Implementation of Engine control model on NI Labview

Author

Ahmed Zafar	10-NUST-BE-ME-09
Aksam Mukhtar	10-NUST-BE-ME-17
Maaz Hussain	10-NUST-BE-ME-45
Muhammad Nehal Akhtar	10-NUST-BE-ME-61
Umar farooq Ghumman	10-NUST-BE-ME-90

Supervisor

Dr. Samiur Rahman SHAH

DEPARTMENT OF MECHANICAL ENGINEERING

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

Contents

Abstract

Engine's performance and its efficiency has been the subject of interest since the invention of first internal combustion engine. The advancement in electronics lead to the invention of Engine control units (ECUs).

Before ECUs, air/fuel mixture, ignition timing, and idle speed were mechanically set and dynamically controlled by mechanical and pneumatic means.

An engine control unit (ECU), is a type of electronic control unit that controls a series of actuators on an internal combustion engine to ensure optimal engine performance. It does this by reading values from different sensors within the engine bay, interpreting the data using multidimensional performance maps (called lookup tables), and adjusting the engine actuators accordingly.

We have developed an engine control model on NI Labview which is a graphical programming software. We have used NI Labview 2013 software to develop our model because it provides a user friendly programming interface for programmers. Our engine control model is compatible with any spark ignition engine. It ensures performance of engine by controlling fuel injection and spark timing. We have used two different engine control strategies depending on the type of sensor used i.e Manifold Absolute Pressure (MAP) sensor and Mass Air Flow (MAF) sensor.

Our software also encompasses different correction factors like temperature correction, pressure correction, acceleration enrichment, and lambda feedback.

Our engine control model is successfully tested on the apparatus that we have designed to control fuel injection by reading real time values from Air mass flow sensor, crank sensor and temperature sensor with the help of NI Data Acquisition (DAQ) card.

This model can be used by industry and car tuners to tune their engines on a test bed for optimal performance and efficiency.

Acknowledgement

Thanks to Allah Almighty for providing us the chance to contribute to scientific Knowledge, strength to persist through hardships and the motivation to see our project to its completion. We would like to express our humble gratitude to National University of Sciences and Technology for providing us the opportunity to get ourselves educated in such a fascinating way.

We would like to acknowledge our project supervisor Dr. Sami-ur-Rehman Shah for his supervision, dedication, invaluable guidance and constructive advice throughout the course of research. His efforts in regards of giving us motivation, encouragement and the most valuable practical solutions made it possible to complete the project. We are also very grateful to Dr. Mushtaq for his valuable advice in determining the scope of project and his helpul criticism without which the project could not have seen completion.

Special thanks to Dr. Mohsin Jamil, Faculty Incharge of Advanced Controls Lab for allowing us to use Lab in late hours.

We would especially like to thank Mr. Usman Chatta, MS student of Robotics and Intelligent Machine Engineering who guided us in Labview software.

We would also like to thank our MRC staff for cooperating with us in hardware problems and to provide us technical support necessary for the completion of the project.

CHAPTER 1
Literature Review

1.1 Working of Internal Combustion Engine:

Internal combustion engines are called that because the fuel is burned inside the working part of the engine (the cylinder) as opposed to the fuel being burned remotely (as in a steam engine, for example). Jet engines are internal combustion engines, but unlike automotive engines, they are not spark-ignition (they are continuously ignited by already burning fuel). The scope of project is limited to internal combustion, spark ignition engines. We begin by explaining how a 4-stroke cycle engine works (by far the most common type of automotive engine).

An engine has three primary 'control parameters' that we can manipulate to optimize the way the engine runs under various conditions:

- The amount of air going into the engine,
- The amount of fuel being mixed with the air that goes into the engine,
- The timing of the spark to ignite the air fuel mixture.

An engine has one or more cylinders (if it isn't a rotary engine, etc.). These cylinders have a moveable piston in them. The piston seals the lower end of the cylinder, and because it is connected to a rotating crankshaft by a connecting rod, it moves from the bottom of the cylinder to the top (and back, repeating endlessly).

For the engine to operate, it has 4 cycles, each of which take one-half a crankshaft revolution, which is one 'stroke' up or down the cylinder. The strokes are:

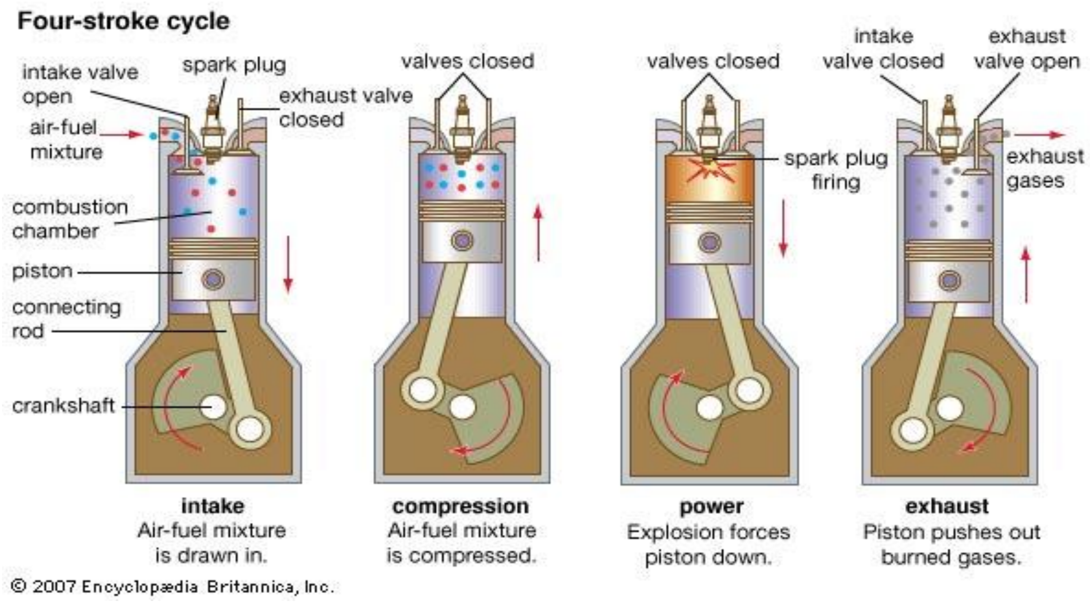


Figure 1.1

1.2 Engine Management system

Engine management system ensures that engines run cleanly and efficiently in a wide variety of conditions. It controls the running of an engine by monitoring the engine speed, load and temperature and providing the ignition spark at the right time for the prevailing conditions and metering the fuel to the engine in the exact quantity required.

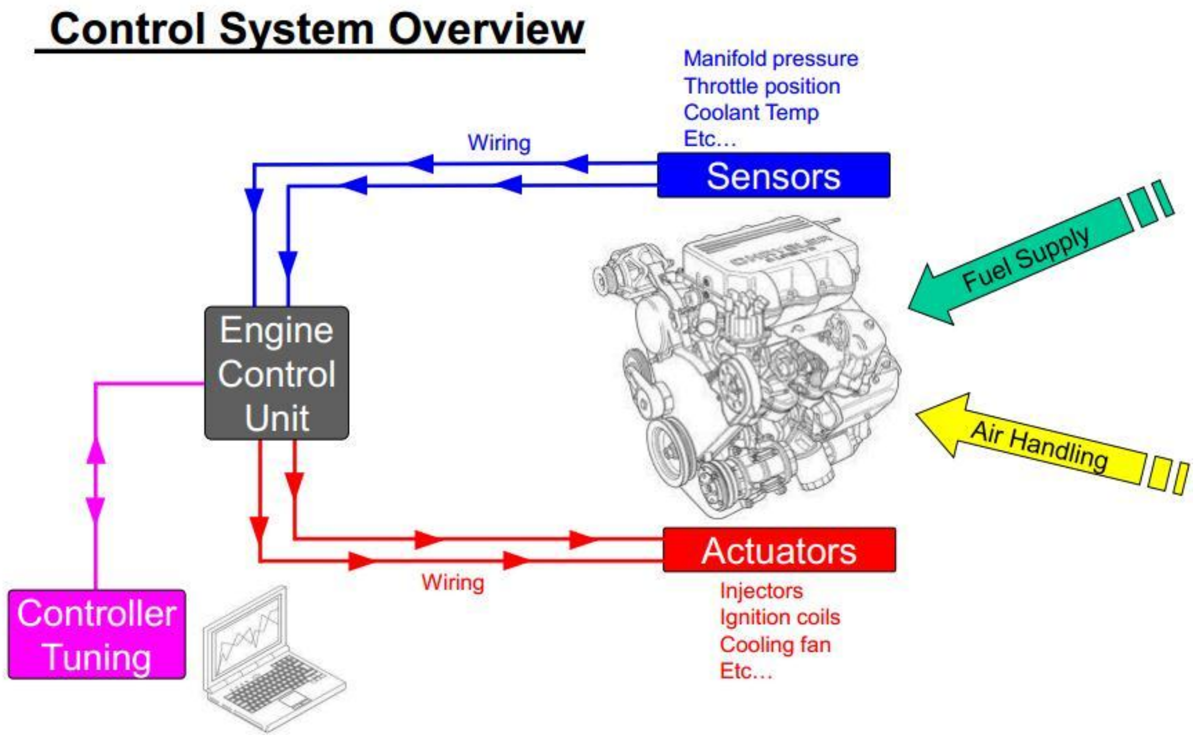


Figure 1.2

The engine control system includes:

- **Sensors** for the detection of the engine operating modes
- **Electronic control unit** (ECU) which elaborates the signal values supplied by the sensor, according to defined control strategies and algorithms, and defines the actions to be delivered to the actuators
- **Actuators** which have the task to actuate the defined commands.

Two main operations which are controlled by Engine Management System (EMS) are:

1. Fuel Injection

2. Spark ignition

1.3 Electronic Fuel Injection

1.3.1 Theory:

The amount of air going into the engine is primarily determined by the throttle (as well as any limitations based on the port and valve design, cam timing, etc.). The throttle can be opened anywhere from 0% to 100%. Larger openings mean more air going into the engine in general, and more power output from the engine. The fuel must be in a narrow range in proportion to the air. The exact ratio varies. The chemically correct ratio is called 'stoichiometric'. More fuel is 'rich', less fuel is 'lean'. Stoichiometric mixtures are around 14.7:1 for gasoline (by mass).

Stoichiometric air/fuel ratios are not necessarily the optimal target for best power or economy though. For best power, you will want to run rich, for best economy you will want to run lean of stoichiometric:

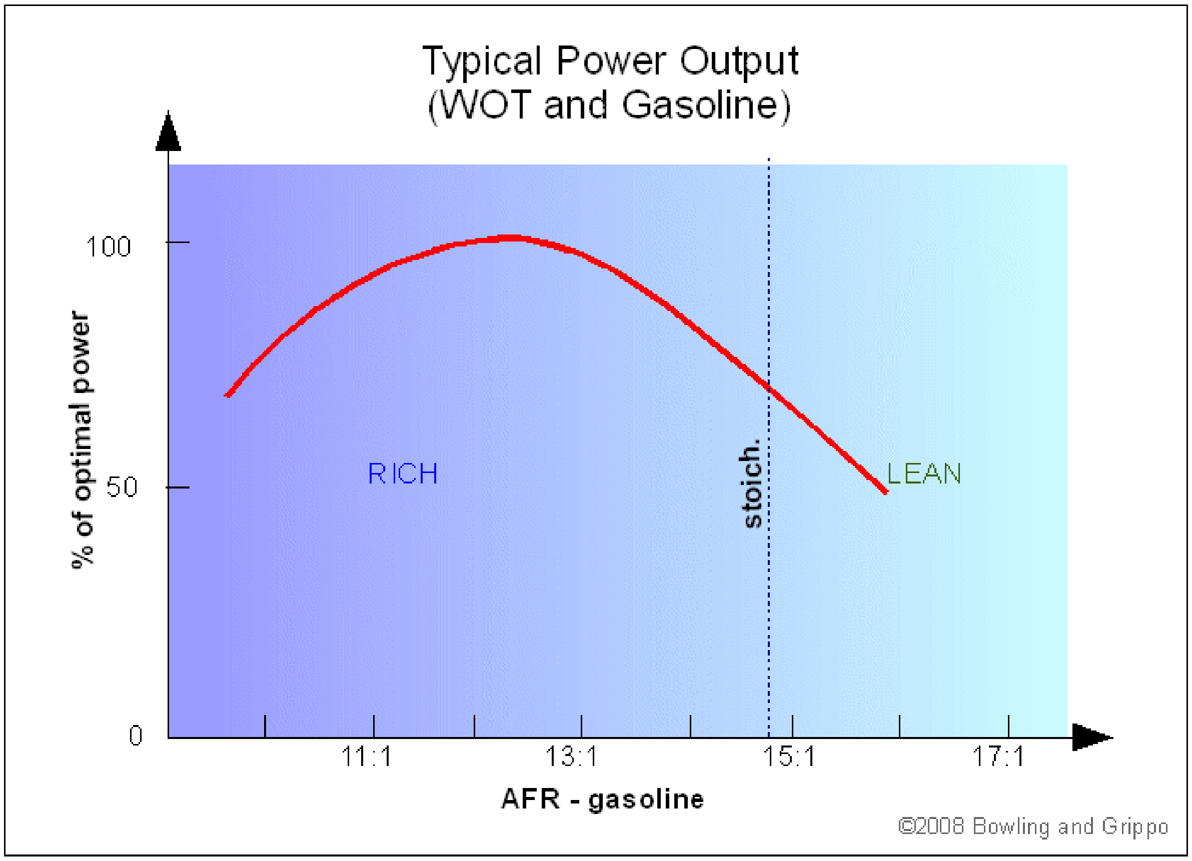


Figure 1.3

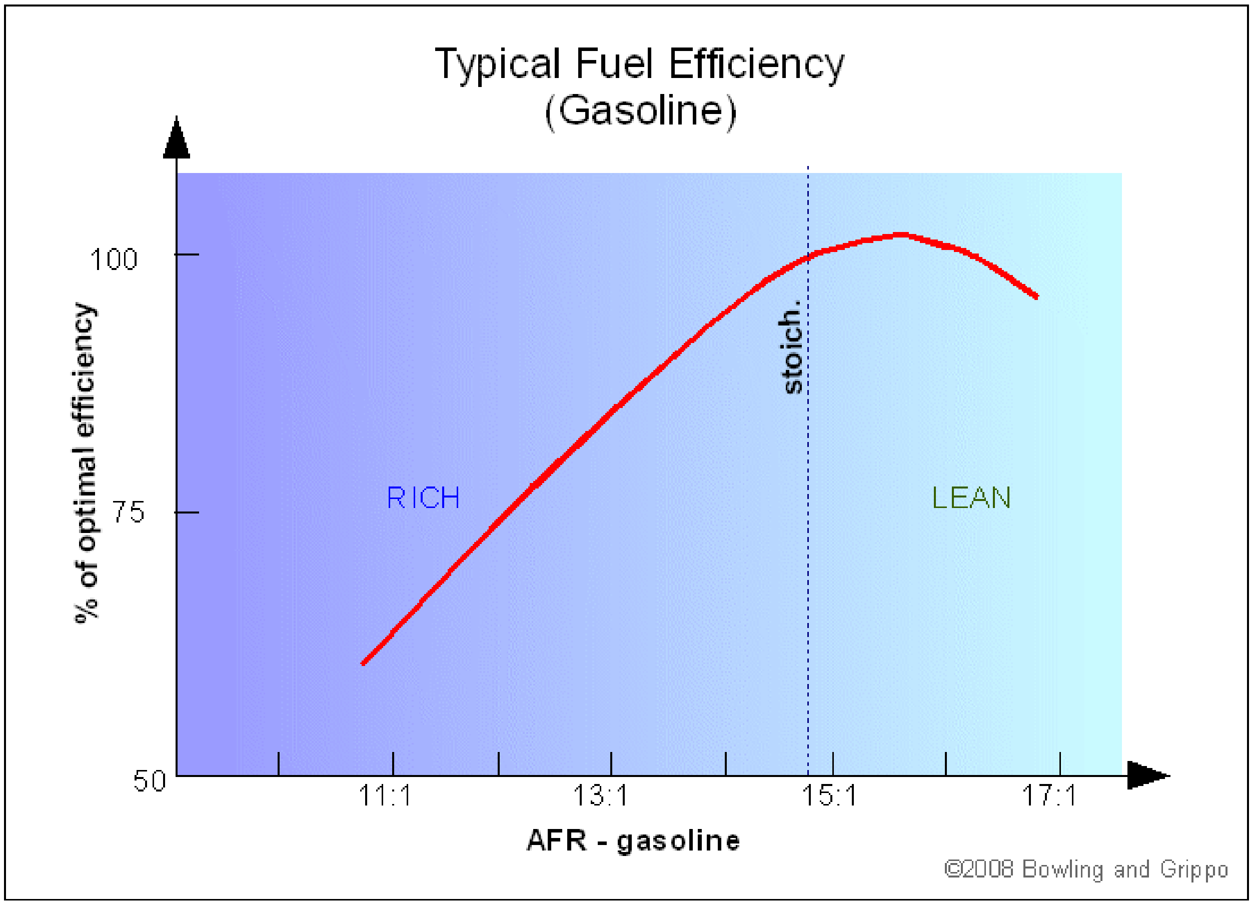


Figure 1.4

1.3.2 Factors affecting Fuel Injection:

Engine temperature:

When an engine starts from cold it is well below its normal operating temperature, this causes some of the fuel injected into the engine to condense rather than atomizing and being drawn in efficiently. Combustion chamber temperatures are also low which leads to incomplete and slow combustion. These affects cause the engine to run weak and require that extra fuel be supplied to the engine to compensate. In a conventional system the "choke" on the carburetor performs this function, on an injection system a coolant temperature sensor provides the EMS with the engines temperature and enables it to "correct" the fuelling. This correction involves adding a percentage of extra fuel according to a pre-determined correction profile by temperature, up to the normal operating temperature of the engine. The amount of extra fuel will vary from engine to engine and according to engines temperature and RPM since the effects of condensing are less when air speeds are higher.

