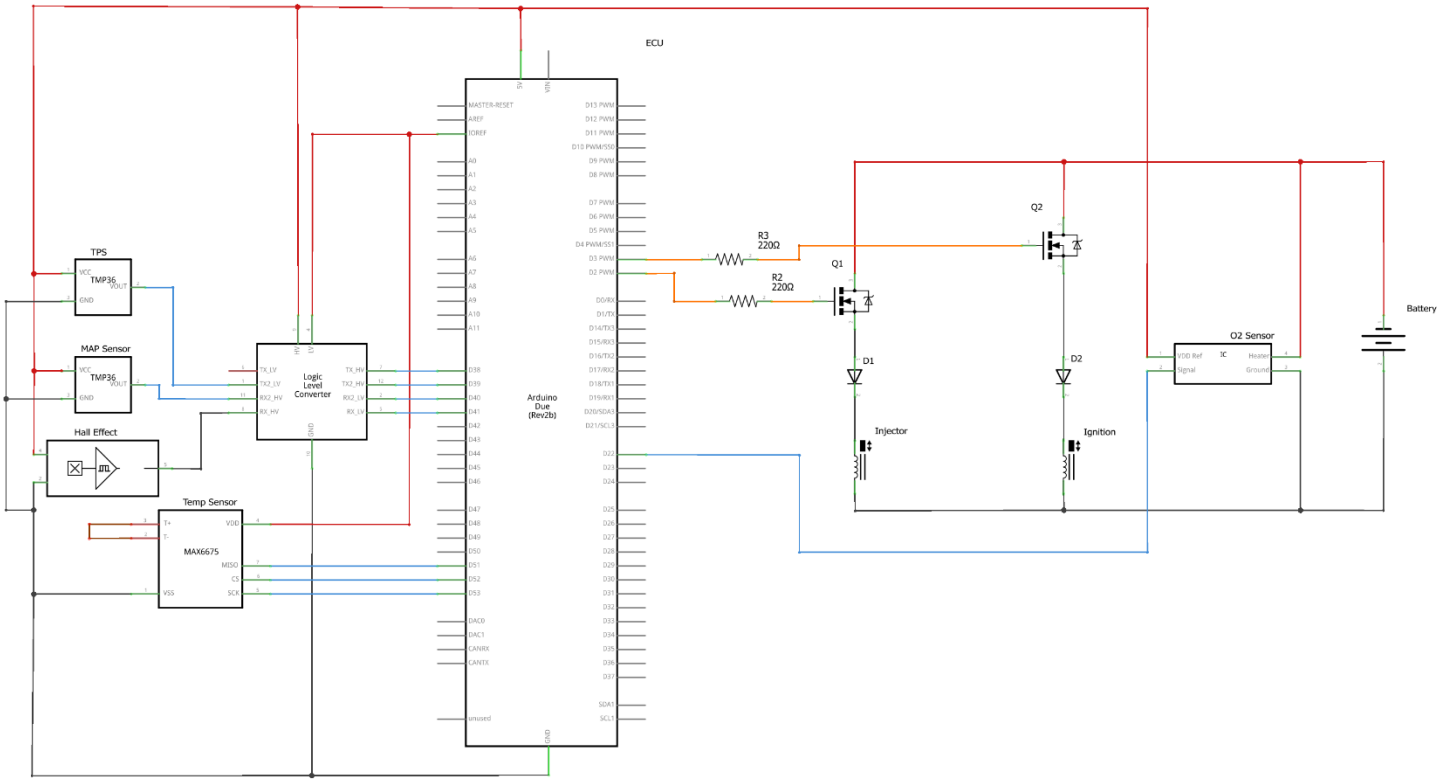


Figure 7 – Wiring Diagram

Exhaust

In figure 8 you can see the exhaust that already exists on the mower with an O2 sensor added to the pipe. The O2 sensor should be added roughly 4 inches from the exhaust port to make accurate readings. The O2 sensor is threaded M18x1.5 and screws into a fitting welded onto the exhaust. This is the only portion of the EFI system that provides the closed loop feedback so it is critical that it works correctly.

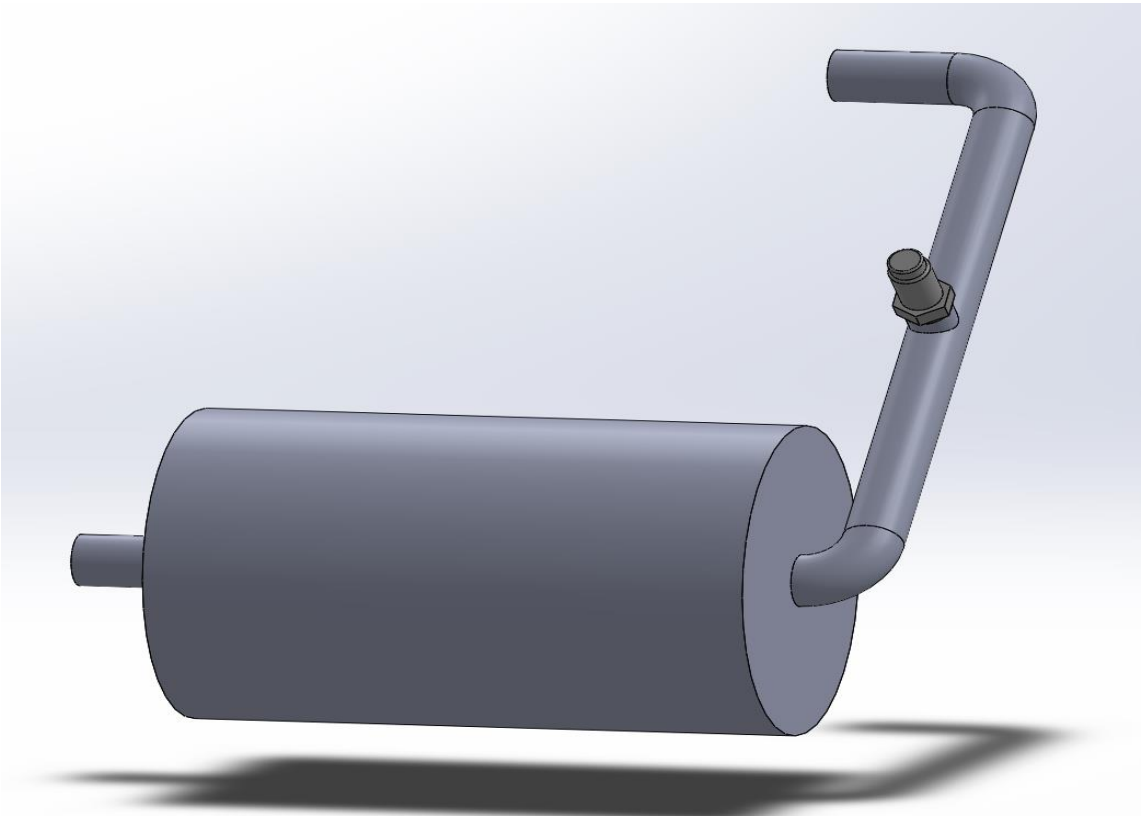


Figure 8 – Exhaust Modified

Flywheel

The flywheel on most mowers is what interfaces with the electric starter and thus has teeth machined on the bottom. In figure 9 you can see how the flywheel and a hall effect RPM sensor will interface. The hall effect sensor is much like an inductive sensor in that a disturbance in its magnetic field produces an output, however unlike the an inductive sensor the hall sensor outputs a digital signal making it capable of interfacing with a logic level converter. The hall sensor will read the teeth on the flywheel as they go by which will give it a very accurate RPM resolution.

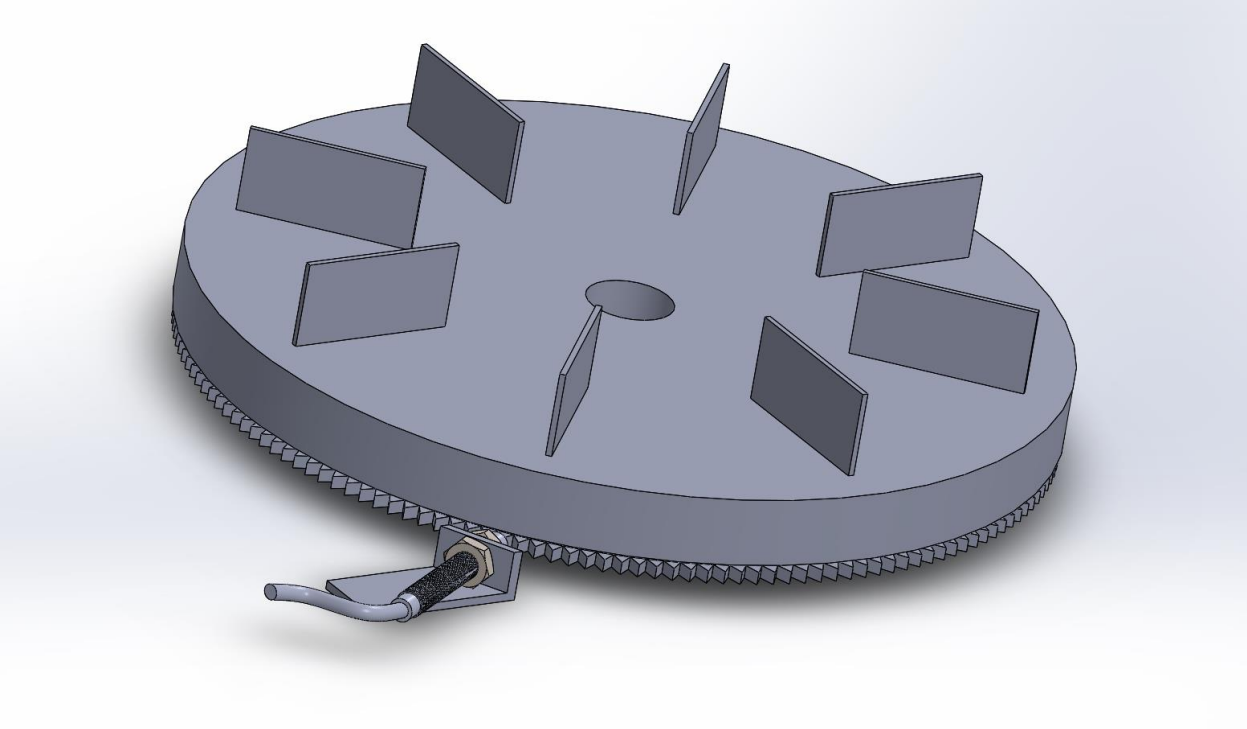


Figure 9 – Flywheel and RPM Sensor

Throttle Body

In this model we can see a generic throttle body sourced from an automotive application to provide quick prototyping. The throttle body operates by having a linear force applied to the circular actuator turning the shaft running through the middle which in turn opens the butterfly valve allowing more air to flow into the engine. The throttle body integrates a throttle position sensor (TPS), fuel injector, as well as a manifold absolute pressure (MAP) sensor. The TPS is used to measure the throttle inputs uses the principals of a potentiometer, feeding in a reference voltage and measuring the output voltage. The fuel injector is pulse width modulated meaning that it will have a constant pressure and volume of fuel flowing in the top and when it receives a pulse of a specific amount of time (width) it will allow the fuel to pass through to the inside of the throttle body, atomizing the fuel as it does so. The MAP sensor measures the pressure of the air entering the engine and is used to adjust fuel at altitude and calculate mass air flow.