Air temperature:

When air temperatures are high, the density of the air being inducted falls off, thereby lessening the volume of Oxygen available for combustion, if the fuel that is injected remains constant then the mixture will become too rich. To compensate for this the EMS applies a correction to the base map according to a predetermined correction profile. As the air temperature rises so air density will continue to fall and hence the fuelling will be reduced. Information about air temperature is relayed to the EMS by an air temperature sensor. To an extent air flow meters can compensate for lower density air since depending on their type they may show less volume of air inducted.

Mixture strength:

Some EMSs make use of a Lambda sensor that sits in the exhaust of an engine and measures the “strength” of the mixture while the engine is running. During conditions of steady state running the EMS is able to tell from this sensor whether the mixture is rich or lean and can make real-time adjustments to bring the mixture back to chemically correct. This generally happens only when in steady state, E.G. at idle or when cruising and is known as “closed loop running”. Over a period of time the EMS can “learn” whether the mixture is rich or lean and make long term adjustments.

Acceleration fuelling:

When the throttle is opened suddenly there is generally a weakening effect on the induction since air is lighter than fuel and is drawn in more rapidly. Weakening on throttle opening transients is also caused by the fact that the fuel has already been injected and the inlet valve is open before changes in the inlet manifold can take place due to a throttle. Transient, this is only a transitory affect but it can cause the engine to stumble or stutter on initial acceleration. To counteract this tendency the EMS can keep track of sudden changes in throttle position or load and add a percentage of extra fuel when this happens. The extra fuel is only added for a short period and is then decayed over another short period; this is normally a number of engine revolutions rather than a period of time. This is known as “accelerator clamp”.

Injector duty cycle:

In order to inject a fuel into the engine the injector is opened for a period of time, known as the pulse width, this time is always the same for a given quantity of fuel, regardless of engine speed. As engine RPM increases the time available per revolution to fire the injector is less, at 6000RPM the time available is exactly half the time at available at 3000RPM. As this injection opportunity gets progressively smaller the injectors are required to fire much more frequently; this can result in the injector being open almost all the time. When the injection system used is sequential the requirement is to be able to deliver the fuel at a time when the inlet valve is closed; this further reduces the injector's opportunity to fire. The percentage of time that the injector is open is known as the "duty cycle" and this represents the relationship between the times the injector is closed measured against the time it is open. If the duty cycle goes above 90% anywhere in the rev band (I.E. the injector is open for more 90 percent of the time) then the injector capacity is being reached and the engine may require larger injectors.

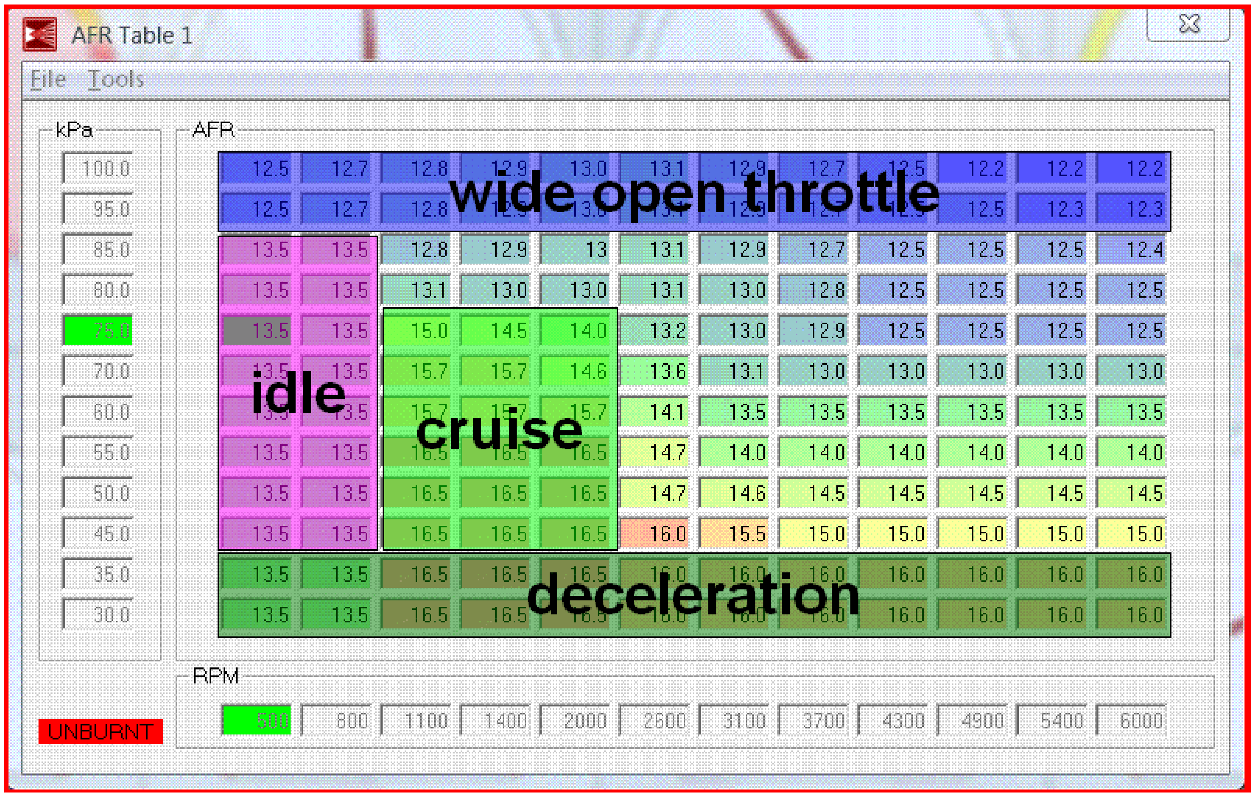
1.3.3 Tuning Fuel:

To adjust the amount of fuel to correct a lean condition, we increase the parameter value (whether it is in % or milliseconds). The parameter we want to increase may be in the AFR table, the accel enrichments, the warm-up enrichments, the afterstart enrichments, or the cranking pulse widths (among others). Which parameter we adjust depends on the conditions under which we find the engine is lean. Conversely, if the engine is rich, we decrease the appropriate parameter.

For maximum power, we want to run richer than stoichiometric. This is because the engine's output is primarily limited by the amount of air that enters the cylinders. That, in turn, limits the amount of fuel we can burn. However, to make sure all of the oxygen is consumed, we must supply a richer than stoich. mixture, so that any residual oxygen always has fuel nearby to combust with. The result is that maximum power typically occurs between 12.5:1 and 13:1 (if the ratio is much richer than that, the excess fuel actually quenches the flame front).

It may also be true that the engine wants to idle rich of stoich., especially if it has an aftermarket camshaft. A 'hot street' engine may idle best at 13:1 to 14:1 (where it will achieve minimum MAP kPa, which should be the tuning goal for idle). However, for emissions regulated applications with a catalytic converter, the idle mixture is usually stoichiometric in order to maximize the conversion efficiency.

For one naturally aspirated engine, here is an example of a target AFR table:



kPa \ RPM	500	800	1100	1400	2000	2600	3100	3700	4300	4900	5400	6000
100.0	12.5	12.7	12.8	12.9	13.0	13.1	12.9	12.7	12.5	12.2	12.2	12.2
95.0	12.5	12.7	12.8	12.9	13.0	13.1	12.9	12.7	12.5	12.5	12.3	12.3
85.0	13.5	13.5	12.8	12.9	13	13.1	12.9	12.7	12.5	12.5	12.5	12.4
80.0	13.5	13.5	13.1	13.0	13.0	13.1	13.0	12.8	12.5	12.5	12.5	12.5
75.0	13.5	13.5	15.0	14.5	14.0	13.2	13.0	12.9	12.5	12.5	12.5	12.5
70.0	13.5	13.5	15.7	15.7	14.6	13.6	13.1	13.0	13.0	13.0	13.0	13.0
60.0	13.5	13.5	15.7	15.7	15.7	14.1	13.5	13.5	13.5	13.5	13.5	13.5
55.0	13.5	13.5	16.5	16.5	16.5	14.7	14.0	14.0	14.0	14.0	14.0	14.0
50.0	13.5	13.5	16.5	16.5	16.5	14.7	14.6	14.5	14.5	14.5	14.5	14.5
45.0	13.5	13.5	16.5	16.5	16.5	16.0	15.5	15.0	15.0	15.0	15.0	15.0
35.0	13.5	13.5	16.5	16.5	16.5	16.0	16.0	16.0	16.0	16.0	16.0	16.0
30.0	13.5	13.5	16.5	16.5	16.5	16.0	16.0	16.0	16.0	16.0	16.0	16.0

Figure 1.5

1.3.4 Injection types:

There are two common sorts of injection:

1. **Throttle body injection** - usually one or two injecotrs for the whole engine
2. **Port injection** (aka. Multi-Port) - one injector per cylinder

Then there are three common modes of injection timing:

- **batch** - all injectors fire at once, but not timed to any specific cylinder,
- **bank** - ½ the injectors fire at once, but not timed to any specific cylinder,
- **sequential** - each injector fires at a specific point in the 4-stroke cyle for each cylinder (i.e., 8 independent timing events)

Throttle body injected cars are usually batch or bank fire, simply because of the geometry. Most port injection set-ups before the mid-1990s were bank fire as well (including GM Tuned Port Injection for the 305/350).

Sequential injection requires:

- as many injectors as you have cylinder, with one dedicated to each cylinder (i.e., not a 4 injector TBI on a 4 cylinder).
- as many injector drivers as you have cylinders,

- and also requires a camshaft position sensor (a crank sensor is not adequate for a 4-stroke cycle engine).

The benefits of sequential injection are that:

- you may get slightly better mileage and lower emissions at low engine speeds,
- you can tune each cylinder's fuel amount independently (if you know how).

1.4 Spark Timing

1.4.1 Theory:

The ignition system on car has to work in perfect concert with the rest of the engine. The goal is to ignite the fuel at exactly the right time so that the expanding gases can do the maximum amount of work. If the ignition system fires at the wrong time, power will fall and gas consumption and emissions can increase.

When the fuel/air mixture in the cylinder burns, the temperature rises and the fuel is converted to exhaust gas. This transformation causes the pressure in the cylinder to increase dramatically and forces the piston down.

In order to get the most torque and power from the engine, the goal is to maximize the pressure in the cylinder during the **power stroke**. Maximizing pressure will also produce the best engine efficiency, which translates directly into better mileage. The timing of the spark is critical to success.

There is a small delay from the time of the spark to the time when the fuel/air mixture is all burns and the pressure in the cylinder reaches its maximum. If the spark occurs right when the piston reaches the top of the compression stroke, the piston will have already moved down part of the way into its power stroke before the gases in the cylinder have reached their highest pressures.

To make the best use of the fuel, **the spark should occur before the piston reaches the top of the compression stroke**, so by the time the piston starts down into its power stroke the pressures are high enough to start producing useful work.

Work = Force * Distance

In a cylinder:

- **Force** = Pressure * Area of the piston
- **Distance** = Stroke length

So cylinder **work = pressure * piston area * stroke length**. And because the length of the stroke and the area of the piston are fixed, the only way to maximize work is by increasing pressure.

Advance refers to the precise crankshaft position where the ignition is initiated by a spark from the spark plug. It is always referenced to the crankshaft position in degrees (the symbol for degrees is °, the same as temperature). Since there are 360° in a crankshaft revolution (or any complete circle), one intake stroke, which takes ½ a revolution, is 180°. Normally the advance is specified as 'before top dead center' (BTDC). This means the number of degrees the crankshaft would have to turn to reach the very top of it travel from the spark point.

Sparking before TDC is necessary because the fuel and air take some milliseconds to burn. Typical values range from 5 degrees BTDC at idle to 35 degrees at wide open throttle (WOT) and possibly even higher under cruise conditions. The flame front moves at about 50 mph (~73 feet/second or ~880 inches per second) at high cylinder pressures and appropriate AFRs. The pistons can travel a considerable distance in the time it takes for the fuel to burn all the way from the spark plug to the most distant regions the cylinder. For example, at 880 in./sec and a 3.5" bore, if the spark was centrally located, the burn would take 1.75/880 = 2.0 milliseconds.

If the burning takes 2 milliseconds to reach maximum pressure, at 3000 rpm the piston & crank will travel 36° in that time. There is an optimal point (peak pressure position - ppp) in the piston movement when we want the burning gases to reach their peak pressure (usually about 17° ADTC), so we need to start the burn early to get the peak pressure where we want it (in this case 36°-17° = 19° BTDC).

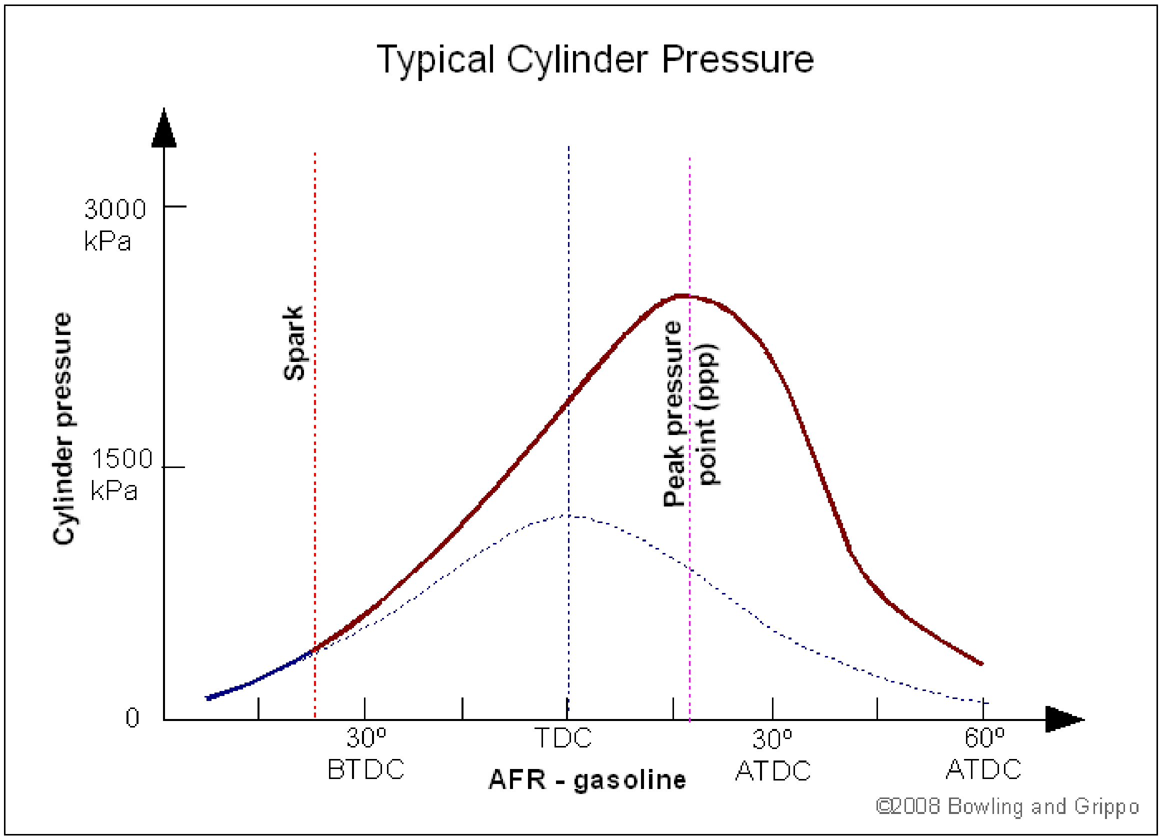


Figure 1.6

With a larger bore and a non-centrally located spark plug (typical of 2-valve engines), more advance is needed. For example, on a 4.00" bore, with a spark plug 1.3" from one side (and 2.7" from the other), the burn time rises to: 2.7/880 = 3.1 milliseconds. In this time, the piston/crank travels about 55°. So under the same conditions as above, the timing needs to be increased to: 55°-17° = 38° BTDC!

Timing advance is low at low engine speeds, because the piston is moving slowly, and the fuel has time to burn near TDC. At higher speeds, the timing must be advance. At some point (usually about 3000 rpm), the combustion turbulence ensures a quick burn, and no further advance is necessary. The details of how optimal spark advance is affect by various factors would fill a large volume, and include relevant topics like bore size and chamber shape, mixture swirl and tumble, and a myriad of other things...

Too much advance isn't good though. The peak pressure is reached too early, and the result can be that the burn doesn't proceed smoothly across the combustion chamber, but instead fuel and air in the furthest areas of the chamber spontaneously ignite from the pressure and radiant heat in the chamber (this is called 'detonation' and it can be very destructive).