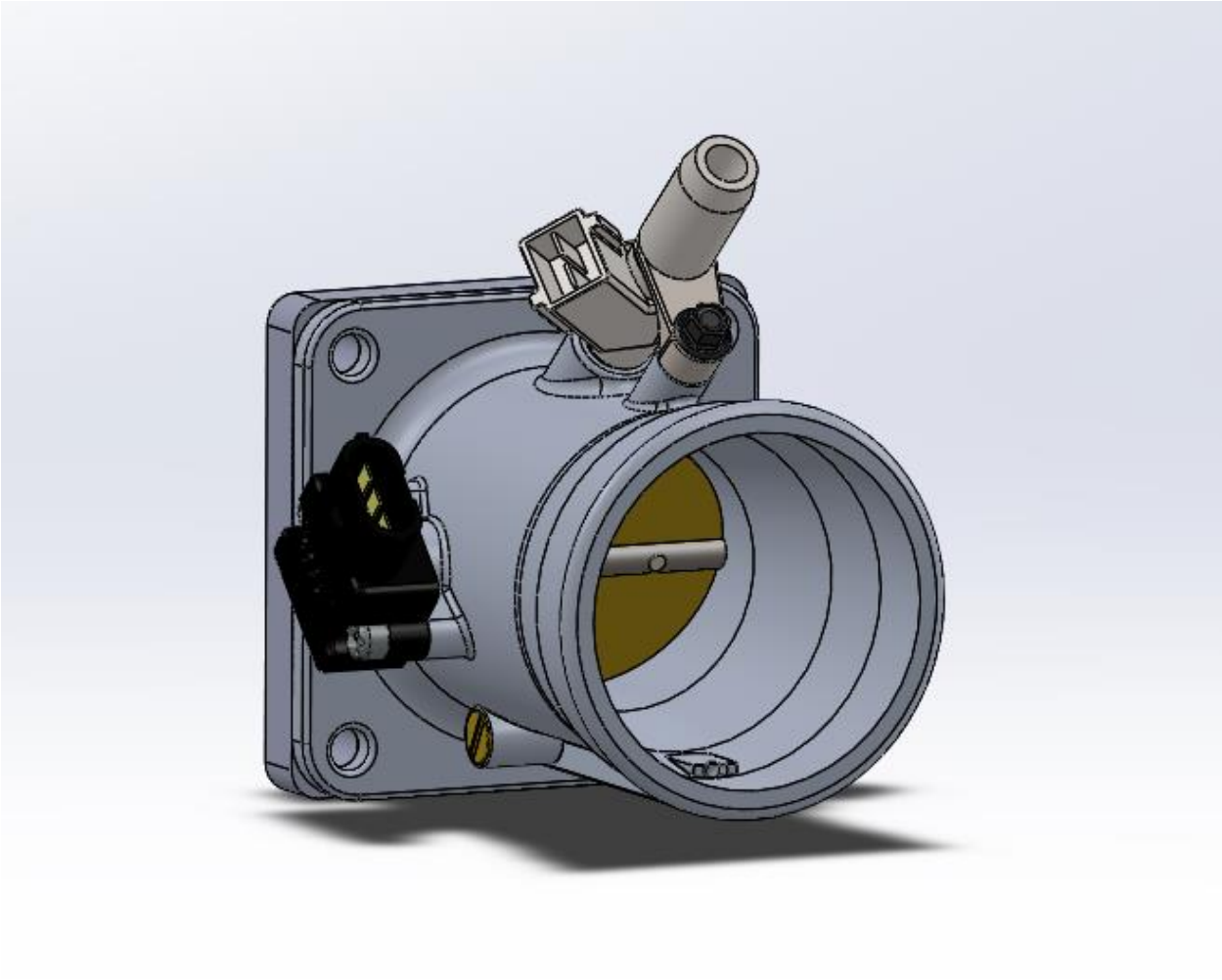


Figure 10 – Throttle Body

BILL OF MATERIAL

Category	Item	Description
Sensors	Manifold Absolute Pressure Sensor (MAP)	AC Delco Pressure Senesor
	Hall Effect Sensor (RPM Sensor)	NPN N/C Capacitive Proximity Sensor
	Temperature Sensor	Thermistor
	O2 Sensor	Narrowband O2 Sensor
Materials	Throttle Body (With TPS and Injector)	Throttle Body from Suzuki SV650
	Fuel Pressure Regulator	Return Style Fuel Pressure Regulator
	Wire	22AWG Solid Wire 6x25ft
	Shrink Tube	Misc Shrink Tubing For Wiring
	Fuel Pump	Turbine Style Fuel Pump
	Throttle Body Mount	SCH 40 PVC & Plexiglass
	O2 Sensor Mount	Clamp On Mount
	ECU Mount	Plexiglass Plate
Logic	Microcontroller	Arduino Due
	Breadboard	2x 400 Pin Breadboard
	MOSFET Driver	Board to activate MOSFET with logic signal
	Resistors	Resistor Pack

Table 4 – Bill of Material

BUILD AND TEST

DESIGN ALTERATIONS

During the build process there were design alterations that had to be made due to the complex nature of the project and the short timeframe. The first design alteration was that the hall effect RPM sensor was not able to read the gear teeth on the flywheel so instead it was set up to read the magneto on the flywheel. This provided a much less accurate tach signal as the magneto only reads once per revolution rather than the 96 of the flywheel teeth. The second design alteration was that due to time constraints the coil on plug ignition was abandoned as that was determined to have the least efficiency benefits as compared to the time required to implement. The third design alteration was, due to the low current and voltage the Arduino outputs, a MOSFET driver was required over the use of a normal MOSFET to trigger the injector. The fourth alteration was relocation of the temperature sensor to the cylinder block, and the use of a thermistor over a thermocouple to reduce the amount of wiring as a secondary thermistor IC would be needed. The fifth and final design alteration was the O2 sensor, the first change was to use a clamp on O2 sensor adaptor to avoid potential issues with welding on an older rusty exhaust and use of a narrowband O2 sensor over the more accurate wideband sensor. This was due to the fact that a narrowband signal can be read directly by the Arduino's analog inputs whereas the wideband would require the use of a proprietary amplifier. The final design updates are reflected in figure 11 and the wiring diagram in figure 12.