As well, spark and fuel tuning interact. That is, the amount of fuel affects the optimal timing, and vice versa. Here is a graph showing the relationship on one typical gasoline engine:

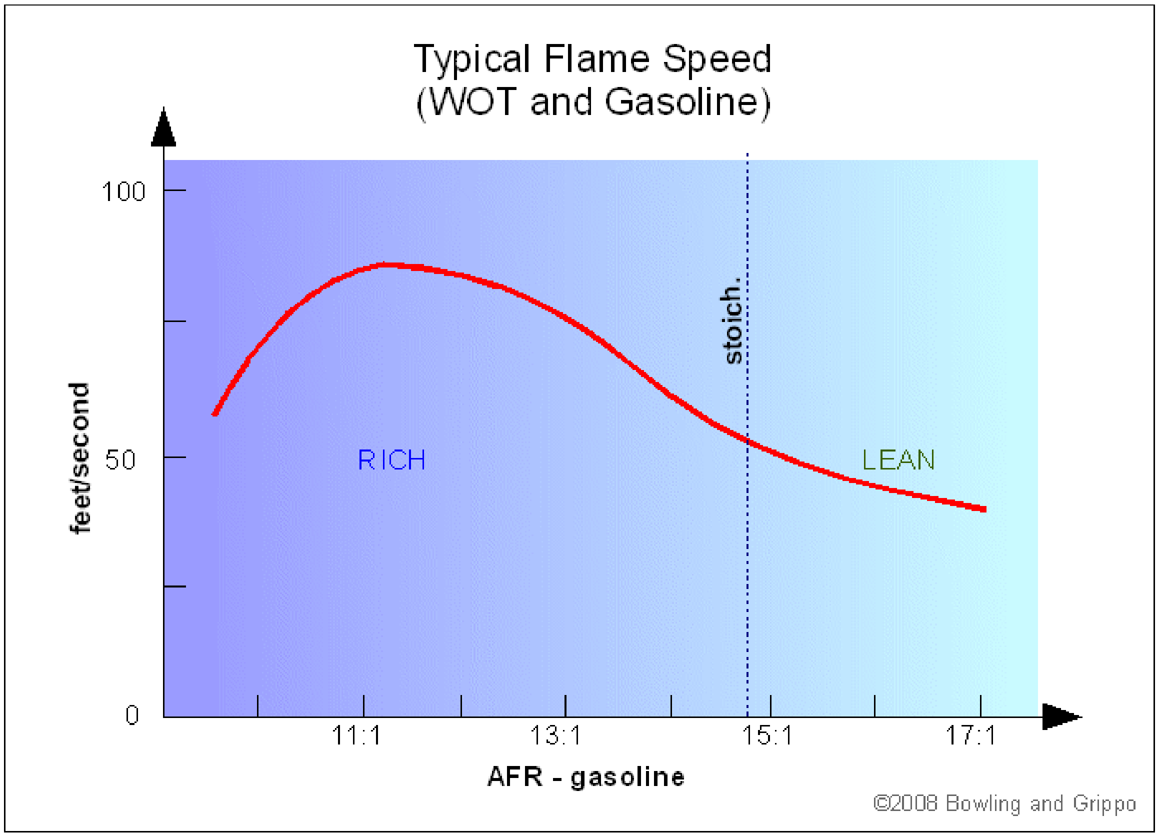


Figure 1.7

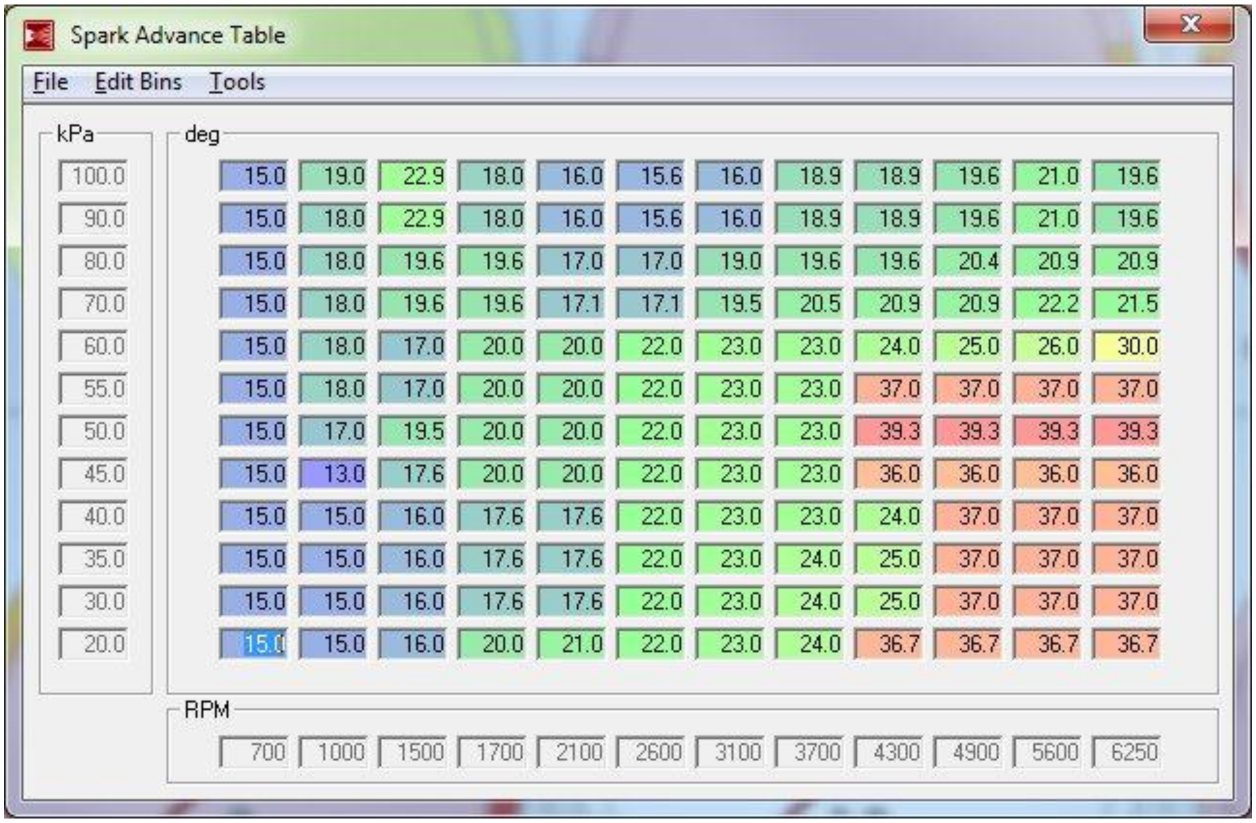
1.4.2 Spark Advance Tuning:

The spark advance value that appears in Labview spark table is the spark advance you should see at the crank with a timing light. To create and tune the spark advance table, you should try to understand what your engine needs in the following areas:

1. **total advance at WOT**: should be from ~24° to ~40° depending on engine bore size and combustion chamber characteristics. Older design engines (i.e. push rods, domed pistons, etc.), and those with large bores (big blocks, etc.) need more advance, about 36 to 38°. Newer designs (4 valve/cylinder, swirl port engines, etc.), and small bores, generally require less, about 28 to 32°.
2. **idle advance**: In Labview, this is the advance at the idle rpm and MAP value. Larger initial advance numbers produce a slightly more fuel efficient idle, but may make the idle unstable and result in higher emissions (this is why most engines use the no vacuum advance at idle). Too much initial advance can also make the engine hard to start. Generally, keep the initial advance at 6° to 10°.
3. **RPM based advance**: This is the advance as read across a row (at a constant MAP kPa). Generally for a performance engine, you want the advance to be maximum by 3000 rpm. So for a given MAP (say 100 kPa) the spark advance should rise from the idle value to the maximum by about 3000 rpm. Your particular settings will depend on your Load and rpm.
4. **vacuum (MAP) advance**: This is the advance as read in a single column of the advance table (at a constant rpm). As the load on the engine is reduced, the fuel burns more slowly and more advance is required. This means that you should have the advance increase for a given rpm as the MAP value decreases in kPa. So, for example, if you have 32° advance at 4000 rpm and 100 kPa, you might have 40° advance at 4000 rpm and 50 kPa.

Note that the optimum amount of total advance is not necessarily the most that doesn't detonate. For example, with a modern cylinder head design, you might get maximum power at 32°, but might not experience any detonation until 38°-40.

The exception to maximizing the total advance is the initial advance the engine uses when cranking. Higher initial advance will generate better 'off-idle' response (especially with an automatic transmission), but can cause hard starting, to the point of physically breaking the starter. Some sources recommend up to 14° to 20° of initial advance for performance engines. However, if engine installed on a high compression, large displacement engine that already puts an additional strain on the starter, limit your initial advance to 4°-12°, then have the advance come in rapidly after 600 to 800 rpm.



deg

kPa	700	1000	1500	1700	2100	2600	3100	3700	4300	4900	5600	6250
100.0	15.0	19.0	22.9	18.0	16.0	15.6	16.0	18.9	18.9	19.6	21.0	19.6
90.0	15.0	18.0	22.9	18.0	16.0	15.6	16.0	18.9	18.9	19.6	21.0	19.6
80.0	15.0	18.0	19.6	19.6	17.0	17.0	19.0	19.6	19.6	20.4	20.9	20.9
70.0	15.0	18.0	19.6	19.6	17.1	17.1	19.5	20.5	20.9	20.9	22.2	21.5
60.0	15.0	18.0	17.0	20.0	20.0	22.0	23.0	23.0	24.0	25.0	26.0	30.0
55.0	15.0	18.0	17.0	20.0	20.0	22.0	23.0	23.0	37.0	37.0	37.0	37.0
50.0	15.0	17.0	19.5	20.0	20.0	22.0	23.0	23.0	39.3	39.3	39.3	39.3
45.0	15.0	13.0	17.6	20.0	20.0	22.0	23.0	23.0	36.0	36.0	36.0	36.0
40.0	15.0	15.0	16.0	17.6	17.6	22.0	23.0	23.0	24.0	37.0	37.0	37.0
35.0	15.0	15.0	16.0	17.6	17.6	22.0	23.0	24.0	25.0	37.0	37.0	37.0
30.0	15.0	15.0	16.0	17.6	17.6	22.0	23.0	24.0	25.0	37.0	37.0	37.0
20.0	15.0	15.0	16.0	20.0	21.0	22.0	23.0	24.0	36.7	36.7	36.7	36.7

RPM

Figure 1.8

CHAPTER 2

Engine Sensors

Any electronic control of a physical system requires feedback of the physical variables through sensors. It is essential that sensors provide an 'accurate enough' perception of the variables involved. In engine control such accuracy is fulfilled by high precision and fast response time sensors essential n high rpm and fuel economy/ high power requirements. Here is a basic overview of the sensors we are using in our engine control project. The sensors used by us are OEM sensors. They had to be calibrated due to the absence of readily available OEM sensor data.

2.1 Mass Air Flow (MAF) sensor:

2.1.1 Working

It is necessary for the engine's computer to calculate and maintain the proper air/fuel ratio for optimum performance and emissions. Engines with "speed-density" fuel injection systems do not have a MAF sensor and use inputs from the throttle position sensor (TPS), manifold absolute pressure (MAP) sensor, incoming air temperature (IAT) sensor and engine rpm to estimate air flow. The MAF sensor is located in the air duct between the air cleaner and throttle body. Here, it can measure all the air that is being drawn into the engine and react almost instantly to changes in throttle position and engine load. The following diagram shows the placement of a MAF in the engine 'breathing tract'.

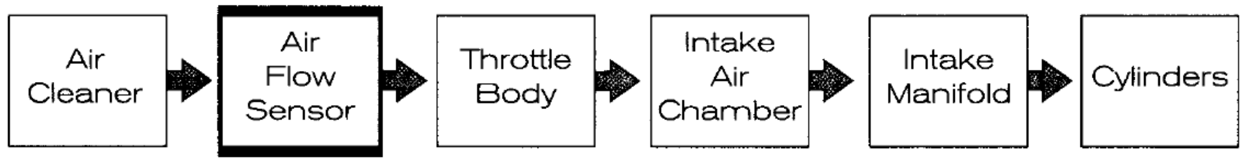


Figure 2.1

There are two basic varieties of MAF sensors: hot-wire and hot-film. Unlike vane air flow (VAF) sensors that have a mechanical spring-loaded flap to measure air flow, MAF sensors have no moving parts. Instead, they use a heated sensing element to measure air flow.

We are using a Toyota Vitz (Toyota Yaris in America) MAF sensor. It employs the cooling effect of air over a platinum hot wire to calculate mass of the air flow across it. A circuit diagram is shown below:

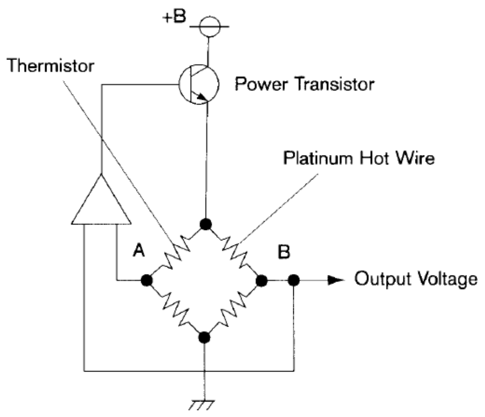
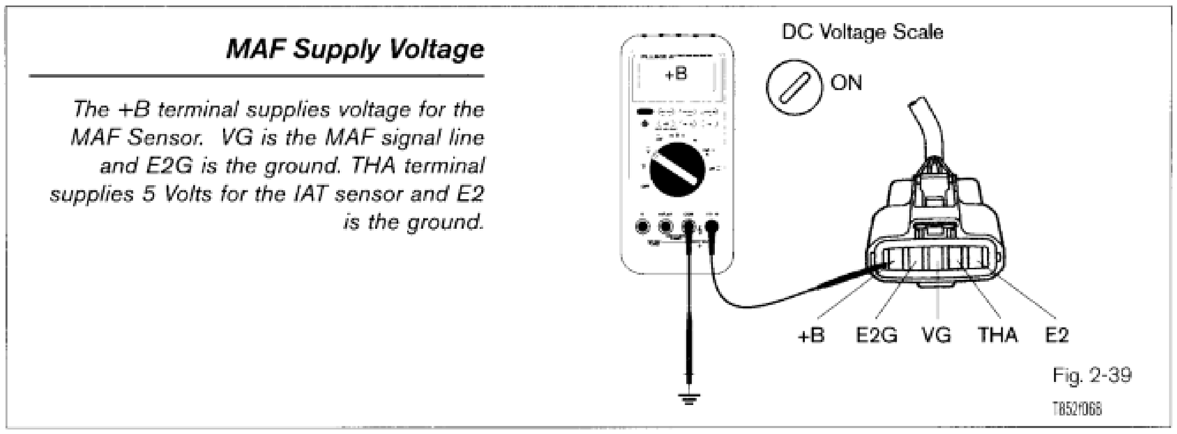


Figure 2.2

The electronic circuitry makes sure that the hot wire remains at a particular temperature in relation to the thermistor allowing more current through the hot wire when more air flows across it to compensate for air's cooling effect. The higher current is then translated into a higher voltage. The MAF also has a temperature sensor to adjust for the different cooling effects of air at different temperatures.


MAF Supply Voltage

The +B terminal supplies voltage for the MAF Sensor. VG is the MAF signal line and E2G is the ground. THA terminal supplies 5 Volts for the IAT sensor and E2 is the ground.

Figure 2.3

- E2G and E2 terminals are ground
- +B terminal supplies 12 V to the MAF sensor
- THA terminal supplies 5 V for intake Air temperature sensor.

- VG gives output voltage according to the mass of air.

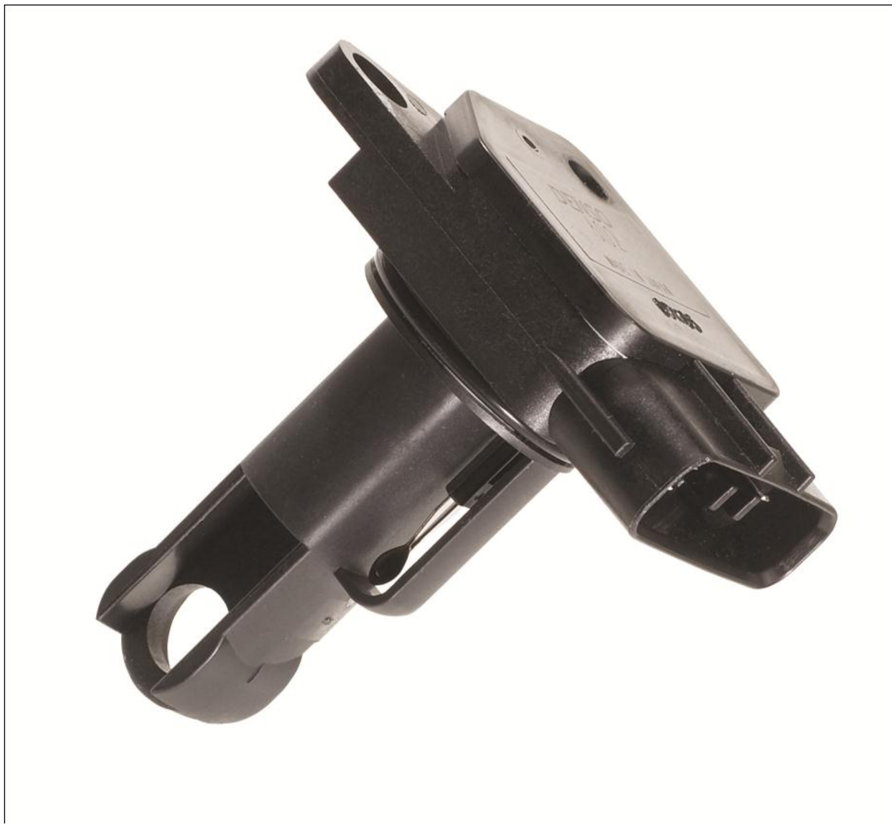


Figure 2.4

We have used denso air mass flow sensor in our hardware model.