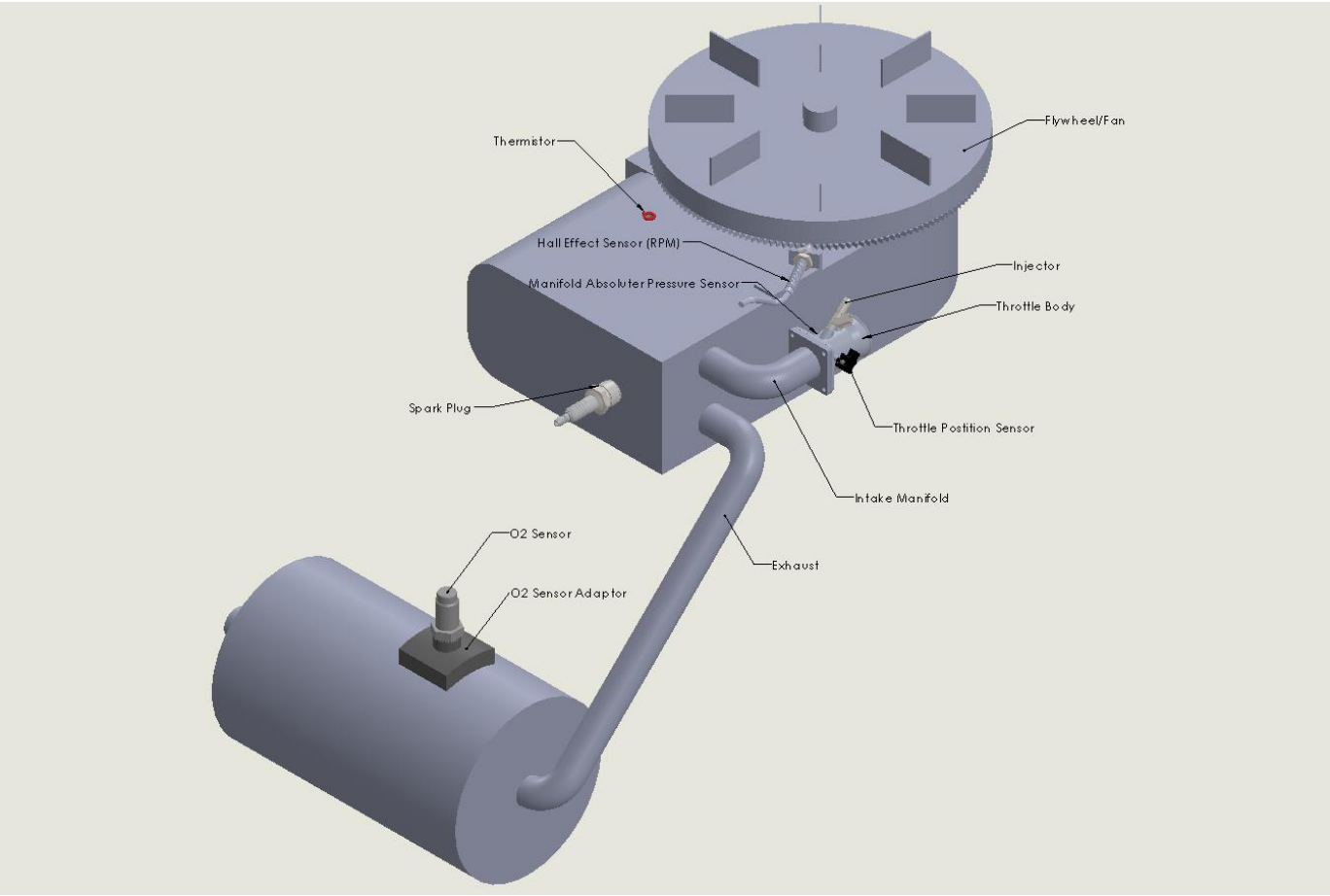


Figure 11 – Updated Concept Design

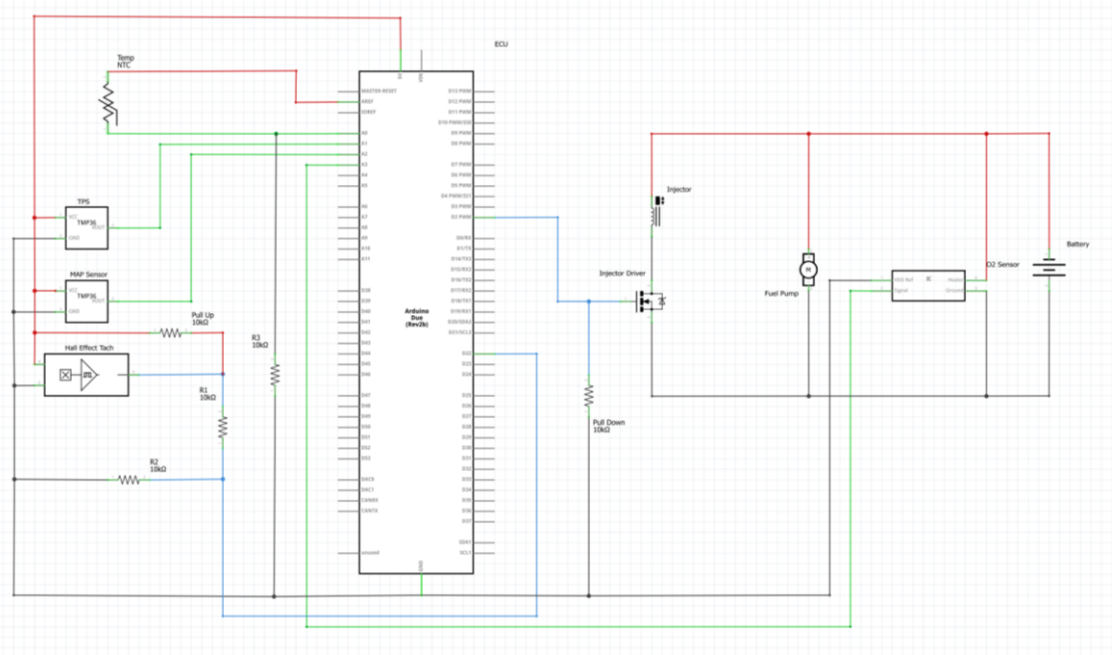


Figure 12 – Updated Wiring Diagram

DISCUSSION OF THE MANUFACTURING PROCESSES UTILIZED

This project's main area of focus was coding the engine control unit for electronic fuel injection. For that reason, to save time, almost all the components were taken from an already existing application and adapted to fit a lawn tractor. The throttle body seen in figure 13 was taken from a Suzuki motorcycle. The throttle position sensor was already integrated into the throttle body. The manifold absolute pressure sensor is an ACDelco unit taken from a General Motors SUV connected to the throttle body via vacuum hose as can be seen in figure 14.

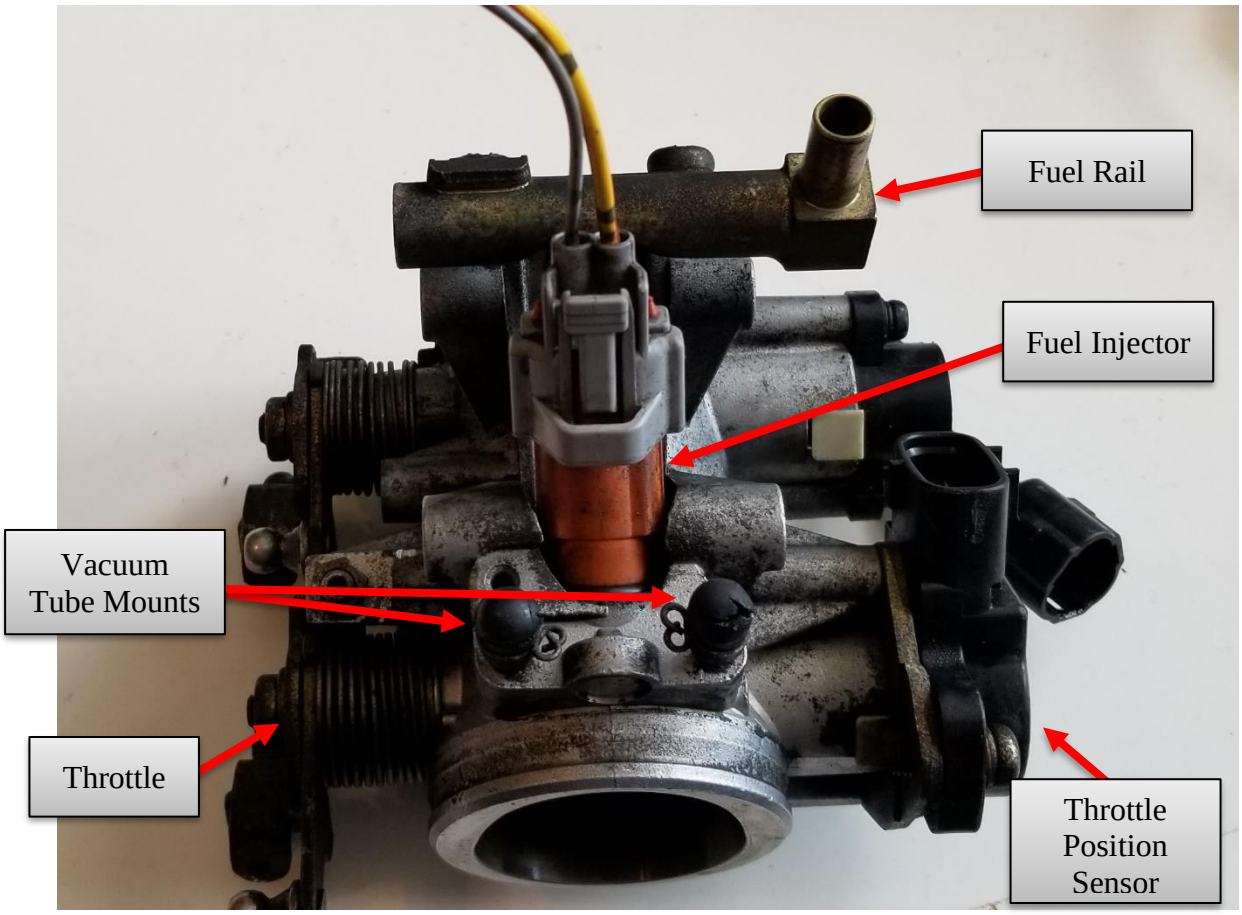


Figure 13 – Throttle Body

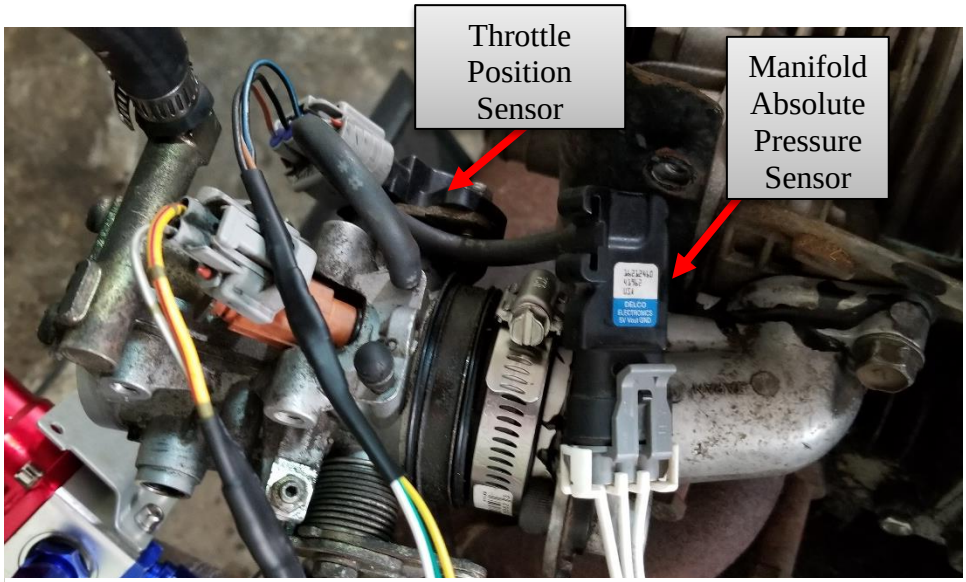


Figure 14 – Throttle Body Mounted

The O2 sensor was attached to the muffler via hose clamps and an aluminum adaptor (seen in figure 15) due to concern with the metal of the old rusty exhaust pipe not being weldable. The adaptor is a generic part with a seal underneath to prevent escaping exhaust gasses. This adaptor would provide the option for customers to install this system on their potentially rusty exhaust. This would also help reduce the time to install as there is very little fabrication work involved with this adaptor.

Figure 15 – O2 Sensor with Adaptor

The temperature sensor was attached to an existing screw found on the engine block seen in figure 16 to minimize fabrication needed. This or a similar screw can be assumed to be found on all small engines reducing the time to install for the customer.