We calibrated output voltage by using method describe below:

2.1.2 Calibration

The MAF sensor was placed in a cylindrical tubing with an internal diameter of 6.32cm. An external 12V battery was used to power it and a voltmeter was used to measure the sensor output. This apparatus was driven on a Honda Prider (100cc) motorbike. Care was taken to hold the cylinder opening area perpendicular to the incoming air flow. Ignoring air the boundary layer effect of air passing through the cylinder (air has a very low viscosity) and the resistance offered by the sensor protrusion in the cylinder (due to the large diameter of the tube), it can be reasonably assumed that the air passes through the cylinder at the speed of the motorbike in still air conditions. Moreover, the use of a motorbike ensures little change to the incoming airflow as compared to a car. The following data is the data obtained from this experiment.

speed km/hr	volts	speed m/s	mass flow rate kg/s	lb/hr
0	0.62	0	0	0
12	1.55	3.3333333	0.01170193	42.1269
22	1.73	6.1111111	0.02145354	77.2327
32	1.89	8.8888888	0.03120516	112.338
42	2.05	11.666666	0.04095677	147.444
52	2.2	14.444444	0.05070838	182.502
62	2.35	17.222222	0.06045999	217.656
72	2.49	20	0.0702116	252.761
82	2.63	22.777777	0.07996321	287.867
92	2.77	25.555555	0.08971482	322.973
102	2.9	28.333333	0.09946643	358.079
112	3.03	31.111111	0.10921804	393.185
122	3.15	33.888888	0.11896966	428.290
132	3.27	36.666666	0.12872127	463.396
142	3.38	39.444444	0.13847288	498.502
152	3.48	42.222222	0.14822449	533.608
162	3.57	45	0.1579761	568.714
172	3.66	47.777777	0.16772771	603.819
182	3.75	50.555555	0.17747932	638.925
192	3.83	53.333333	0.18723093	674.031
202	3.91	56.111111	0.19698254	709.137
212	3.98	58.888888	0.20673416	744.243
222	4.04	61.666667	0.21648577	779.348

2.1.3 Graph of Voltage (V) vs Mass flow rate (lb/hr)

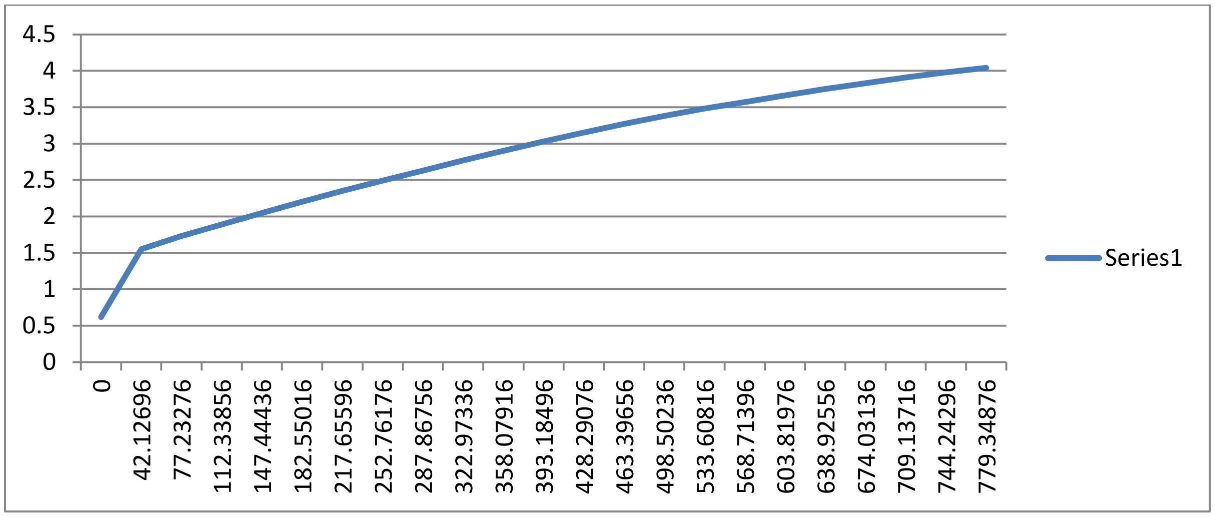


Figure 2.5

Against the Manifold air pressure (MAP) sensor and Volume air flow (VAF) combination, the MAF has advantage of greater adaptability and easier tuning. However, this comes with a few problems:

- They have to be accurate they must be calibrated over the range they will be used in.
- MAF sensors can be calibrated using accurate laminar flow element to determine mass air flow. However, the problem is that when a MAF sensor is bolted to the vehicle, a new set of readings is required for the change in ducting (bends, air filters etc.).

2.2 Engine Coolant Thermistor

2.2.1 Working

The ECT sensor has a thermistor that varies its resistance depending on the temperature of the engine coolant. When the coolant temperature is low, the resistance in the thermistor increases. When the temperature is high, the resistance drops. The variations in resistance are reflected in the voltage output from the sensor.

2.2.2 Calibration:

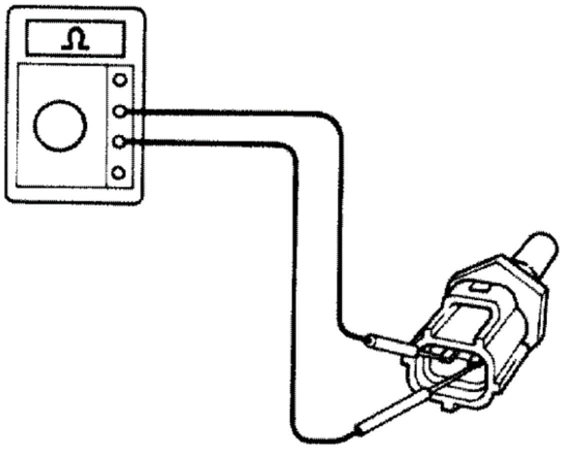
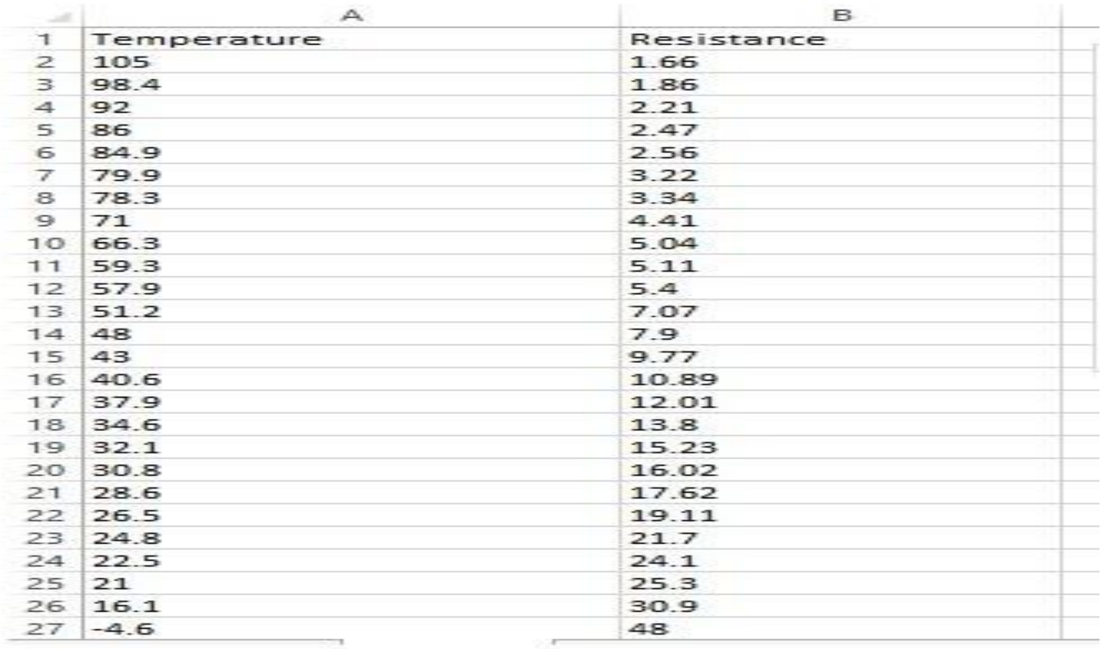


Figure 2.6

A K-type thermocouple from the thermodynamics lab in SMME was used to calibrate a Toyota engine coolant thermistor. Only the resistance circuitry of the thermistor was used. The following procedure was used:

1. Fill the electric kettle with ice cubes and some water.
2. Connect the thermocouple to the terminals in the SOLTEQ temperature measurement board.
3. Connect the thermistor resistor to a multimeter set to measure resistance in the appropriate range.
4. Dip the thermistor and thermocouple probes in the water.
5. Note the thermocouple resistance against the temperature indicated by the SOLTEQ board (reading from) the K-type thermocouple.
6. Turn on the kettle and allow the ice to melt.
7. Follow steps 1-5 until the water boils.

Using this method, the thermocouple resistance was determined. It must be noted that the high salt content in NUST's ground water lead to an impressive temperature range between the melting an boiling points. The following readings were obtained:



	A	B
1	Temperature	Resistance
2	105	1.66
3	98.4	1.86
4	92	2.21
5	86	2.47
6	84.9	2.56
7	79.9	3.22
8	78.3	3.34
9	71	4.41
10	66.3	5.04
11	59.3	5.11
12	57.9	5.4
13	51.2	7.07
14	48	7.9
15	43	9.77
16	40.6	10.89
17	37.9	12.01
18	34.6	13.8
19	32.1	15.23
20	30.8	16.02
21	28.6	17.62
22	26.5	19.11
23	24.8	21.7
24	22.5	24.1
25	21	25.3
26	16.1	30.9
27	-4.6	48

Figure 2.7

These readings resulted in the following graph in excel:

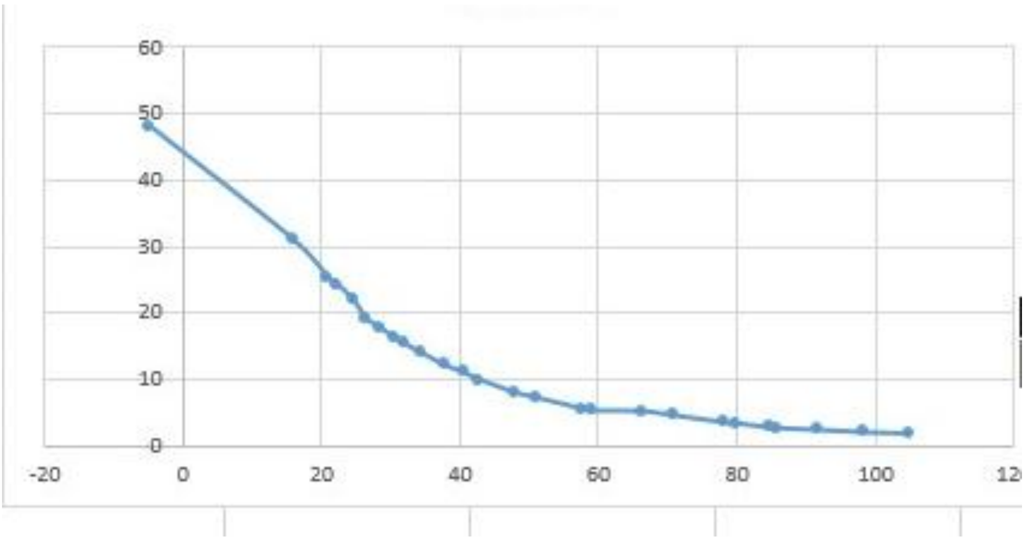
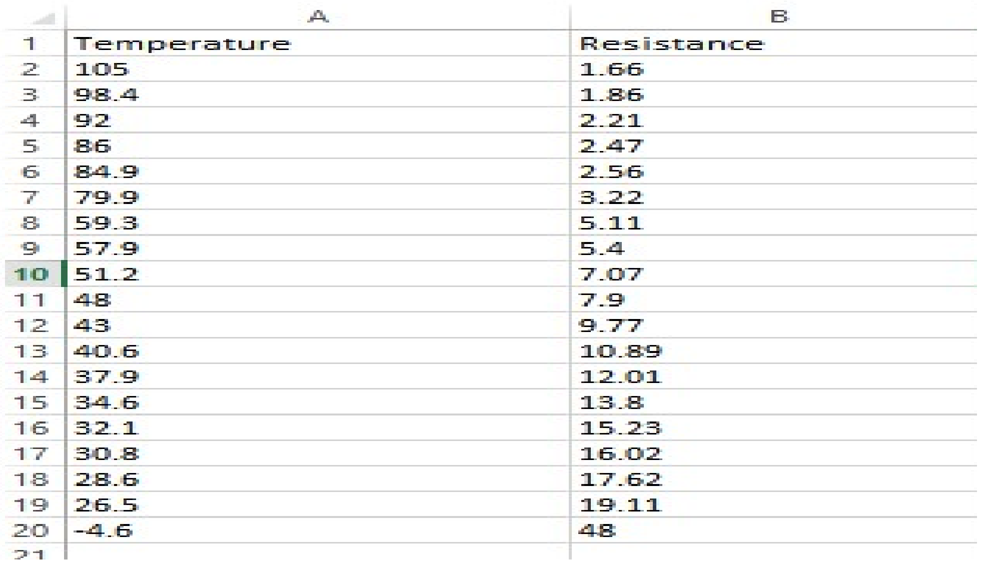


Figure 2.8

This graph was used to get rid of bad readings resulting in the following readings and graph:



	A	B
1	Temperature	Resistance
2	105	1.66
3	98.4	1.86
4	92	2.21
5	86	2.47
6	84.9	2.56
7	79.9	3.22
8	59.3	5.11
9	57.9	5.4
10	51.2	7.07
11	48	7.9
12	43	9.77
13	40.6	10.89
14	37.9	12.01
15	34.6	13.8
16	32.1	15.23
17	30.8	16.02
18	28.6	17.62
19	26.5	19.11
20	-4.6	48
21		

Figure 2.9

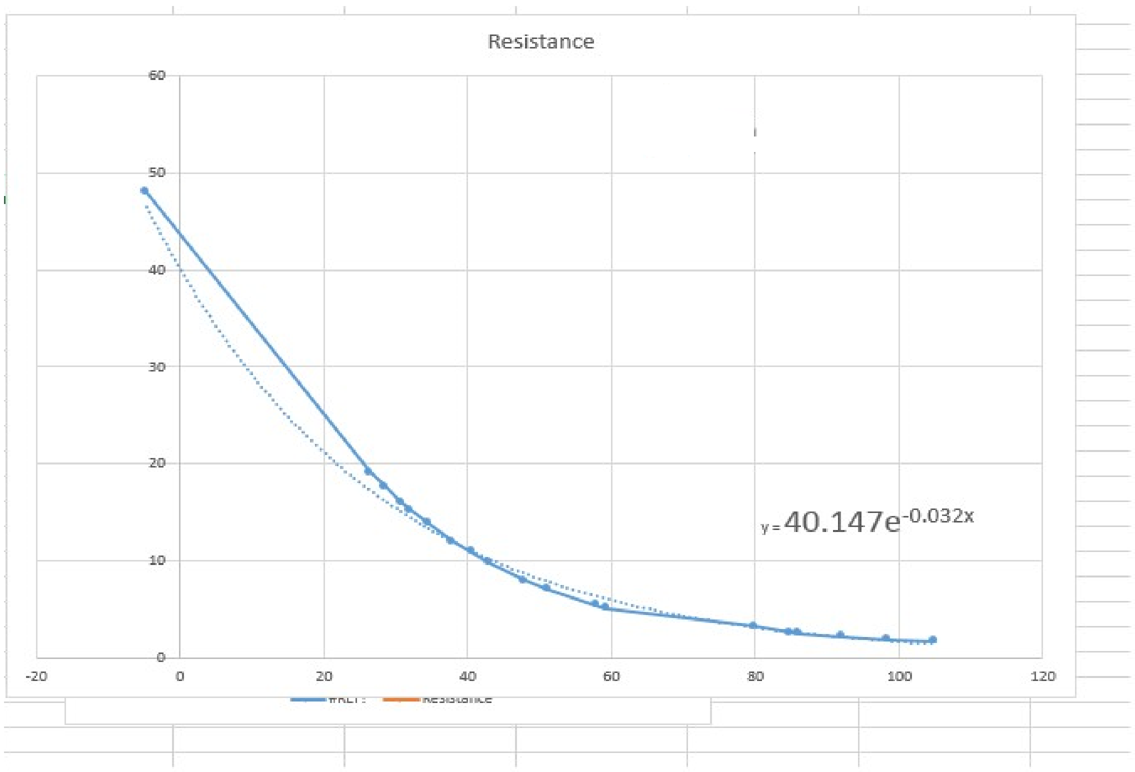


Figure 2.10

The readings were identified as logarithmic in trend under the following criteria for logarithmic functions:

$$[f(x + Dx) / f(x)]^{1/Dx} = b$$

We can recognize this pattern even in data sets without constant intervals between the inputs:

We calculate the various $[f(x + Dx) / f(x)]^{1/Dx}$ we find:

$$\left(\frac{1.86}{1.66}\right)^{\frac{1}{-6.6}} = 0.983$$

$$\left(\frac{1.86}{1.66}\right)^{\frac{1}{-6.6}} = 0.973$$

$$\left(\frac{48}{5.11}\right)^{\frac{1}{-63.9}} = 0.966$$

$$e^{-0.032} = 0.969$$

Hence the equation obtained is: $y = 40.147e^{-0.032x}$.