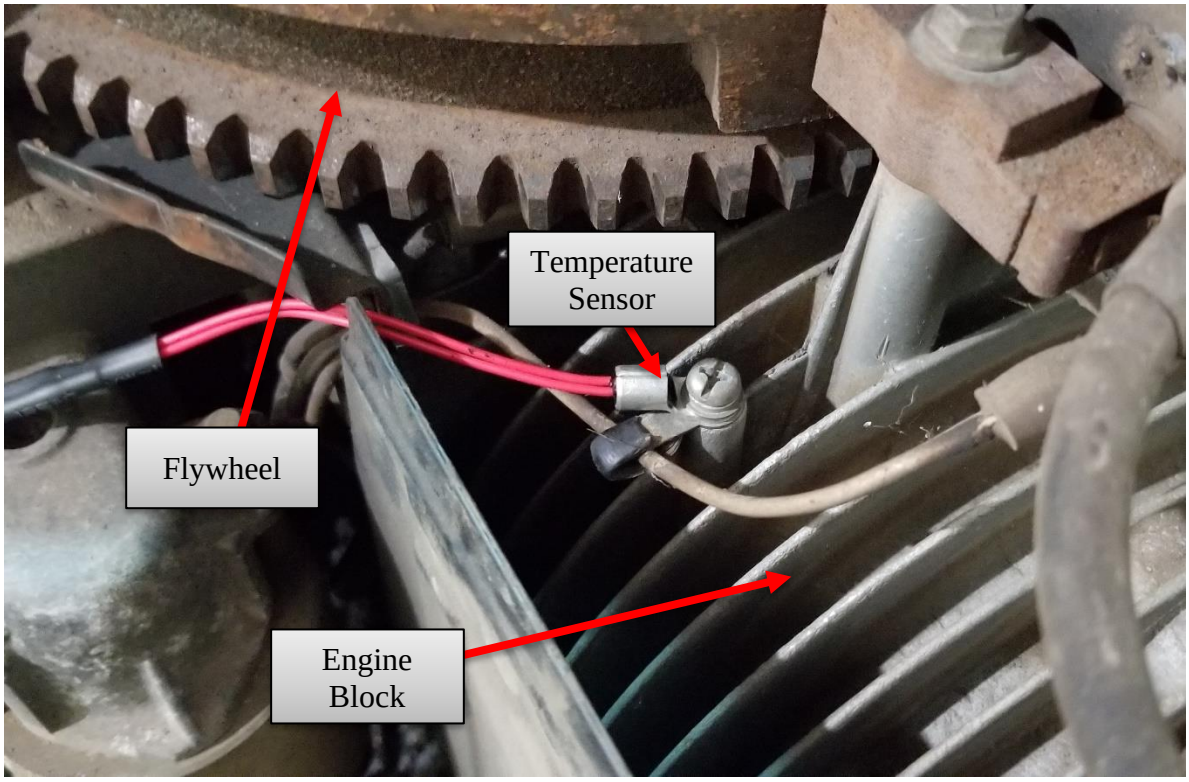


Figure 16 – Temperature Sensor Mounted

The hall effect tachometer was attached to the fan shroud of the engine. This provides a solid mounting location while minimizing install time for the customer as they wouldn't have to spend time fabricating a mounting bracket.

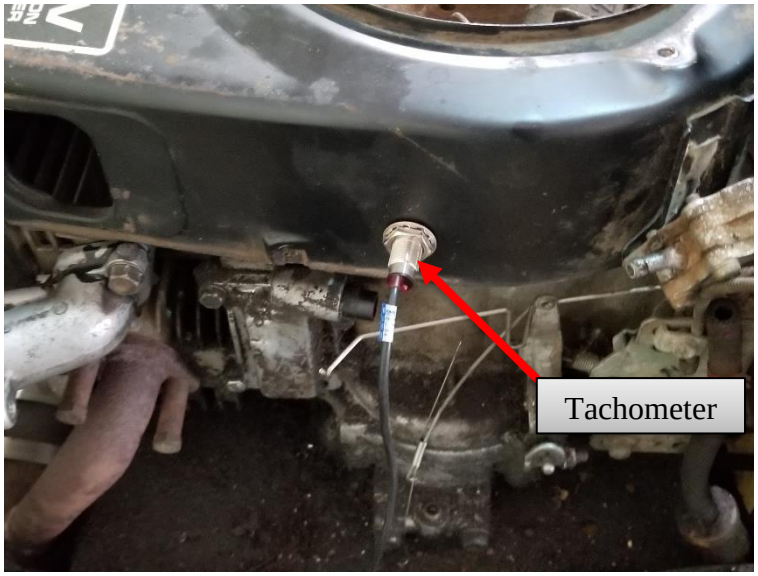


Figure 17 – Hall Effect Tachometer Mounted

The ECU was assembled using two 400 pin breadboards to separate logic inputs and outputs from 12V inputs and outputs. This was all mounted to a plexiglass panel to provide a simpler view and make wiring cleaner. If designed for production this would be replaced by a printed circuit board and a generic 15 pin connector to reduce time spent wiring.

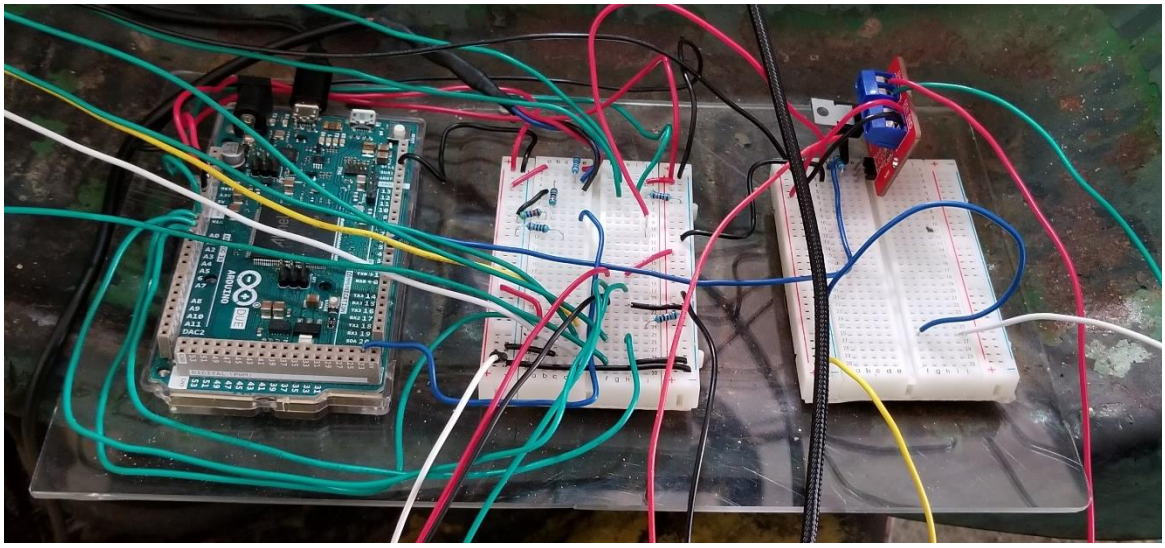


Figure 18 – Engine Control Unit

The throttle body was mounted to the intake runner (seen in figure 19) by a throttle body adaptor made with SCH40 PVC piping epoxied to a plexiglass mounting surface. This was for prototyping purposes only and if put to production would be replaced by an aluminum part or perhaps something 3D printed to maximize the number of engines that could be fit with the system as this is the only part that is specific to one engine.

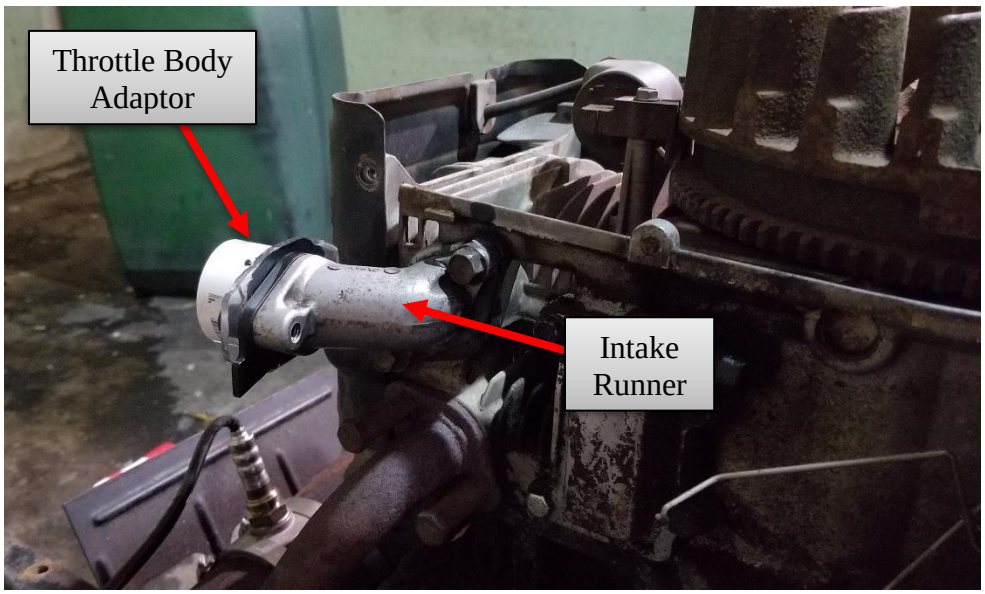


Figure 19 – Intake Runner and Adaptor

The fuel pump used was a turbine style pump from an automotive application, this was chosen as it could provide the 40 PSI needed for the injector to work properly. This was coupled to a return style pressure regulator necessary to prevent the pump from burning itself out during low fuel usage. The fuel pump would be located in the tank of the lawn tractor with the pressure regulator next to it with the return line of the pressure regulator attached back to the tank. Due to time constraints the pump and pressure regulator were placed in a temporary fuel tank.

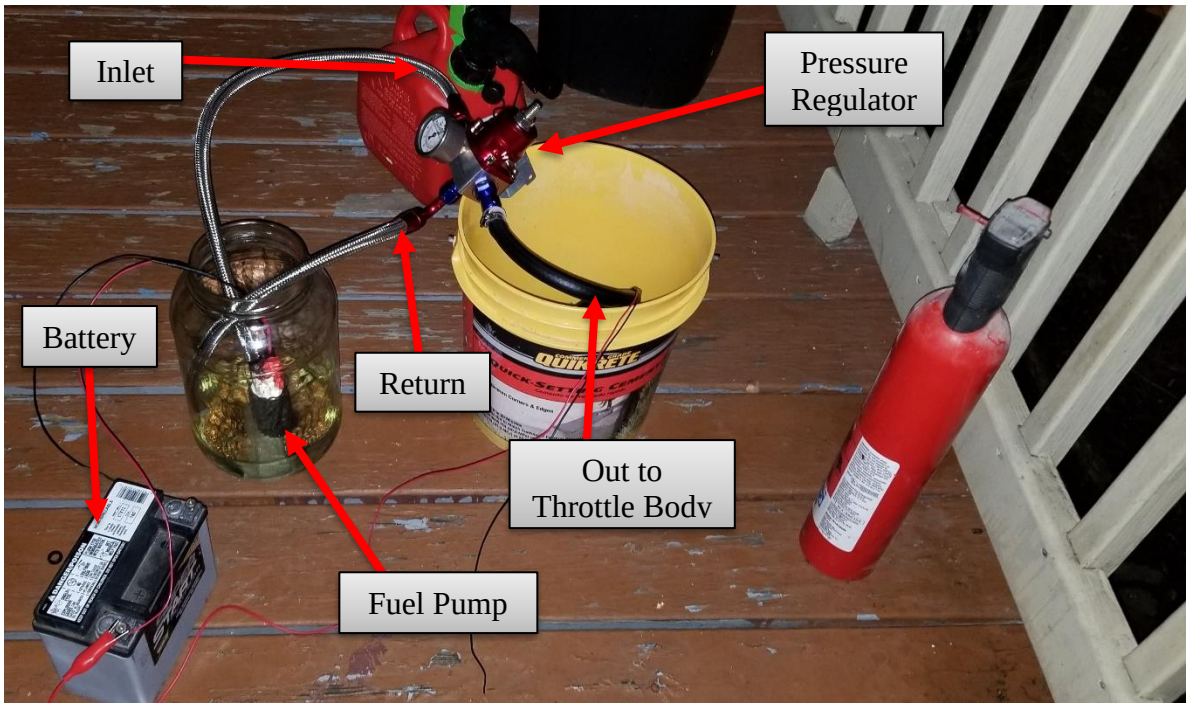


Figure 20 – Fuel Pump and Pressure Regulator

TEST PROCEDURE AND CRITERIA

The first testing procedure was to determine the fuel flow into the engine via an in-line fuel flow sensor. This would be used to measure fuel efficiency at idle, full throttle, and full throttle under load. Unfortunately, the commercially available flow sensors (example in figure 21) were not able to accurately detect the fuel flow as it was too low. To find the fuel flow, use of other techniques such as a timed fuel rundown or use of a rotameter would be needed.

Figure 21 – In-Line Fuel Flow Sensor

To test the power increase by use of the EFI system, RPM drop when engine is under load was used. This is designed only to compare relative power change and cannot be used to accurately give an absolute power figure. This test does however relate the benefits of increased performance to real world benefits as, in the case of the mower, being able to turn the mower deck faster would make for a more even and clean cut.

Install time would take place from the time that the hood is opened to the time that the engine first starts. This would give an accurate depiction of cleaning and/or adjusting a carburetor versus just installing this EFI system. Often carburetors have to be taken off to clean or adjust, if my system takes less time to install than three or four carburetor adjustments, it may prove beneficial in the long run on time consumption.

Efficiency will be measured by the O2 sensor in the exhaust. It is assumed that the closer to a perfect stoichiometric combustion, the cleaner the exhaust emissions are. The narrowband O2 sensor doesn't provide as much information as the wideband but it is enough to understand if the EFI system is more or less efficient.

Time to start will be measured form the time that the key is turned to the time that the engine runs without intervention. This is an important measure of usability as often if it is a colder day or the engine hasn't been run for a while it can be difficult to start a carbureted engine.

TEST RESULTS AND FINDINGS

Through testing, the EFI system compared very favorably to the carburetor. All measures that could be recorded except for install time showed the EFI system as preferable as can be seen in table 5. The product objective of maximizing power was reached as the RPM drop under load for the EFI system was less. The product objective of hitting target O2 ratios was demonstrated to be better with the EFI system as seen in figure 22. The product objective of minimizing time to start was thoroughly reached as the EFI system started in under a quarter of the time the carburetor took and didn't require any intervention. The product objective of install time unfortunately shows the carburetor to be better however if the carburetor has to be taken off multiple times for adjustment or cleaning this could also be a win for the EFI system as most all adjustments of the EFI system don't require removal of the component. Though fuel usage was not able to be determined, it can be assumed that with a more efficient combustion it would decrease.

The EFI system as can be seen through the testing process, reached the initial goal of the project which was to increase the power and efficiency of older small engines. This would provide a very compelling package to any potential customer as they may see a complete benefit from installing this EFI system with little to no drawbacks. With the increase in efficiency the customer may also see a return on investment in the immediate life of the tractor.

Measure	Carb	EFI
O2 Ratio	Very Rich	Stoichiometric
Fuel Flow (L/M)	0.14 (Inaccurate)	Unknown
RPM Drop (High:Load) Diff	4006:3529 – 477	4118:3742 – 376
Install Time	23 Min	Unknown (Estimated 1.5 hours)
Time to Start Cold (S)	20.52 | Engine stalled 4 times | Had to quickly adjust throttle	4.24 | Started immediately | Engine didn't stall | No throttle adjustments needed

Table 5 – Test Results

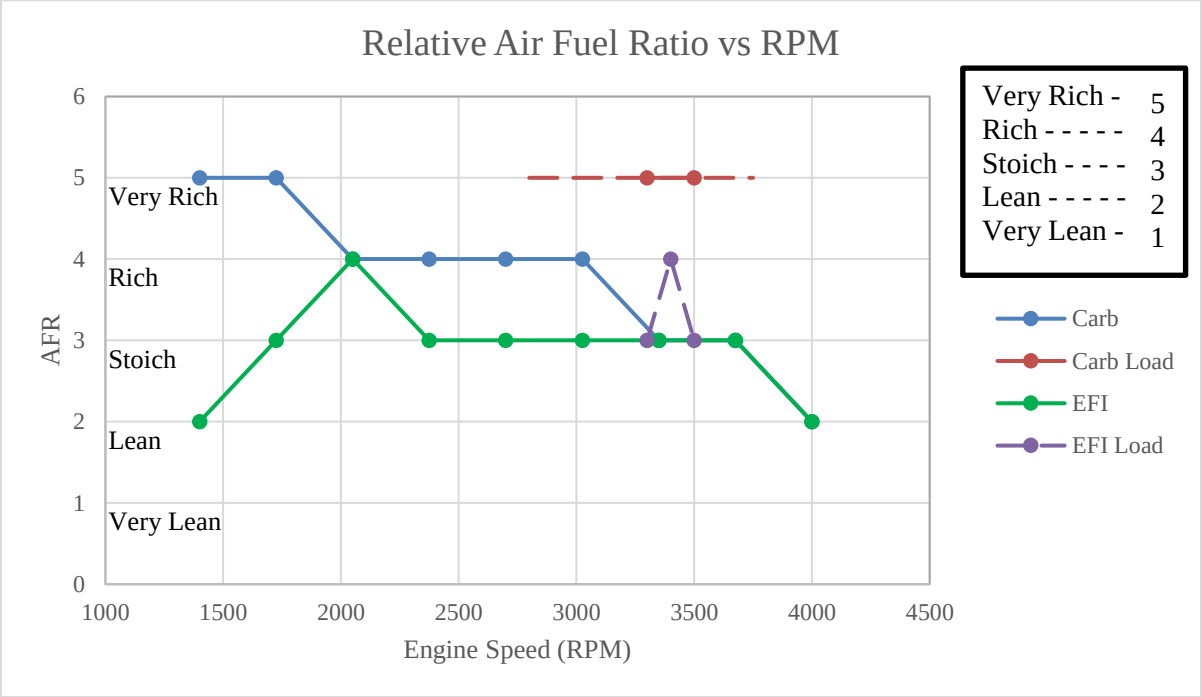


Figure 22 – Relative Air Fuel Ratio Graph

PROJECT MANAGEMENT

BUDGET, PROPOSED/ACTUAL

The budget for this project was a target discovered from the surveys, which said pretty emphatically that above $500 to $1000 dollars this EFI system would not be marketable. Given the limit of $500 the initial budget (table 6) was well withing that, providing enough space left to spend double and still have a marketable project. The actual budget was around 30 percent higher than the proposed due to design changes necessitating different parts used and taken from different sources. This is still far below the $500 budget limit and makes the system very marketable with a 20 percent profit margin at $500. If this project were to be brought to market, use of a printed circuit board as well as bulk pricing of components would help decrease the cost of the product even more.