A constant base of approximately b = 0.969 is hiding in the data.

The thermistor was then placed in a voltage divider configuration: 12V were applied across the thermistor and a 4.7 kilo ohm resistor. By measuring the voltage across the 4.7 kilo ohm resistor, the inverse-to-temperature readings of the thermocouple are converted into readings which increase with increasing temperature. This helps ease the programming in LabView.

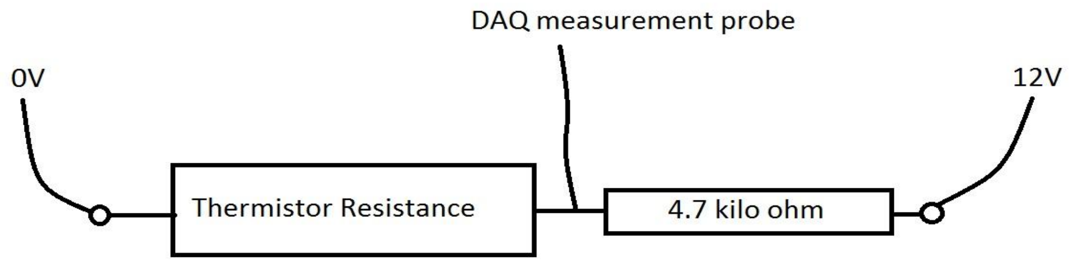


Figure 2.11

2.3 Crankshaft Sensor

The type of crankshaft sensor that we used is variable reluctance sensor. Its normal resistance is 1370 ohm. It has two output wires: ground and pulse.

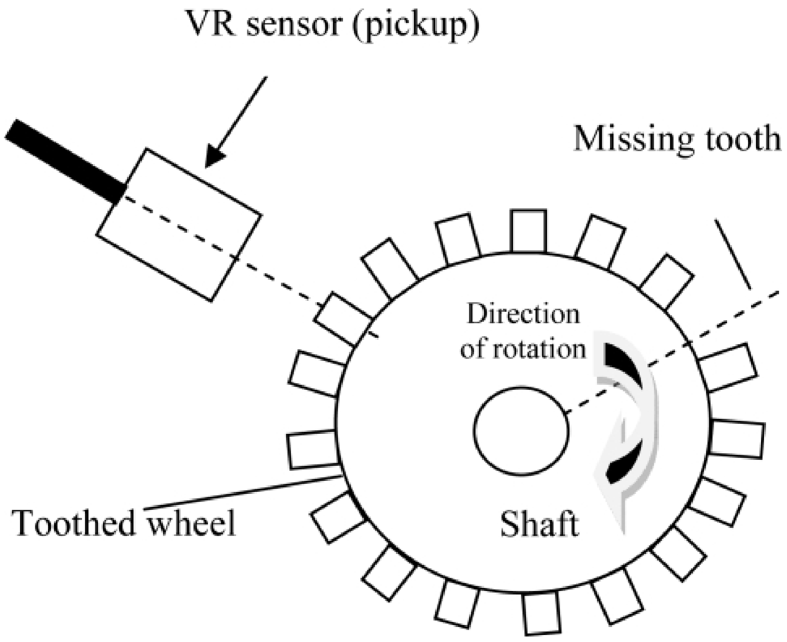


Figure 2.12

2.3.1 Working:

As shown in diagram, the teeth of the rotating wheel (or other target features) pass by the face of the magnet, the amount of magnetic flux passing through the magnet and consequently the coil varies. When the gear tooth is close to the sensor, the flux is at a maximum. When the tooth is further away, the flux drops off. The moving target results in a time-varying flux that induces a proportional voltage in the coil. Subsequent electronics are then used to process this signal to get a digital waveform that can be more readily counted and timed.

2.3.2 Experiment:

We took off the chain cover of a bike (Honda Pridor), and held the crankshaft sensor near the gears of the bike to get peaks. The output was to be displayed on oscilloscope. The bike was turned on and the gears rotated at high speed. But there was no reading on the oscilloscope.

The gears of the bike were not made of a pure ferrous material and the bike was not rotating at high enough a speed. So the output from the sensor was visible.

2.4 Fuel Injector

It is basically an actuator which acts as a valve. It is supplied with fuel at some standard pressure by the fuel pump. And when it is energized by the ECU it atomizes the fuel into a fine mist so that it can burn properly in the internal combustion engine cylinder.

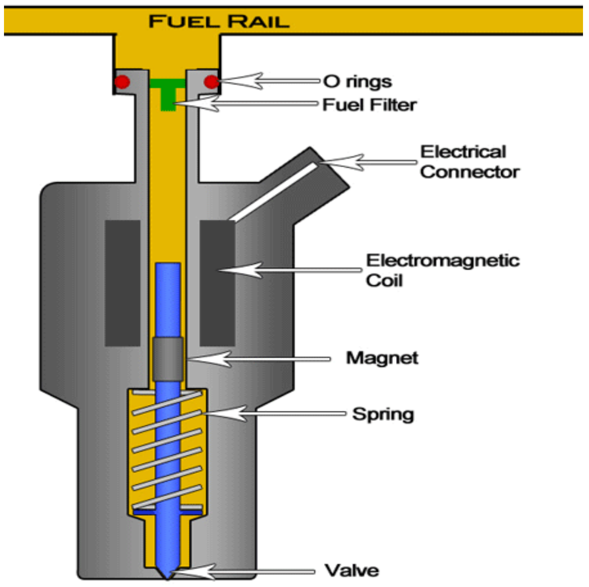


Figure 2.13

2.4.1 Working:

When electrical connector is given a voltage difference current flows through the electromagnetic coil. Consequently a magnetic flus is produced which pushes the magnet upwards against the springs and the valve is opened. When the voltage applied is removed. The energy stored in the spring forces the magnet to come back down and the valve is closed.

2.4.2 Calculation of Injector flow rate:

The injector we have used is SD26 ABD type. We have calculated its flow experimentally. We used a graduated measuring cylinder and a stop watch. The pump was supplied 12V from a battery. And 12V were also supplied to the injector.

The readings were as follows:

Serial no	Time(s)	Volume(cm^3)
1	25	78
2	25	75
3	25	76

Average flow rate: 3.05cm^3/s

CHAPTER 3

Engine Control Strategies

Each EFI car today is equipped with either a MAF sensor (mass air flow) or a MAP sensor (mass air pressure). Some cars are even equipped with both a MAF and a MAP sensor . So what do these sensors do and what is the difference?

3.1 MAP sensor VS. MAF sensor

While both sensors accomplish the same thing, both have its own advantages.

MAP sensor:

1. Even if an intake pipe blows off or there is a vacuum hose leak the car will run the same, as the actual manifold pressure will not be any different so you will not be left stranded some where.

2. Sensor reads actual load so there is no guessing what load level the engine is seeing.

MAF sensor:

1. The amount of air mass flow entering the motor is very accurate even at low air flow levels. Because of this MAF sensors have a tendency to get better fuel efficiency.

2. Most have a built in AIT(Air Intake Temperature) sensor so you can monitor your actual air intake temperatures at the sensor.

3.2 MAP or Speed Density engine control:

In speed density based engine control, airflow is never actually measured. Instead incoming air mass is calculated based on temperature, manifold pressure, and engine speed, using a reference volumetric efficiency table. The engine’s volumetric efficiency is a constantly moving value based upon RPM, camshaft design, displacement, compression and instantaneous manifold pressure. The main volumetric efficiency or AFR table is usually shown as manifold absolute pressure versus engine speed. We have used AFR table in our Labview program.

Temperature correction becomes critical in a speed density system. Unlike the mass air systems that have meters compensated for ambient changes, actual intake air density must be calculated in a speed density strategy.

The downsides to speed density are a reduction in airflow resolution, increased time needed to calibrate, and reduced flexibility.

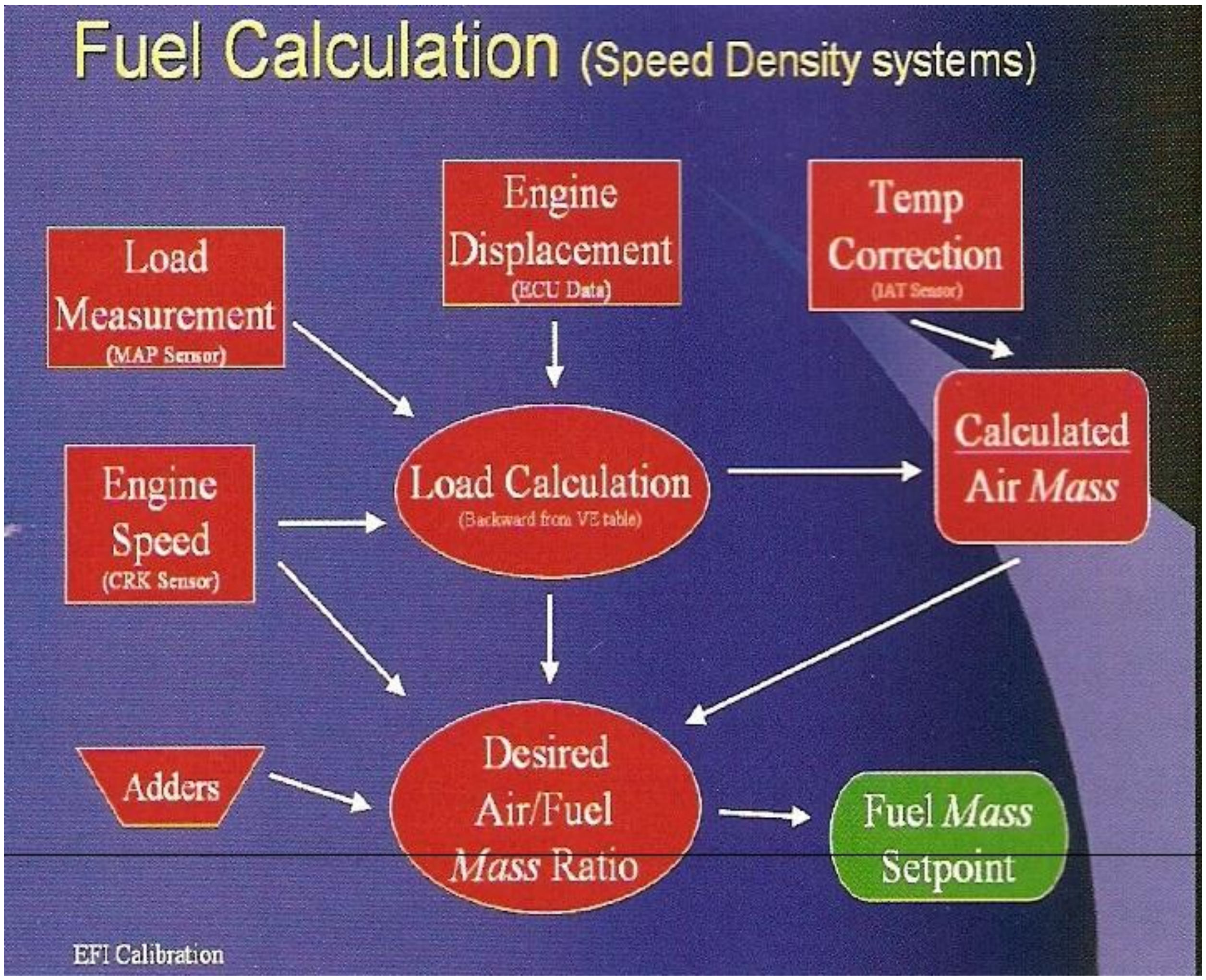

Fuel Calculation (Speed Density systems)
Load Measurement (MAP Sensor)
Engine Displacement (ECU Data)
Temp Correction (IAT Sensor)
Engine Speed (CRK Sensor)
Load Calculation (Backward from VE table)
Calculated Air Mass
Adders
Desired Air/Fuel Mass Ratio
Fuel Mass Setpoint
EFI Calibration

Figure 3.1

For a speed density system, air mass must be first calculated from a reference table before proceeding to the fuel calculation.

In MAP tuning, the MAP sensor is used to calculate the load. The amount of fuel injected by the injectors depends on several factors:

- **The Ideal Gas Law** that relates the amount of air to its pressure, volume and temperature, (this is a fundamental part of the embedded code).
- **Measured values** manifold pressure, engine and intake air temperature, rpm, etc. (these are taken from the sensor measurements).
- **Tuning parameters** REQ_FUEL, volumetric efficiency, injector open time, etc. (these are input and adjusted using the Labview tuning software).

Labview tuning software uses all these factors to determine the fuel pulse width - longer pulse widths mean more fuel (richer), shorter pulse widths mean less fuel (leaner).

The Ideal Gas Law:

The ideal gas law states that:

PV = nRT……………………………………………………………..(3.1)

Where:

P = pressure,
V = volume,
n = number of moles (which is related to the mass of the gas, i.e. 1 mol = 6.023×10^{23} molecules of the gas, and n = mass (in grams)/molar mass(MM)),
R = the ideal gas constant, and

T = the absolute temperature.

In order to know how much fuel to inject, we need to know how much air is going into the engine so the chemically correct mixture (called "stoichiometric") can be achieved. So for a fuel injected engine, we use sensors to determine the pressure in the intake manifold and the air temperature. However, the temperature in this equation is "absolute temperature" measured in Kelvins which is equal to degrees Celsius + 273°.

The volumetric efficiency (VE) is a percentage that tells us the pressure inside the cylinder versus the pressure in the manifold. We know the volume (V) from the displacement of the engine. Thus we can calculate the mass of air (M) in the cylinder (proportional to n) from

n = PV/RT
=> M = n x MM = PV/RT x MM……………………………………………………………(3.2)
= (VE * MAP * CYL_DISP) / (R * (IAT-32) * $^5/_9$ + 273)) x MM_{air}

Since:

P = VE * MAP (i.e. the pressure in the cylinder in kPa),
V = CYL_DISP = the displacement of one cylinder (in liters),
R = 8.3143510 J/mol K,
and **T** = (IAT-32)* $^5/_9$ + 273 to convert IAT from °Fahrenheit to Kelvin.

Since we now know the amount of air in a cylinder from the MAP and IAT values and the 'tuned' value for VE, we need to know the amount of fuel to inject. We specify this with a parameter called REQ_FUEL.

REQ_FUEL:

REQ_FUEL (short for "required fuel") is the part of the computation that tells Labview software how big your injectors are, and what your cylinder displacement (CYL_DISP) is. It is the length of time in milliseconds [ms] that injectors should open to give the stoichiometric amount of fuel (14.7 Air/Fuel Ratio for gasoline) at 100% VE, a manifold absolute pressure (MAP) of 100kPa, and an air temperature of 70 degrees Fahrenheit for a complete stroke cycle.

The air/fuel ratio (AFR) is the mass of air compared to the mass of fuel entering the engine, so for a 14.7:1 AFR we have 14.7 times as much air (by weight) as fuel.

A **stoichiometric mixture** is chemically correct for complete burning with no extra fuel **OR** air left over. For gasoline, a 14.7:1 AFR is considered the correct amount for burning with no leftover air or fuel.

Req_Fuel is calculated from the equation:

$$REQ\_FUEL*10 = {}^{36{,}000{,}000 * CID * AIRDEN(100kPA,\ 70°F)}/_{(NCYL*AFR*INJFLOW)}\ \ 1/DIVIDE\_PULSE$$

Where:

36,000,000 is the number of tenths of a millisecond in an hour, used to get the pounds per 1/10 milllisecond from the pounds/hours rating of the injectors.
REQ_FUEL = Computed injector open time in tenths of millisecond.
CID = Cubic Inch Displacement.
AIRDEN = Air density (pounds per cubic inch) at MAP pressure of 100 Kpa, Air Temperature of 70
NCYL = Number of Cylinders
INJFLOW = Injector Flow Rate in pounds per hour.
DIVIDE_PULSE = injection divide number for number of injections per engine cycle.

The AIRDEN function (used above) is defined by:

$$AIRDEN(MAP,\ temp) = {}^{0.0391568*\ (MAP*10-31.0)}/_{((temp+459.7)\ *\ 1728)}\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(3.4)$$

Where:

MAP = Manifold Air Pressure in kPa,
Temp = Air Temperature in Degrees F,
459.7 is used to convert from Fahrenheit to absolute temperature,
1728 is used to convert from pounds per cubic feet to pounds per cubic inch.

Hence, the REQ_FUEL value is the amount of fuel (in milliseconds) required for a MAP reading of 100 Kpa, manifold air temp of 70 degrees F, for one complete filling of one cylinder (Volumetric Efficiency = 100%), without any enrichments.

Final Pulse width Fuel Equation:

Therefore, pulse width is:

***PW = REQ_FUEL * VE * MAP * E + accel + Injector_open_time*…………(3.5)**

The "E" above is the multiplied result of all enrichments, like warm-up, after-start, barometer and air temperature correction, closed-loop, etc:

***E = gamma_Enrich = (Warmup/100) * (O2_Closed Loop/100) * (AirCorr/100) * (BaroCorr/100)*………(3.6)**

and

- *Warmup* is the warm-up enrichment value,
- *O2_Closed Loop* is the EGO adjustment based on the EGO sensor feedback in Labview
- *AirCorr* is the adjustment for air density (based on the intake air temperature), and
- *BaroCorr* is the barometric correction based on the ambient air pressure

Gamma_Enrich (E) is the scaling factor applied to the REQ_FUEL value, along with VE(RPM,MAP) and MAP. For all of the corrections, 100% means no enrichment/enleanment, since the value is normalized by 100 to get a fractional multiplier.

If you notice the pulse width equation, there are two other factors added to this - one is the acceleration enrichment, and the other is the injector open time.

Even if you set REQ_FUEL to zero you are still left with the injector open time (and accel enrichment if activated). The reason for adding in the open time is that it takes a finite amount of time to open the injector before one reaches a linear control state where injector time relates to fuel flow. The controller compensates for the open time by adding it to the applied total pulse width, otherwise the pulse would be too short.

3.3 MAF based engine control:

Rather than calculating the air flow from the AFR table and manifold pressure (MAP), the mass air flow sensor measure the amount of air directly. In some cases this can make the engine easier to tune, and better at adapting to changing engine parameters. On the other hand, the MAF sensor itself can present a significant airflow restriction to the engine (limiting power), and sometimes can have range limitations if the engine flow a lot more (or less) than the application it was designed for.

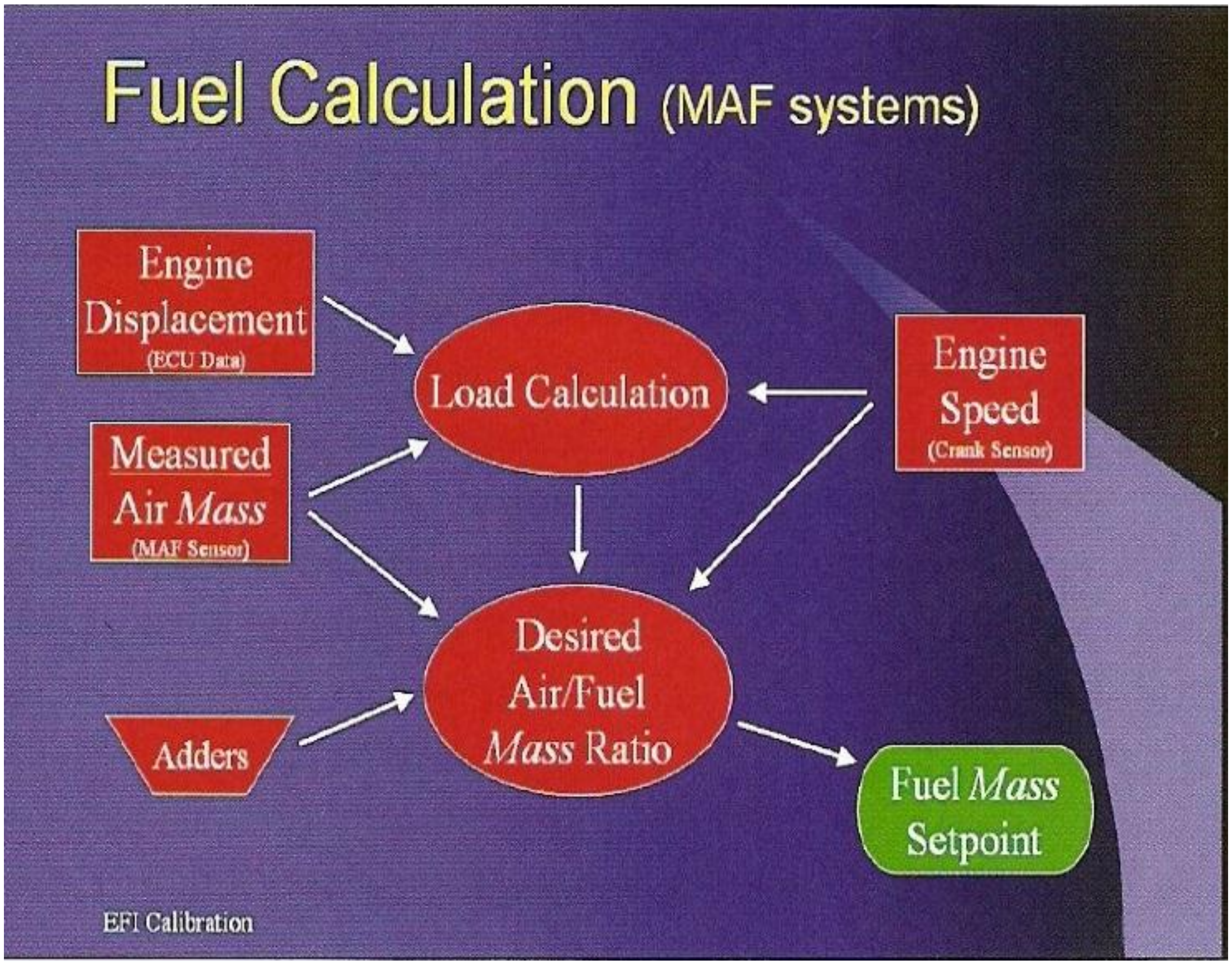

Fuel Calculation (MAF systems)
Engine Displacement (ECU Data)
Load Calculation
Engine Speed (Crank Sensor)
Measured Air Mass (MAF Sensor)
Desired Air/Fuel Mass Ratio
Adders
Fuel Mass Setpoint
EFI Calibration

Figure 3.2

For a mass air based system, the amount of fuel delivered is calculated directly from the MAF sensor input and desired ratio.

The primary benefit to the mass air system is that any changes in actual air system that fall within the MAF sensor's range and resolution can be instantly accommodated by the engine control unit. This means that a change to a larger camshaft or higher flowing intake manifold simply show up to the ECU as slightly higher fuel delivery. Mass air systems tend to be very forgiving of relatively drastic modifications in the name of horsepower. If it is possible to cleanly install a MAF sensor, this is the most desirable method of engine control due to its flexibility and accuracy.

To begin the actual calibration process for an engine, one of the most important steps is to recognize the instantaneous airflow. This airflow can then be processed to determine the necessary fuel delivery to maintain smooth engine operation.

Mass air flow systems rely heavily upon input from the sensor. These systems take the output of the MAF sensor as a direct representation of current engine airflow. This approach makes for very simple and straightforward calculation of engine load and fuel requirements. In this case engine load can be instantly shown as:

$$\boldsymbol{Engine\ load} = \frac{\boldsymbol{MAF}}{(\boldsymbol{Displacement} \times \boldsymbol{density} \times \boldsymbol{speed})}$$

Once the sensor calculates the incoming air, the amount of fuel required can be calculated using following formula:

$$\boldsymbol{Fuel\,rate\ required}\left(\frac{\boldsymbol{kg}}{\boldsymbol{s}}\right) = \frac{\boldsymbol{Mass\ air\ flow\ rate}\ (\frac{\boldsymbol{kg}}{\boldsymbol{s}})}{\boldsymbol{Air\ Fuel\,ratio}\ (\boldsymbol{from\ table})} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(3.7)}$$

The pulse width duty cycle can be calculated using following equation:

$$\boldsymbol{Duty\ cycle}\ (\%) = \frac{\boldsymbol{Fuel\,rate\ required}\ (\frac{\boldsymbol{kg}}{\boldsymbol{s}})}{\boldsymbol{Injector\ flow\ rate}\ (\frac{\boldsymbol{kg}}{\boldsymbol{s}})} \times \mathbf{100} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(3.8)}$$

The pulse width in ms can be calculated from the following equation.

$$\boldsymbol{Pulse\,width}\ = \frac{\boldsymbol{duty\ cycle}(\%)}{\boldsymbol{frequency}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(3.9)}$$

$$\boldsymbol{frequency} = \frac{\boldsymbol{RPM}}{\mathbf{120}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(3.10)}$$

Combining equation 1,2,3 and 4

The final pulse width equation can be written as

$$\boldsymbol{Pulse\,width}(\boldsymbol{ms}) = \frac{\mathbf{120} \times \boldsymbol{MFR} \times \mathbf{1000}}{\boldsymbol{RPM} \times \boldsymbol{INJECTOR\ FLOW\ RATE}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(3.11)}$$

CHAPTER 4

Introduction to Labview control Model:

Our project team has developed an ECU which can be used to control wide range of spark ignition engine. Engine control module is developed on National Instruments LABVIEW software. The main components of this ECU are:

1. Sensors
2. Data Acquisition Card
3. Computer having LABVIEW software installed

4.1 Working:

LABVIEW Control module takes input from different sensors via DAQ card to determine engine load, speed and temperature. Then use these values to determine pulse width of injector and ignition timing using fuel and ignition tables stored in the module. Our ECU module is capable of measuring real time rpm, load and temperature of engine and then calculates the optimum pulse width and spark advance required at these values and sends command to injectors and spark plugs accordingly.

4.2 Block Diagram:

Our Labview Engine control model works in following manner.

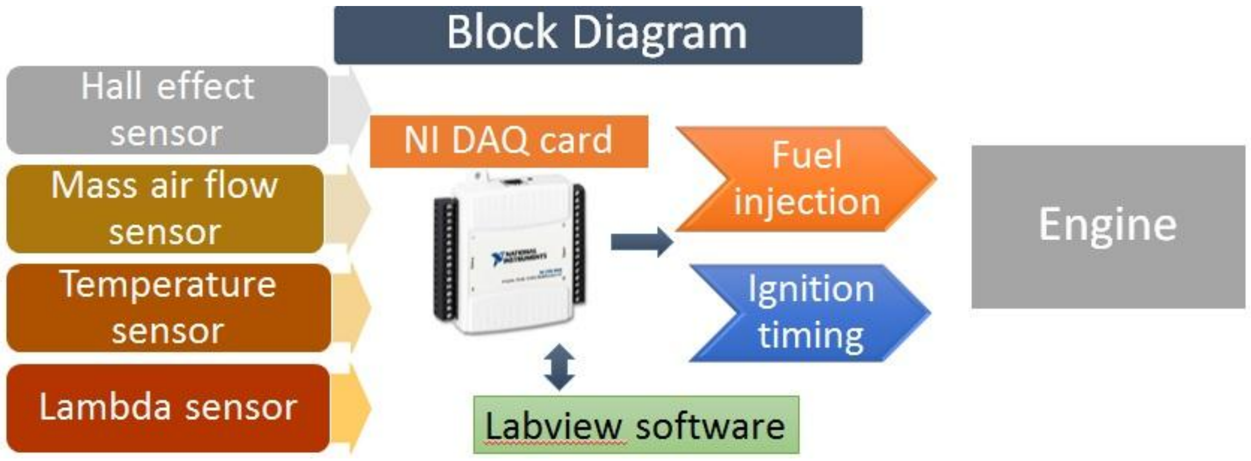


Figure 4.1

Our control model works in three stages.

(1) NI DAQ card takes input from different sensors such as crank sensor, mass air flow sensor, temperature sensor and communicate with the Labview programme.
(2) Labview programme interpret these values from sensors and based on algorithm calculates different parameters.
(3) Labview gives instructions to different actuators through NI DAQ.

The software that we have used is National Instruments labVIEW 2013. The interface is quite user friendly. The detail of the user interface is given below:

4.3 User Interface:

User interface consists of three tabs:

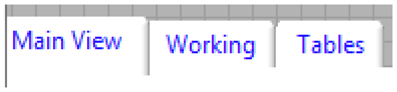
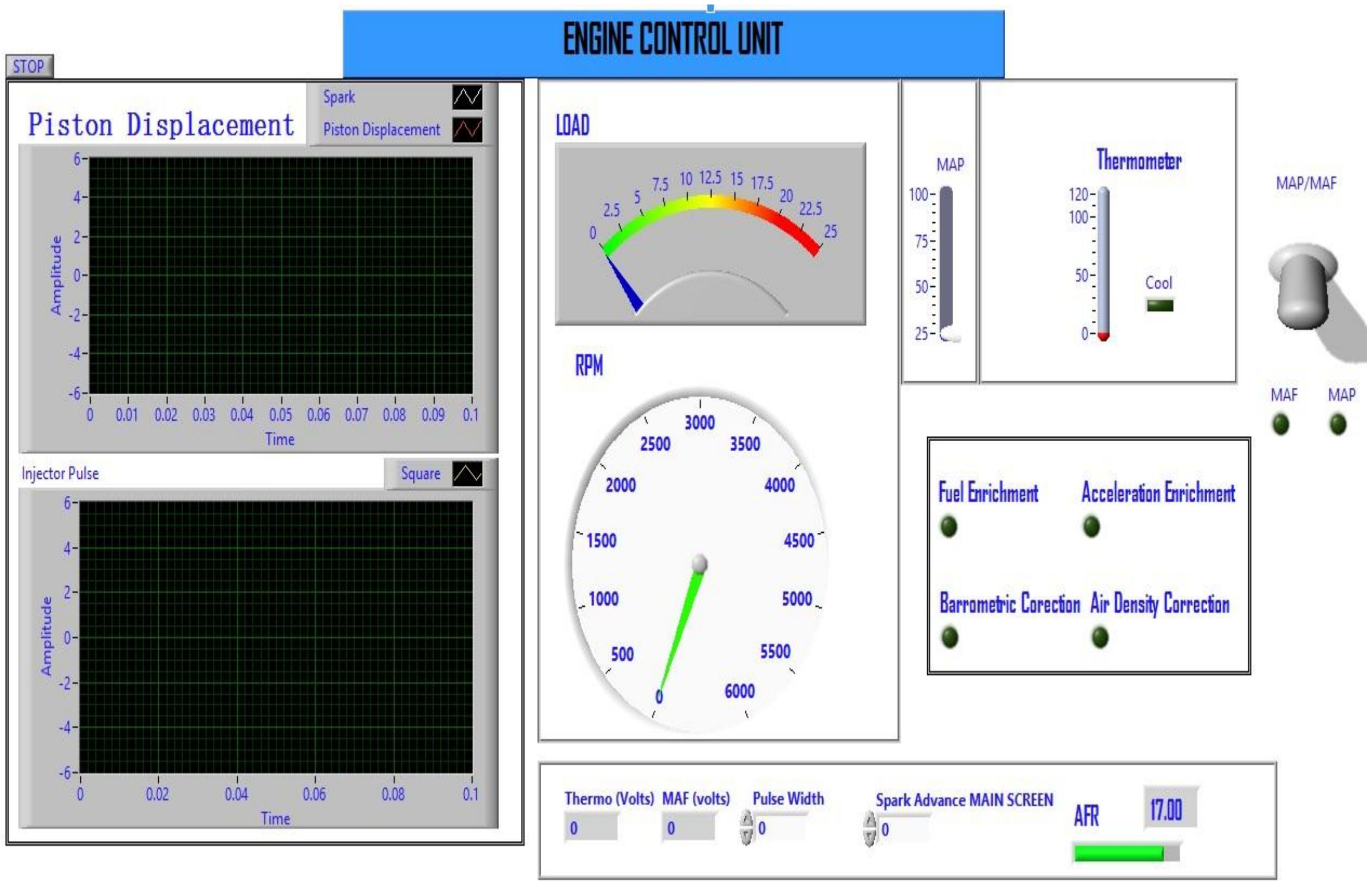
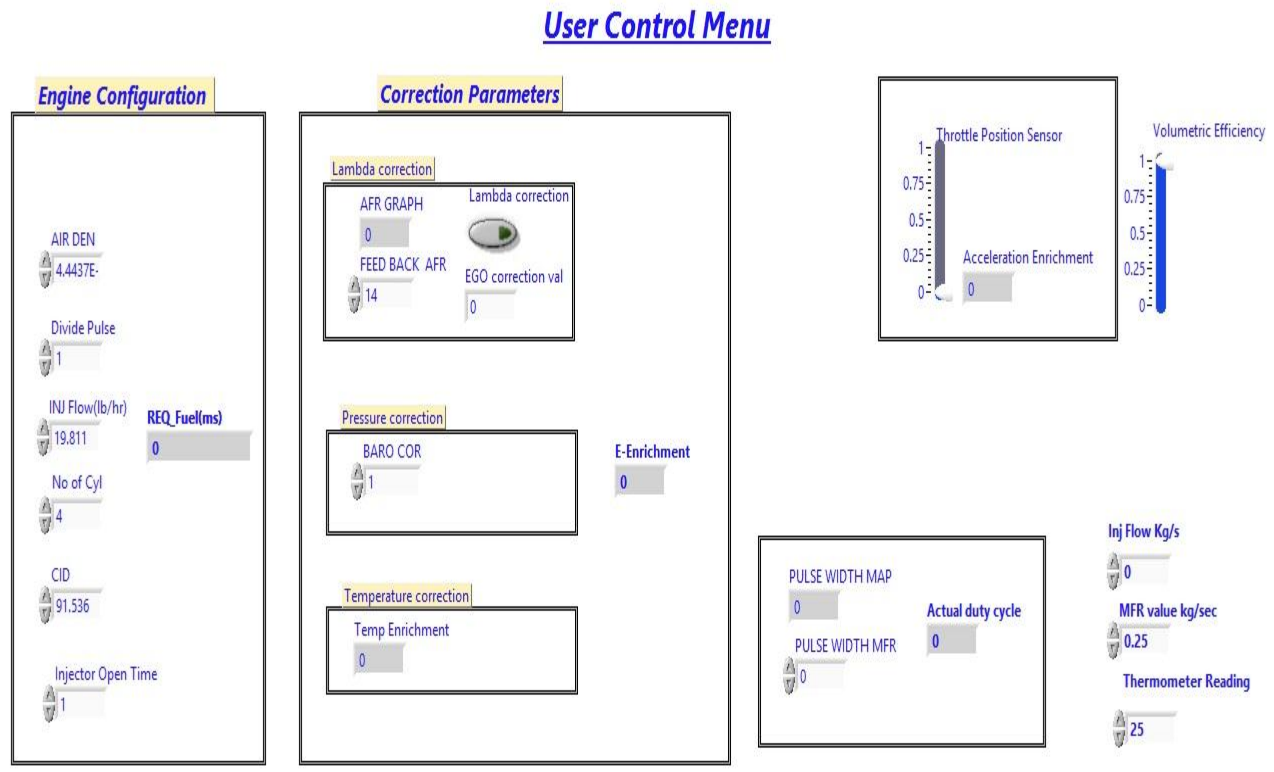


Figure 4.2

Main View shows the real time working of the software.