Category	Item	Description	Source	Price ($)
Sensors	Throttle Position Sensor (TPS)	Potentiometer	Micro Center	2.99
	Manifold Absolute Pressure Sensor (MAP)	Barometric Pressure Sensor	Amazon	5.99
	Fuel Injector	PWM Fuel Injector	Junk Yard	7.99
	Hall Effect Sensor (RPM Sensor)	NPN N/C Capacitive Proximity Sensor	Amazon	9.99
	Temperature Sensor	Thrmocouple	Amazon	14.99
	Coil on Plug Ignition (COP)	Ignition Module	Junk Yard	11.99
	O2 Sensor	Wideband O2 Sensor	Junk Yard	8.99
	Fuel Flow Sensor	Fluid Flow Sensor	Amazon	29.99
Materials	Fuel Pump	PWM Fuel Pump	Junk Yard	21.99
	Throttle Body	Air Metering Device	Junk Yard	31.99
	ECU Enclosure	3D Printed Plastic or Steel	1819/VPC	TBD
Logic	Microcontroller	Arduino Due	Arduino	40
	Breadboard	Breakout Breadboard Kit	Amazon	11.99
	FQP30N06L	MOSFETx2	Amazon	8.99
	Resistor	220 Ohm	Amazon	4.99
	Flyback Diode	Diode	Amazon	4.99
			Sub-Total	$ 217.86
			Total	$ 233.11

Table 6 – Proposed Budget

Category	Item	Description	Source	Price ($)
Sensors	Manifold Absolute Pressure Sensor (MAP)	AC Delco Pressure Senesor	Junk Yard	10.00
	Hall Effect Sensor (RPM Sensor)	NPN N/C Capacitive Proximity Sensor	Amazon	9.99
	Temperature Sensor	Thermistor	Digi-Key	7.27
	O2 Sensor	Narrowband O2 Sensor	O'Rileys	26.38
Materials	Throttle Body (With TPS and Injector)	Throttle Body from Suzuki SV650	Junk Yard	42.00
	Fuel Pressure Regulator	Return Style Fuel Pressure Regulator	Amazon	35.99
	Wire	22AWG Solid Wire 6x25ft	Micro Center	16.25
	Shrink Tube	Misc Shrink Tubing For Wiring	Amazon	10.00
	Fuel Pump	Turbine Style Fuel Pump	Amazon	16.99
	Throttle Body Mount	SCH 40 PVC & Plexiglass	Ace	10.00
	O2 Sensor Mount	Strap On Mount	Ace/Amazon	30.00
	ECU Mount	Plexiglass Plate	Ace	2.50
Logic	Microcontroller	Arduino Due	Arduino	42.00
	Breadboard	2x 400 Pin Breadboard	Micro Center	11.99
	MOSFET Driver	Board to activate MOSFET with logic signal	Amazon	8.99
	Resistors	Resistor Pack	Micro Center	7.99
			Sub-Total	$ 288.34
			Total	$ 308.52

Table 7 – Actual Budget

SCHEDULE, PROPOSED /ACTUAL

In regard to schedule, this project ended up taking much more time than expected. All the small items such as adapting the throttle body, assembling sensors, or testing went mostly according to schedule. However, the time for programming and troubleshooting went far above what had been assumed, more than doubling both. This was due to a lack of experience in C++ and the difficult nature of this project. The initial schedule allowed five weeks of flexibility to complete the project before the tech expo as it stated the project would only take eight weeks, however the actual schedule required working right up to the due date of the project.

Task	Duration	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Adapting Throttle Body	15 Hours									
Assembling Sensor Array	25 Hours									
Carburetor Testing	5 Hours									
Fuel Pump	8 Hours									
Wiring	20 Hours									
Programming	40 Hours									
Troubleshooting	50 Hours									

Table 8 – Proposed Schedule

Task	Duration	Week													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Adapting Throttle Body	15 Hours														
Assembling Sensor Array	25 Hours														
Carburetor Testing	5 Hours														
Fuel Pump	2 Hours														
Wiring	20 Hours														
Programming	95 Hours														
Troubleshooting	135 Hours														
Total	297 Hours														

Table 9 – Actual Schedule

SUSTAINABILITY AND MATERIAL USAGE

The material used to construct the throttle body adaptor was chosen due to its inexpensive nature and ability to be machined with relatively simple tools. Due to the nature of working in the victory parkway campus during the pandemic I decided to do as much of the work with my own tools as possible. If this product were to be sold, the throttle body mount would need to be made out of metal, presumably aluminum, or with a thermoplastic that is designed to handle high temperatures.

CONCLUSIONS

The EFI system designed in this project met and even exceeded the expectations of the initial project statement of increasing efficiency and power in older small engines through use of an EFI system. This system would be able to reduce emissions, increase power, increase efficiency, increase ease of use, and potentially ease of maintenance. While some alterations had to be made in the design of the project due to time constraints, the system still performed admirably with potential to grow as more time is invested into it. This project would be a very compelling package to any potential customer as they may see very little if any drawbacks from installing this EFI system. The project came in a little over the proposed budget but well under the limit providing the potential for a very marketable price point. The project did however take much longer than expected due to issues coding and troubleshooting.

If this product were to be changed, the first thing I would recommend is that a volumetric efficiency table be used as the initial fueling values rather than the throttle position table that is currently used. Another thing that could be improved would be the use of a wideband O2 sensor over the narrowband currently installed. The final thing to be

improved would be the use of a trigger wheel specifically designed for this application. This would allow for a more accurate RPM resolution as well as enable the usage of computer-controlled ignition and more fuel-efficient sequential fuel injection over the batch injection cycle currently used.

WORKS CITED

1. **International Agency for Research on Cancer.** *Diesel and Gasoline Engine Exhausts and Some Nitroarenes.* Lyon, France : International Agency for Research on Cancer, 1989.
2. **Miller.** Electronic Fuel Injection Technology Improves Efficiency of Engine-Driven Welding Power Sources. *MillerWelds.* [Online] February 13, 2020. [Cited: August 12, 2020.] https://www.millerwelds.com/resources/article-library/electronic-fuel-injection-technology-improves-efficiency-of-engine-driven-welding-power-sources.
3. *The effect of air/fuel ratio on the CO and NOx emissions for a twin-spark motorcycle gasoline engine under wide range of operating conditions.* **Deng, Banglin, et al.** Shenzen, China : ScienceDirect, 2019, Vol. 169.
4. **NHS.** Carbon Monoxide Poisoning. *NHS.* [Online] April 10, 2019. [Cited: August 11, 2020.] https://www.nhs.uk/conditions/carbon-monoxide-poisoning/.
5. **CDC.** Carbon Monoxide Hazards from Small Gasoline Powered Engines. *CDC.* [Online] June 5, 2012. [Cited: August 11, 2020.] https://www.cdc.gov/niosh/topics/co/default.html.
6. **EPA.** Regulations for Emissions from Small Equipment & Tools. *EPA.* [Online] [Cited: August 12, 2020.] https://www.epa.gov/regulations-emissions-vehicles-and-engines/regulations-emissions-small-equipment-tools.
7. **Holley.** Terminator X Universal MPFI Kit. *Holley.* [Online] [Cited: August 12, 2020.] https://www.holley.com/products/fuel_systems/fuel_injection/terminator_x/terminator_x_and_x_max_universal_kits/terminator_x_universal_kits/parts/550-936.
8. **Walbro.** Walbro EEM™: An All-Around Ideal Fuel Injection Solution . *Walbro.* [Online] [Cited: 9 16, 2020.] https://www.walbro.com/walbro-eem-around-ideal-fuel-injection-solution/.
9. **NanoEFI.** AFFORDABLE FUEL INJECTION FOR SMALL ENGINES. *NanoEFI Developmet.* [Online] [Cited: 9 16, 2020.] https://www.nanoefi.com/.

APPENDIX A: SOURCE CODE

Arduino Source Code

```
// Scheduler - Version: Latest
#include <Scheduler.h>

/*
  This is the final EFI file. This includes injector, tachometer, O2 sensor
correction,
  manifold absolute pressure (MAP) sensor correction, and a serial data print.

  Code Created by Ian Lind

*/



// Global Variables
// Digital Input
#define hallPin 22            // This is the pin for the RPM sensor

// Analog Input
#define tempPin 0             // This is the analog pin for the temperature sensor
#define tpsPin 1              // This is the analog pin for the throttle position
sensor (TPS)
#define mapPin 2              // This is the analog pin for the manifold absolute
pressure (MAP) sensor
#define O2Pin 3               // This is the input pin for the O2 sensor

//Digital Output
#define pulsePin 2            // This is the injector pin

#define thresh 2              // Set number of hall trips for RPM reading (higher
improves accuracy)
double rpm_val;               // Calculated RPM value
int INJTrig = 0;              // Trigger for injector
double temp;                  // Temperature reading variable

int lean_c = 0;               // Variable for rich count
int rich_c = 0;               // Variable for lean count
double O2;                    // O2 Value
double O2_cf;                 // O2 Correction factor

int inj = 0;

double MAP_cf;                // MAP Correction factor
double MAP_p = 1050;          // MAP Reading (provided with inital value to not
cause calculation errors)

float TPS_L = 27.5;           // TPS calibration for closed
float TPS_H = 222.6;          // TPS calibration for open
int RPM;                      // RPM value for testing
double TPS;                   // TPS value for testing
```

```
double pulse;                  // Creates pulse variable

// Injector Timing Tables
double pulseWidth[59][21] = {
  {1.775, 1.835, 1.895, 1.96, 2.02, 2.08, 2.14, 2.205, 2.265, 2.325, 2.39, 2.41,
2.51, 2.57, 2.63, 2.69, 2.75, 2.815, 2.875, 2.94, 3},
  {1.78, 1.84, 1.9, 1.965, 2.025, 2.085, 2.145, 2.21, 2.27, 2.33, 2.395, 2.415,
2.515, 2.575, 2.635, 2.695, 2.755, 2.82, 2.88, 2.945, 3.005},
  {1.785, 1.845, 1.905, 1.97, 2.03, 2.09, 2.15, 2.215, 2.275, 2.335, 2.4, 2.42,
2.52, 2.58, 2.64, 2.7, 2.76, 2.825, 2.885, 2.95, 3.01},
  {1.79, 1.85, 1.91, 1.975, 2.035, 2.095, 2.155, 2.22, 2.28, 2.34, 2.405, 2.425,
2.525, 2.585, 2.645, 2.705, 2.765, 2.83, 2.89, 2.955, 3.015},
  {1.795, 1.855, 1.915, 1.98, 2.04, 2.1, 2.16, 2.225, 2.285, 2.345, 2.41, 2.43,
2.53, 2.59, 2.65, 2.71, 2.77, 2.835, 2.895, 2.96, 3.02},
  {1.8, 1.86, 1.92, 1.985, 2.045, 2.105, 2.165, 2.23, 2.29, 2.35, 2.415, 2.435,
2.535, 2.595, 2.655, 2.715, 2.775, 2.84, 2.9, 2.965, 3.025},
  {1.805, 1.865, 1.925, 1.99, 2.05, 2.11, 2.17, 2.235, 2.295, 2.355, 2.42, 2.44,
2.54, 2.6, 2.66, 2.72, 2.78, 2.845, 2.905, 2.97, 3.03},
  {1.81, 1.87, 1.93, 1.995, 2.055, 2.115, 2.175, 2.24, 2.3, 2.36, 2.425, 2.445,
2.545, 2.605, 2.665, 2.725, 2.785, 2.85, 2.91, 2.975, 3.035},
  {1.815, 1.875, 1.935, 2, 2.06, 2.12, 2.18, 2.245, 2.305, 2.365, 2.43, 2.45, 2.55,
2.61, 2.67, 2.73, 2.79, 2.855, 2.915, 2.98, 3.04},
  {1.82, 1.88, 1.94, 2.005, 2.065, 2.125, 2.185, 2.25, 2.31, 2.37, 2.435, 2.455,
2.555, 2.615, 2.675, 2.735, 2.795, 2.86, 2.92, 2.985, 3.045},
  {1.825, 1.886, 1.945, 2.01, 2.07, 2.13, 2.19, 2.255, 2.315, 2.375, 2.44, 2.46,
2.56, 2.62, 2.68, 2.74, 2.8, 2.865, 2.925, 2.99, 3.05},
  {1.83, 1.89, 1.952, 2.015, 2.075, 2.135, 2.195, 2.26, 2.32, 2.38, 2.445, 2.465,
2.565, 2.625, 2.685, 2.745, 2.805, 2.87, 2.93, 2.995, 3.055},
  {1.835, 1.895, 1.9573, 2.02, 2.08, 2.14, 2.2, 2.265, 2.325, 2.385, 2.45, 2.47,
2.57, 2.63, 2.69, 2.75, 2.81, 2.875, 2.935, 3, 3.06},
  {1.84, 1.9, 1.962, 2.025, 2.085, 2.145, 2.205, 2.27, 2.33, 2.39, 2.455, 2.475,
2.575, 2.635, 2.695, 2.755, 2.815, 2.88, 2.94, 3.005, 3.065},
  {1.845, 1.905, 1.967, 2.028, 2.09, 2.15, 2.21, 2.275, 2.335, 2.395, 2.46, 2.48,
2.58, 2.64, 2.7, 2.76, 2.82, 2.885, 2.945, 3.01, 3.07},
  {1.85, 1.91, 1.97, 2.033, 2.095, 2.155, 2.215, 2.28, 2.34, 2.4, 2.465, 2.485,
2.585, 2.645, 2.705, 2.765, 2.825, 2.89, 2.95, 3.015, 3.075},
  {1.855, 1.915, 1.975, 2.038, 2.1, 2.16, 2.22, 2.285, 2.345, 2.405, 2.47, 2.49,
2.59, 2.65, 2.71, 2.77, 2.83, 2.895, 2.955, 3.02, 3.08},
  {1.86, 1.92, 1.98, 2.045, 2.105, 2.165, 2.225, 2.29, 2.35, 2.41, 2.475, 2.495,
2.595, 2.655, 2.715, 2.775, 2.835, 2.9, 2.96, 3.025, 3.085},
  {1.865, 1.925, 1.985, 2.05, 2.11, 2.171, 2.23, 2.295, 2.355, 2.415, 2.48, 2.5,
2.6, 2.66, 2.72, 2.78, 2.84, 2.905, 2.965, 3.03, 3.09},
  {1.87, 1.93, 1.99, 2.055, 2.115, 2.176, 2.237, 2.3, 2.36, 2.42, 2.485, 2.505,
2.605, 2.665, 2.725, 2.785, 2.845, 2.91, 2.97, 3.035, 3.095},
  {1.875, 1.935, 1.995, 2.06, 2.12, 2.18, 2.242, 2.305, 2.365, 2.425, 2.49, 2.51,
2.61, 2.67, 2.73, 2.79, 2.85, 2.915, 2.975, 3.04, 3.1},
  {1.88, 1.94, 2, 2.065, 2.125, 2.185, 2.247, 2.31, 2.37, 2.43, 2.495, 2.515, 2.615,
2.675, 2.735, 2.795, 2.855, 2.92, 2.98, 3.045, 3.105},
  {1.885, 1.945, 2.005, 2.07, 2.13, 2.19, 2.252, 2.313, 2.375, 2.435, 2.5, 2.52,
2.62, 2.68, 2.74, 2.8, 2.86, 2.925, 2.985, 3.05, 3.11},
  {1.89, 1.95, 2.01, 2.075, 2.135, 2.195, 2.257, 2.318, 2.379, 2.44, 2.505, 2.525,
2.625, 2.685, 2.745, 2.805, 2.865, 2.93, 2.99, 3.055, 3.115},
  {1.895, 1.955, 2.015, 2.08, 2.14, 2.2, 2.26, 2.323, 2.384, 2.445, 2.51, 2.53,
2.63, 2.69, 2.75, 2.81, 2.87, 2.935, 2.995, 3.06, 3.12},
  {1.9, 1.96, 2.02, 2.085, 2.145, 2.205, 2.265, 2.33, 2.389, 2.45, 2.515, 2.535,
2.635, 2.695, 2.755, 2.815, 2.875, 2.94, 3, 3.065, 3.125},
```

```
  {1.905, 1.965, 2.025, 2.09, 2.15, 2.21, 2.27, 2.335, 2.395, 2.455, 2.517, 2.54,
2.64, 2.7, 2.76, 2.82, 2.88, 2.945, 3.005, 3.07, 3.13},
  {1.91, 1.97, 2.03, 2.095, 2.155, 2.215, 2.275, 2.34, 2.4, 2.46, 2.522, 2.545,
2.645, 2.705, 2.765, 2.825, 2.885, 2.95, 3.01, 3.075, 3.135},
  {1.915, 1.975, 2.035, 2.1, 2.16, 2.22, 2.28, 2.345, 2.405, 2.465, 2.527, 2.55,
2.65, 2.71, 2.77, 2.83, 2.89, 2.955, 3.015, 3.08, 3.14},
  {1.92, 1.98, 2.04, 2.105, 2.165, 2.225, 2.285, 2.35, 2.41, 2.47, 2.532, 2.555,
2.655, 2.715, 2.775, 2.835, 2.895, 2.96, 3.02, 3.085, 3.145},
  {1.925, 1.985, 2.045, 2.11, 2.17, 2.23, 2.29, 2.355, 2.415, 2.475, 2.537, 2.56,
2.659, 2.72, 2.78, 2.84, 2.9, 2.965, 3.025, 3.09, 3.15},
  {1.93, 1.99, 2.05, 2.115, 2.175, 2.235, 2.295, 2.36, 2.42, 2.48, 2.54, 2.565,
2.664, 2.725, 2.785, 2.845, 2.905, 2.97, 3.03, 3.095, 3.155},
  {1.935, 1.995, 2.055, 2.12, 2.18, 2.24, 2.3, 2.365, 2.425, 2.485, 2.545, 2.57,
2.669, 2.73, 2.79, 2.85, 2.91, 2.975, 3.035, 3.1, 3.16},
  {1.94, 2, 2.06, 2.125, 2.185, 2.245, 2.305, 2.37, 2.43, 2.49, 2.55, 2.575, 2.674,
2.735, 2.795, 2.855, 2.915, 2.98, 3.04, 3.105, 3.165},
  {1.945, 2.005, 2.065, 2.13, 2.19, 2.25, 2.31, 2.375, 2.435, 2.495, 2.555, 2.58,
2.68, 2.74, 2.8, 2.86, 2.92, 2.985, 3.045, 3.11, 3.17},
  {1.95, 2.01, 2.07, 2.135, 2.195, 2.255, 2.315, 2.38, 2.44, 2.5, 2.56, 2.585,
2.685, 2.745, 2.806, 2.865, 2.925, 2.99, 3.05, 3.115, 3.175},
  {1.955, 2.015, 2.075, 2.14, 2.2, 2.26, 2.32, 2.385, 2.445, 2.505, 2.565, 2.59,
2.69, 2.75, 2.811, 2.872, 2.93, 2.995, 3.055, 3.12, 3.18},
  {1.96, 2.02, 2.08, 2.145, 2.205, 2.265, 2.325, 2.39, 2.45, 2.51, 2.57, 2.595,
2.695, 2.755, 2.816, 2.877, 2.938, 3, 3.06, 3.125, 3.185},
  {1.965, 2.025, 2.085, 2.15, 2.21, 2.27, 2.33, 2.395, 2.455, 2.515, 2.575, 2.6,
2.7, 2.76, 2.82, 2.882, 2.943, 3.005, 3.065, 3.13, 3.19},
  {1.97, 2.03, 2.09, 2.155, 2.215, 2.275, 2.335, 2.4, 2.46, 2.52, 2.58, 2.605,
2.705, 2.765, 2.825, 2.887, 2.948, 3.01, 3.07, 3.135, 3.195},
  {1.975, 2.035, 2.095, 2.16, 2.22, 2.28, 2.34, 2.405, 2.465, 2.525, 2.585, 2.61,
2.71, 2.77, 2.83, 2.892, 2.953, 3.015, 3.075, 3.14, 3.2},
  {1.98, 2.04, 2.1, 2.165, 2.225, 2.285, 2.345, 2.41, 2.47, 2.53, 2.59, 2.615,
2.715, 2.775, 2.835, 2.895, 2.958, 3.02, 3.08, 3.145, 3.205},
  {1.985, 2.045, 2.105, 2.17, 2.23, 2.29, 2.35, 2.415, 2.475, 2.535, 2.595, 2.62,
2.72, 2.78, 2.84, 2.9, 2.953, 3.025, 3.085, 3.15, 3.21},
  {1.99, 2.05, 2.11, 2.175, 2.235, 2.295, 2.355, 2.42, 2.48, 2.54, 2.6, 2.625,
2.725, 2.785, 2.845, 2.905, 2.96, 3.03, 3.09, 3.157, 3.223},
  {1.995, 2.055, 2.115, 2.18, 2.24, 2.3, 2.36, 2.425, 2.485, 2.545, 2.605, 2.63,
2.73, 2.79, 2.85, 2.91, 2.965, 3.035, 3.096, 3.162, 3.228},
  {2, 2.06, 2.12, 2.185, 2.245, 2.305, 2.365, 2.43, 2.49, 2.55, 2.61, 2.635, 2.735,
2.795, 2.855, 2.915, 2.97, 3.04, 3.101, 3.167, 3.233},
  {2.005, 2.065, 2.125, 2.19, 2.25, 2.31, 2.37, 2.435, 2.495, 2.555, 2.615, 2.64,
2.74, 2.8, 2.86, 2.92, 2.975, 3.045, 3.106, 3.172, 3.238},