Working contains the real time equation variables and constants.



User Control Menu

Engine Configuration

AIR DEN 4.4437E-

Divide Pulse 1

INJ Flow(lb/hr) 19.811

REQ_Fuel(ms) 0

No of Cyl 4

CID 91.536

Injector Open Time 1

Correction Parameters

Lambda correction

AFR GRAPH 0

Lambda correction

FEED BACK AFR 14

EGO correction val 0

Pressure correction

BARO COR 1

E-Enrichment 0

Temperature correction

Temp Enrichment 0

Throttle Position Sensor

Acceleration Enrichment 0

Volumetric Efficiency

PULSE WIDTH MAP 0

Actual duty cycle 0

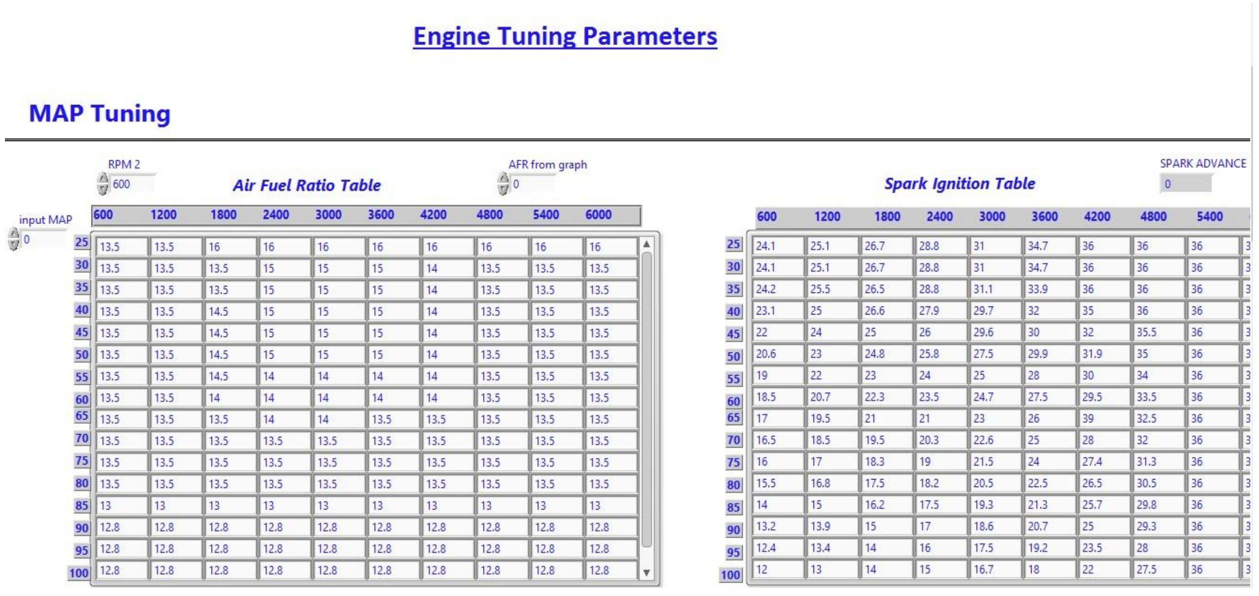
PULSE WIDTH MFR 0

Inj Flow Kg/s 0

MFR value kg/sec 0.25

Thermometer Reading 25

Tables consists of the Air fuel ratio and spark advance tables.



Engine Tuning Parameters

MAP Tuning

RPM 2 600

Air Fuel Ratio Table

AFR from graph 0

input MAP 0

	600	1200	1800	2400	3000	3600	4200	4800	5400	6000
25	13.5	13.5	16	16	16	16	16	16	16	16
30	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
35	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
40	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
45	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
50	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
55	13.5	13.5	14.5	14	14	14	14	13.5	13.5	13.5
60	13.5	13.5	14	14	14	14	14	13.5	13.5	13.5
65	13.5	13.5	13.5	14	14	13.5	13.5	13.5	13.5	13.5
70	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
75	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
80	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
85	13	13	13	13	13	13	13	13	13	13
90	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8
95	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8
100	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8

Spark Ignition Table

SPARK ADVANCE 0

	600	1200	1800	2400	3000	3600	4200	4800	5400
25	24.1	25.1	26.7	28.8	31	34.7	36	36	36
30	24.1	25.1	26.7	28.8	31	34.7	36	36	36
35	24.2	25.5	26.5	28.8	31.1	33.9	36	36	36
40	23.1	25	26.6	27.9	29.7	32	35	36	36
45	22	24	25	26	29.6	30	32	35.5	36
50	20.6	23	24.8	25.8	27.5	29.9	31.9	35	36
55	19	22	23	24	25	28	30	34	36
60	18.5	20.7	22.3	23.5	24.7	27.5	29.5	33.5	36
65	17	19.5	21	21	23	26	39	32.5	36
70	16.5	18.5	19.5	20.3	22.6	25	28	32	36
75	16	17	18.3	19	21.5	24	27.4	31.3	36
80	15.5	16.8	17.5	18.2	20.5	22.5	26.5	30.5	36
85	14	15	16.2	17.5	19.3	21.3	25.7	29.8	36
90	13.2	13.9	15	17	18.6	20.7	25	29.3	36
95	12.4	13.4	14	16	17.5	19.2	23.5	28	36
100	12	13	14	15	16.7	18	22	27.5	36

4.3.1 Main View:

The indicators/graphs of **Main View** tab are described below:

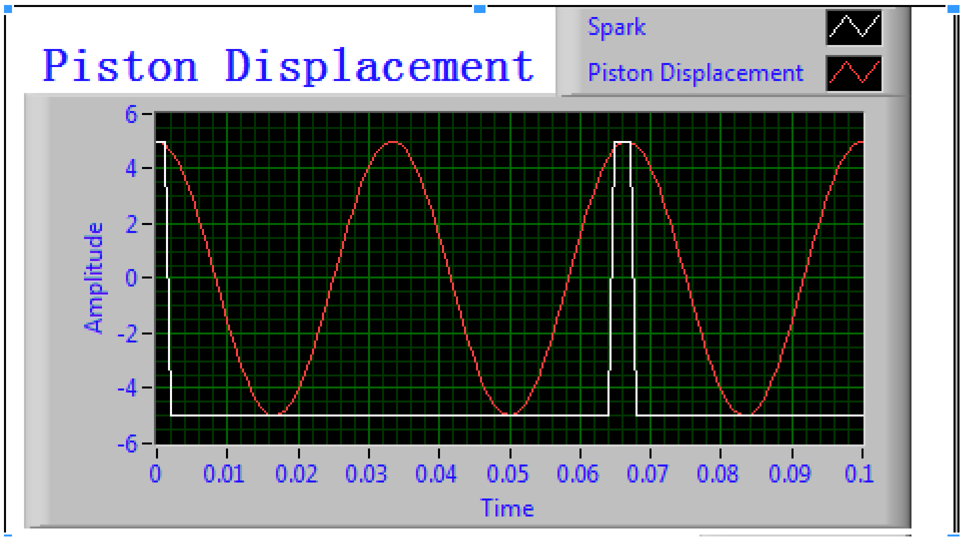


Figure 4.3

This graph shows the relation of piston displacesment and spark advance.

The y axis is amplitude and the amplitude of both ‘spark’ and ‘piston displacement’ has been set to (+, -) 5 arbitrarily. The x axis is time in seconds.

The red line indicates piston displacement as a sine wave. The maximum point (+5) is the TDC (Top Dead Centre) and the minimum point (-5) is BDC (bottom Dead Centre).

The **white** line indicates the spark. The white line changes its phase with the read line according to spark advance.

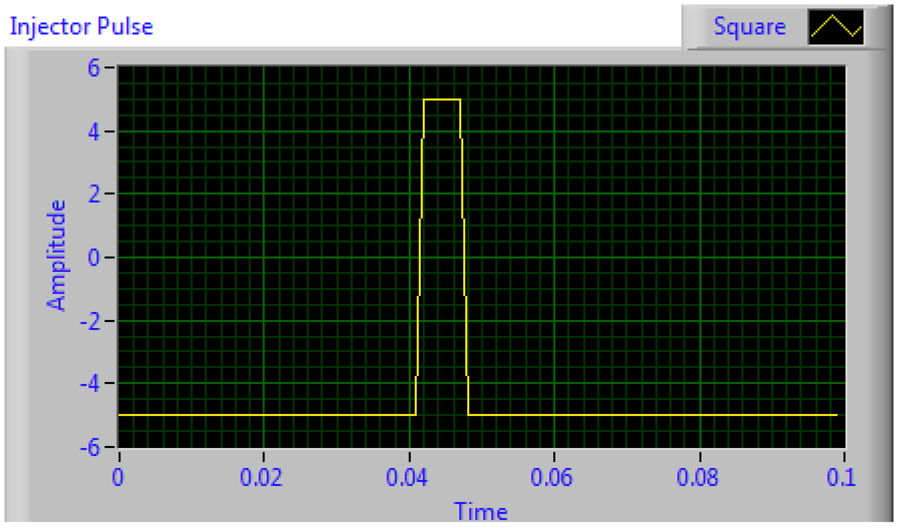


Figure 4.4

This graph shows the injector pulse width with time. Y axis is amplitude and x axis is time in seconds. The amplitude is set to (+, -) 5 arbitrarily. The graph also gives the graphical representation of duty cycle.

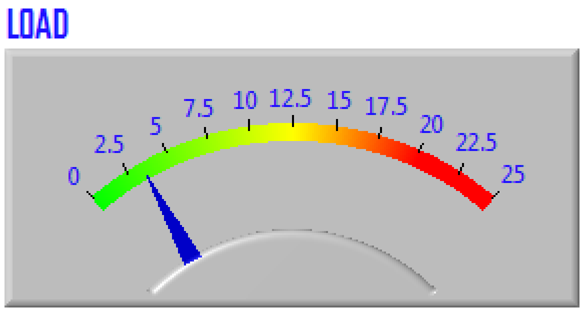


Figure 4.5

This meter indicates the LOAD calculated from the Mass Air Flow sensor. The range is from 0g/rev to 25g/rev.

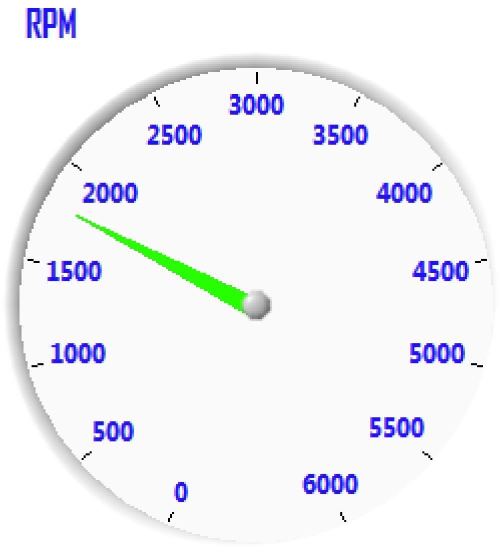


Figure 4.6

This meter represents the RPM (rev/min) of the engine or crankshaft.

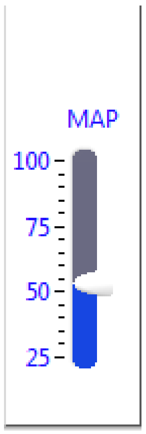


Figure 4.7

This bar is used to change the Manifold Absolute Pressure value. It ranged from 25kPa to 100kPA.

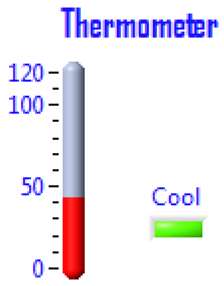


Figure 4.8

This indicates the engine coolant temperature. The LED is lit when the engine coolant temperature is below a certain limit.

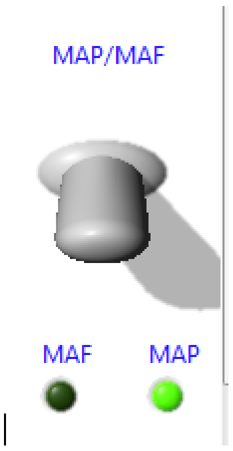


Figure 4.9

This switch controls the method of pulse width and spark advance calculation. It is used to switch from MAF to MAP and vice versa. The LEDs indicate which method is currently switched on.

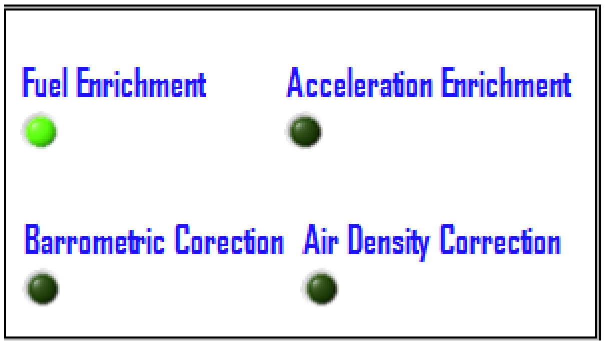


Figure 4.10

The LEDs indicate which logic is currently being used.

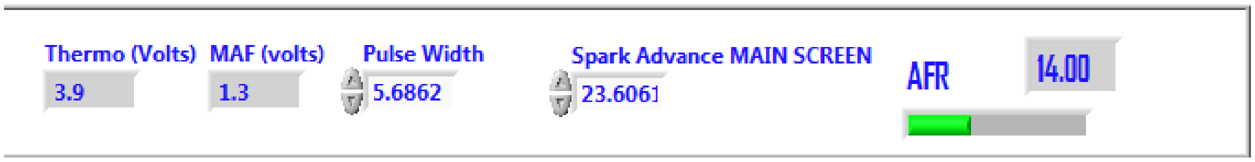


Figure 4.11

Thermo (Volts) indicates the voltage input from temperature circuit.

MAF (volts) indicates the voltages input from MAF sensor.

Pulse Width is the value of calculated final Pulse width in milliseconds.

Spark advance MAIN SCREEN is the value of calculated final spark advance in degrees.

AFR indicates the feedback Air to Fuel Ratio.

4.3.2 Working Tab:

The indicators/graphs of **Working** tab are described below:

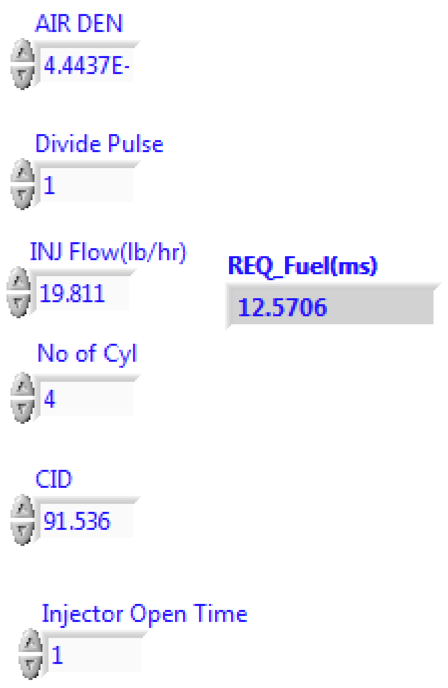

AIR DEN 4.4437E-
Divide Pulse 1
INJ Flow(lb/hr) 19.811
REQ_Fuel(ms) 12.5706
No of Cyl 4
CID 91.536
Injector Open Time 1

Figure 4.12

AIR DEN is the value of air density which the user enters. The units should be pounds per inch.

Divide Pulse is the value which the user enters. It is the injection divide number for number of injections per engine cycle.

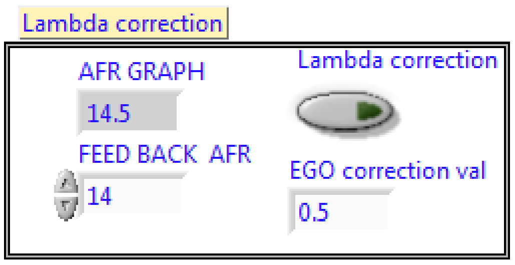
INJ Flow(lb/hr) is the value of injector flow rate of the injectors that are being used. Units should be **lb/hr** or Pounds per hour.

No of Cyl is the number of cylinders of the engine.

CID is the volume of the engine in cubic inch displacement.

Injector Open Time is the time in milliseconds that must be added to adjust for the time taken to open and close the injector.

REQ_fuel(ms) is the constant calculated from the parameters entered. It is the length of time in milliseconds [ms] that give the stoichiometric amount of fuel (14.7 Air/Fuel Ratio for gasoline) at 100% VE, a manifold absolute pressure (MAP) of 100kPa, and an air temperature of 70 degrees Fahrenheit for a complete stroke cycle.