  {2.01, 2.07, 2.13, 2.195, 2.255, 2.315, 2.375, 2.44, 2.5, 2.56, 2.62, 2.645,
2.745, 2.805, 2.865, 2.925, 2.98, 3.05, 3.11, 3.177, 3.243},
  {2.015, 2.075, 2.135, 2.2, 2.26, 2.32, 2.38, 2.445, 2.505, 2.565, 2.625, 2.65,
2.75, 2.81, 2.87, 2.93, 2.985, 3.055, 3.115, 3.182, 3.248},
  {2.02, 2.08, 2.14, 2.205, 2.265, 2.325, 2.385, 2.45, 2.51, 2.57, 2.63, 2.655,
2.755, 2.815, 2.875, 2.935, 2.99, 3.06, 3.12, 3.187, 3.253},
  {2.025, 2.085, 2.145, 2.21, 2.27, 2.33, 2.39, 2.455, 2.515, 2.575, 2.635, 2.66,
2.76, 2.82, 2.88, 2.94, 2.995, 3.065, 3.125, 3.192, 3.258},
  {2.03, 2.09, 2.15, 2.215, 2.275, 2.335, 2.395, 2.46, 2.52, 2.58, 2.64, 2.665,
2.765, 2.825, 2.885, 2.945, 3, 3.07, 3.13, 3.197, 3.263},
  {2.035, 2.095, 2.155, 2.22, 2.28, 2.34, 2.4, 2.465, 2.525, 2.585, 2.645, 2.67,
2.77, 2.83, 2.89, 2.95, 3.005, 3.075, 3.135, 3.202, 3.268},
  {2.04, 2.1, 2.16, 2.225, 2.285, 2.345, 2.405, 2.47, 2.53, 2.59, 2.65, 2.675,
2.775, 2.835, 2.895, 2.955, 3.01, 3.08, 3.14, 3.207, 3.273},
```

```
  {2.045, 2.105, 2.165, 2.23, 2.29, 2.35, 2.41, 2.475, 2.535, 2.595, 2.655, 2.68,
2.78, 2.84, 2.9, 2.96, 3.015, 3.085, 3.145, 3.212, 3.278},
  {2.05, 2.11, 2.17, 2.235, 2.295, 2.355, 2.415, 2.48, 2.54, 2.6, 2.66, 2.685,
2.785, 2.845, 2.905, 2.965, 3.02, 3.09, 3.15, 3.217, 3.283},
  {2.055, 2.115, 2.175, 2.24, 2.3, 2.36, 2.42, 2.485, 2.545, 2.605, 2.665, 2.69,
2.79, 2.85, 2.91, 2.97, 3.025, 3.095, 3.155, 3.222, 3.288},
  {2.06, 2.12, 2.18, 2.245, 2.305, 2.365, 2.425, 2.49, 2.55, 2.61, 2.67, 2.695,
2.795, 2.855, 2.915, 2.975, 3.03, 3.1, 3.16, 3.227, 3.293},
  {2.065, 2.125, 2.185, 2.25, 2.31, 2.37, 2.43, 2.495, 2.555, 2.615, 2.675, 2.7,
2.8, 2.86, 2.92, 2.98, 3.035, 3.105, 3.165, 3.232, 3.298},
};                // This is the setup for the main injection timing table
double pulseStart[] = {1.510, 1.535, 1.560, 1.585, 1.610, 1.635, 1.660, 1.685,
1.710, 1.735, 1.760, 1.785}; // Pulse values for starting




void setup() {
  // Setup code
  pinMode(pulsePin, OUTPUT);                        //Sets the pin as an output
  pinMode(hallPin, INPUT);                          // Sets the hall pin an input
  Scheduler.startLoop(loopINJ);                     // Configures injector loop to run
indefinitly as part of the task scheduler
  Scheduler.startLoop(loopRPM);                     // Configures RPM loop to run
indefinitly as part of the task scheduler
  Scheduler.startLoop(loopO2);                      // Configures O2 loop to run
indefinitly as part of the task scheduler
  Scheduler.startLoop(loopMAP);                     // Configures MAP loop to run
indefinitly as part of the task scheduler
  attachInterrupt(digitalPinToInterrupt(hallPin), Trig, FALLING);   // Configures
interrupt

  Serial.begin(9600);                 // Starts serial communication at 9600 baud
  Serial.println("Initialized");      // Confirms system is running
}




void Trig() {
  // Event based trigger for the injector
  INJTrig = 1;
}




// Injector Loop
void loopINJ() {
  // Batch injection cycle
  digitalWrite(pulsePin, LOW);    // Resets pulse pin to low
  int RPM_LK;                     // Creates lookup RPM variable
  int TPS_LK;                     // Creates lookup TPS variable
  TPS = analogRead(tpsPin);       // Reads analog value from TPS
  RPM = rpm_val;

  if (RPM < 1000) {
```

```
    // Starting Procedure
    // Calcualtes whole number to lookup RPM in pulse table
    RPM_LK = min(round(((RPM - 250) / 50) - 1), 12); // Sets pulse to highest value
in starting procedure

    pulse = pulseStart[RPM_LK];      // Uses RPM lookup value to find injector pulse
width
  }
  else {
    // Running Procedure
    RPM_LK = min(round(((RPM - 1200) / 50) - 1), 59);                 // Calcualtes
whole number to lookup RPM in pulse table
    TPS_LK = min(round((((TPS - TPS_L) / (TPS_H - TPS_L)) * 20) - 1), 21); //
Calcualtes whole number to lookup TPS in pulse table
    pulse = pulseWidth[RPM_LK][TPS_LK];                               // Uses lookup
values to find injector pulse width from table
  }


  pulse = pulse * MAP_cf * O2_cf * 1.9;   // Calculates pulse with correction value
  // Temperature correction
  temp = analogRead(tempPin);
  if (temp >= 500) {
    // Cold Start
    pulse = pulse * 1.05;
  }
  else if (temp >= 700) {
    // Warming Up
    pulse = pulse * 1.025;
  }

  if (INJTrig != 0) {

    INJTrig = 0;
    digitalWrite(pulsePin, HIGH);     // Turns on injector
    delay(min(pulse, 10));           // Keeps injector on for duration of pulse
width
    digitalWrite(pulsePin, LOW);      // Turns off injector

    inj = 1;
  }

  digitalWrite(pulsePin, LOW);      // Turns off injector
  delay (1);
}




void loopRPM() {
  // Preallocate variables for tachometer
  int hall_count = 0;          // Resets hall count
  float start = micros();      // Starts timer for tachometer
  bool hall_state = false;     // Resets hall state
  // Counting number of times the hall sensor is tripped
  while (true) {  // Runs infinitly till the loop is broken
    if (digitalRead(hallPin) == 0) {  // Runs if hall sensor is tripped
```

```
      if (hall_state == false) {       // Condition that runs if the hall state
wasn't previously run
        hall_state = true;             // Sets hall to true so hall doesn't trigger
twice
        hall_count += 1;               // Increments hall count
        delay(5);                      // Delay between reads to avoid double count
      }
    } else {
      hall_state = false;              // Keeps hall state false while sensor is open
    }

    if (hall_count >= thresh) {        // Breaks loop when hall count is reached
      break;
    }
    delay(1);
  }

  // Calculate RPM
  float end_time = micros();                                  // Gets end timer for RPM
calculation
  float time_passed = ((end_time - start) / 1000000.0); // Calculates time passed
over the RPM period
  rpm_val = (hall_count / time_passed) * 60.0;          // Calculates RPM and stores
it in the global variable for print

  delay(10);   // Delay for stability
}



void loopO2() {
  O2 = analogRead(O2Pin);
  O2_cf;

  if (O2 < 25) {
    // Not up to temp
    lean_c = 0;
    rich_c = 0;
    O2_cf = 1;
  }
  else if (O2 < 85) {
    // Lean
    lean_c = min(20, (lean_c + 1));
    rich_c = 0;

    O2_cf = 1 - (lean_c * 0.0025);
  }
  else if (O2 > 250) {
    // Rich
    rich_c = min(20, (rich_c + 1));
    lean_c = 0;

    O2_cf = 1 + (rich_c * 0.0025);
  }
  else {
    // Stoich
```

```
    rich_c = 0;
    lean_c = 0;
    O2_cf = 1;
  }

  delay(10);   // Delay for stability
}



void loopMAP() {
  double MAP_in = analogRead(mapPin);
  if (RPM < 1200) {
    if (RPM > 3500) {
      if (MAP_in > 950) {
        // No Load High RPM
        MAP_cf = 0.95;
      }
      else if (MAP_in > 750) {
        // Load
        MAP_cf = 1.05;
      }
      else if (MAP_in < 400) {
        //Throttle Cut
        MAP_cf = 0.3;
      }
      if (MAP_in > (MAP_p + 25)) {
        // Extra Load
        MAP_cf = 1.075;
      }
    }

    if (RPM < 2000)
      if (MAP_in > 900) {
        // Full throttle at low speed
        MAP_cf = 1.1;
      }
  }
  else {
    MAP_cf = 1;
  }

  MAP_p = MAP_in;
  delay(10);   // Delay for stability
}



void loop() {
  // Loop to serial print variables in batch
  Serial.println(" ");                      // Prints new line
  Serial.print(rpm_val);                    // Prints the calculate RPM value
  Serial.println(" RPM");                   // Prints ASCII string and creates new
line
  Serial.print("O2 Voltage : ");            // Prints ASCII string
```

```
  Serial.println(O2);                          // Prints O2 voltage
  Serial.print("Count (Lean, Rich) : ");       // Prints ASCII string
  Serial.print(lean_c);                        // Prints lean condition count
  Serial.print(", ");                          // Prints comma and space
  Serial.println(rich_c);                      // Prints rich condition count
  Serial.print("O2 Correction Factor : ");     // Prints ASCII string
  Serial.println(O2_cf);                       // Prints O2 correction factor
  Serial.print("Pulse Width : ");              // Prints ASCII string
  Serial.println(pulse);                       // Prints O2 correction factor

  if (inj == 1) {
    // Prints only if injector cycles
    Serial.println("Injector Loop Works");
    inj = 0;
  }

  delay(2500);    // Delay to provide readability
}
```