Lambda correction

AFR GRAPH 14.5

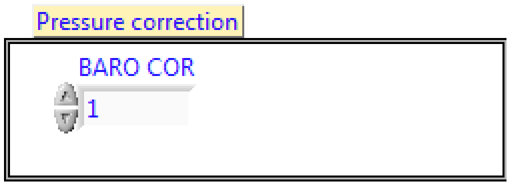
FEED BACK AFR 14

Lambda correction

EGO correction val 0.5

Figure 4.13

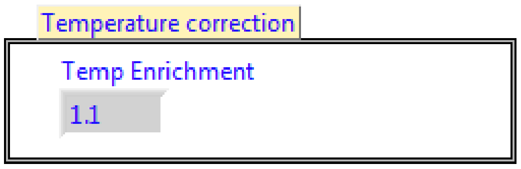
This is air fuel ratio correction. **AFR GRAPH** is the AFR calculated from the stored tables. **FEED BACK AFR** is the AFR that the user controls to see the effect of varying AFR. **Lambda correction** is the button which pressed equalizes AFR GRAPH and FEED BACK AFR, indicating that the feed back AFR is the same as the required AFR. **EGO correction value** is the correction factor to get the desired AFR in the output.


Pressure correction

BARO COR 1

Figure 4.14

Pressure correction is the correction factor corresponding to the change in ambient pressure.


Temperature correction

Temp Enrichment 1.1

Figure 4.15

Temperature correction is the correction factor corresponding to the cooolant temperature of the engine.

E-Enrichment 0.55

Figure 4.16

E-Enrichment is the final enrichment correction value calculated by EGO correction value, Temp Enrichment and BARO COR.

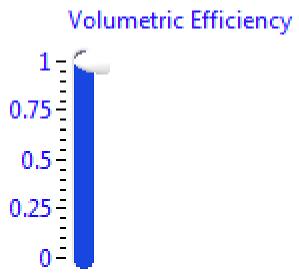


Figure 4.17

This bar can be controlled to see the variations resulting from change in volumetric efficiency of the engine.

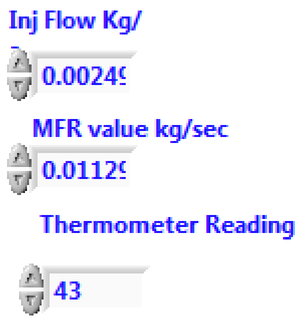


Figure 4.18

Inj Flow Kg/s is the injector flow converted to kg/s from input injector flow value in lb/hr. This value is used for calculating the load.

MFR value kg/sec is the mass flow rate in the manifold.

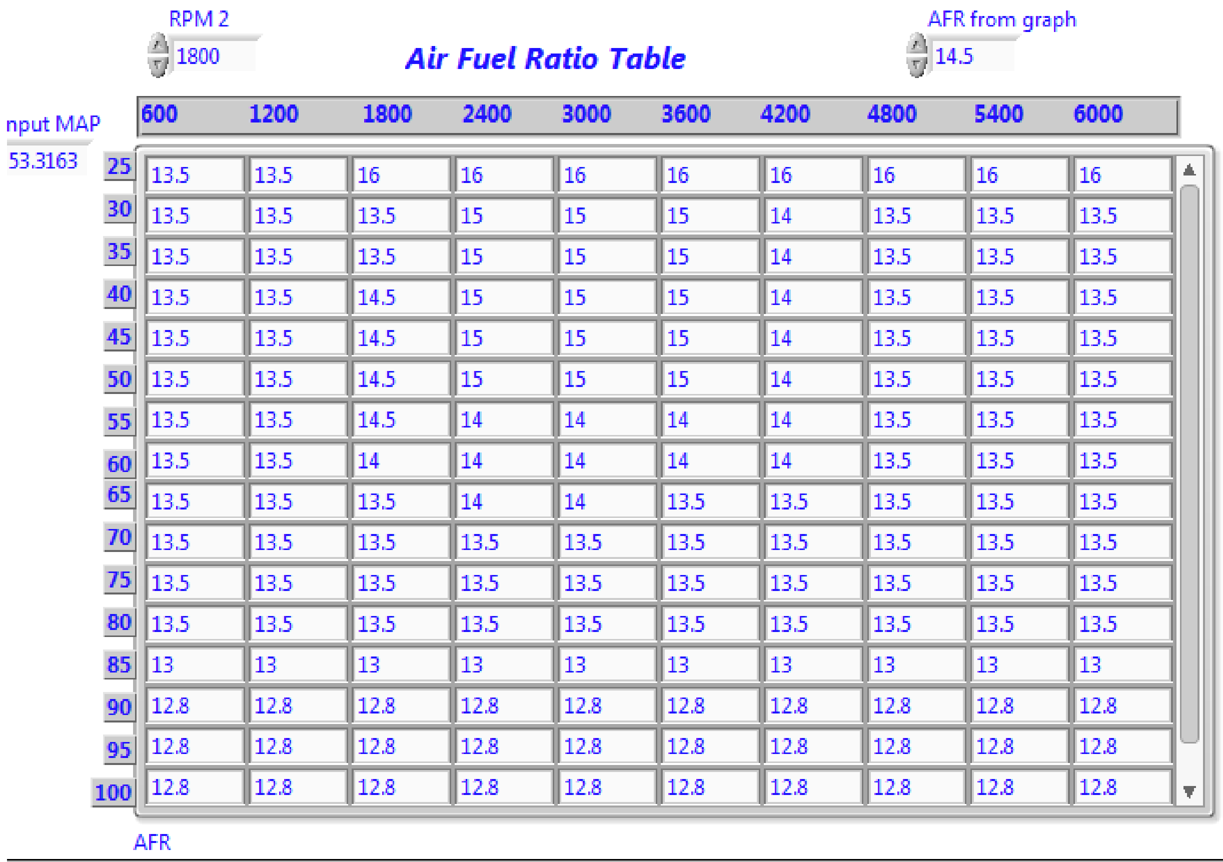
Thermometer reading is the engine temperature.

4.3.3 Tables

The indicators/graphs of **tables** tab are described below:

There are two methods to calculate AFR and Spark advance; **MAF** and **MAP**.

MAP Tuning



RPM 2
1800

Air Fuel Ratio Table

AFR from graph
14.5

nput MAP
53.3163

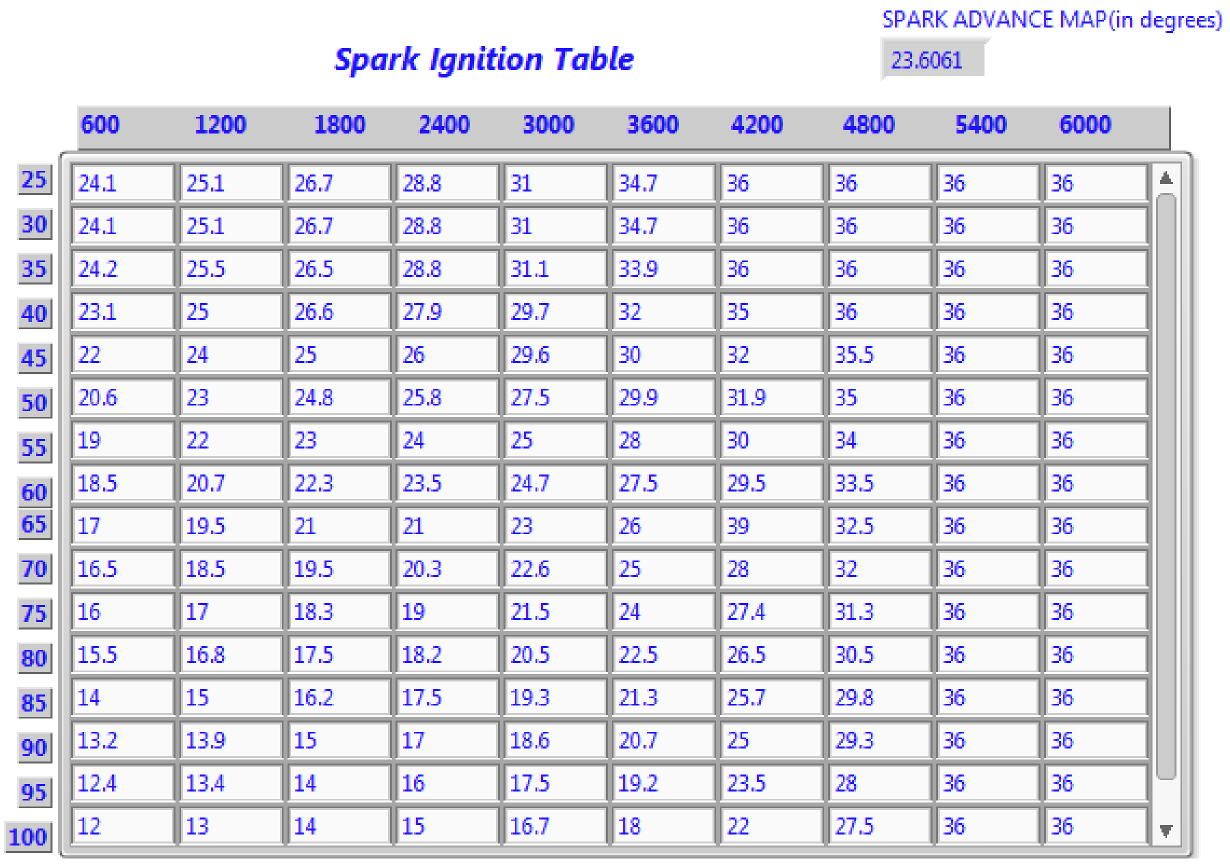
	600	1200	1800	2400	3000	3600	4200	4800	5400	6000
25	13.5	13.5	16	16	16	16	16	16	16	16
30	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
35	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
40	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
45	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
50	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
55	13.5	13.5	14.5	14	14	14	14	13.5	13.5	13.5
60	13.5	13.5	14	14	14	14	14	13.5	13.5	13.5
65	13.5	13.5	13.5	14	14	13.5	13.5	13.5	13.5	13.5
70	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
75	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
80	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
85	13	13	13	13	13	13	13	13	13	13
90	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8
95	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8
100	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8	12.8

AFR

Figure 4.19

Y axis is MAP (in kPa) and x axis is RPM (in rev/min). The graph values can be changed to obtain desired performance. It is used for tuning the engine.

AFR from graph is the final value obtained by interpolation by using the **RPM 2** and **input MAP** values.



Spark Ignition Table

SPARK ADVANCE MAP(in degrees)

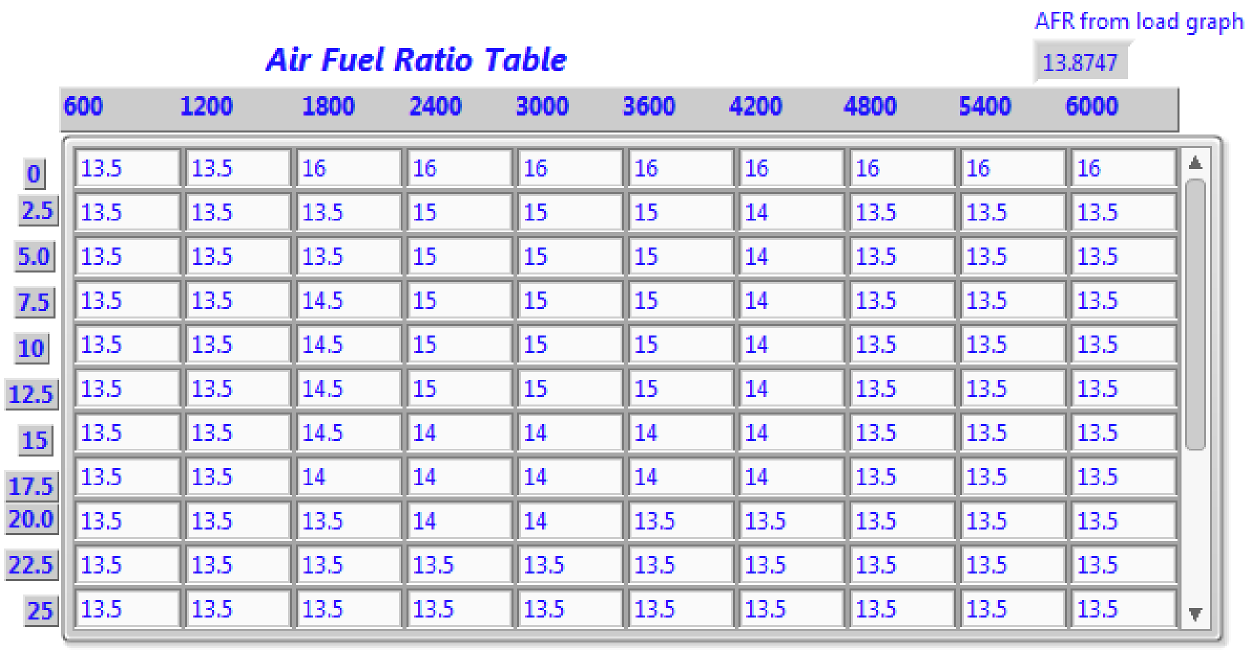
23.6061

	600	1200	1800	2400	3000	3600	4200	4800	5400	6000
25	24.1	25.1	26.7	28.8	31	34.7	36	36	36	36
30	24.1	25.1	26.7	28.8	31	34.7	36	36	36	36
35	24.2	25.5	26.5	28.8	31.1	33.9	36	36	36	36
40	23.1	25	26.6	27.9	29.7	32	35	36	36	36
45	22	24	25	26	29.6	30	32	35.5	36	36
50	20.6	23	24.8	25.8	27.5	29.9	31.9	35	36	36
55	19	22	23	24	25	28	30	34	36	36
60	18.5	20.7	22.3	23.5	24.7	27.5	29.5	33.5	36	36
65	17	19.5	21	21	23	26	39	32.5	36	36
70	16.5	18.5	19.5	20.3	22.6	25	28	32	36	36
75	16	17	18.3	19	21.5	24	27.4	31.3	36	36
80	15.5	16.8	17.5	18.2	20.5	22.5	26.5	30.5	36	36
85	14	15	16.2	17.5	19.3	21.3	25.7	29.8	36	36
90	13.2	13.9	15	17	18.6	20.7	25	29.3	36	36
95	12.4	13.4	14	16	17.5	19.2	23.5	28	36	36
100	12	13	14	15	16.7	18	22	27.5	36	36

Figure 4.20

The table is similar to AIR FUEL RATIO TABLE. But it gives the value of spark advance in degrees.

MAF Tuning



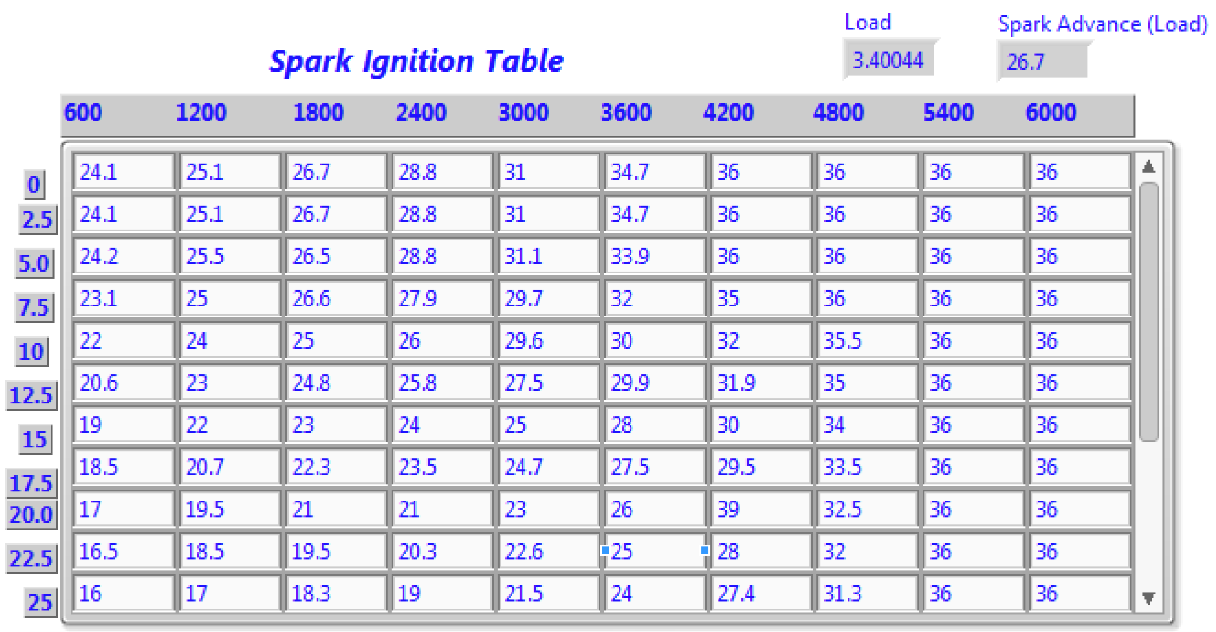
Air Fuel Ratio Table

AFR from load graph: 13.8747

	600	1200	1800	2400	3000	3600	4200	4800	5400	6000
0	13.5	13.5	16	16	16	16	16	16	16	16
2.5	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
5.0	13.5	13.5	13.5	15	15	15	14	13.5	13.5	13.5
7.5	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
10	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
12.5	13.5	13.5	14.5	15	15	15	14	13.5	13.5	13.5
15	13.5	13.5	14.5	14	14	14	14	13.5	13.5	13.5
17.5	13.5	13.5	14	14	14	14	14	13.5	13.5	13.5
20.0	13.5	13.5	13.5	14	14	13.5	13.5	13.5	13.5	13.5
22.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5
25	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5	13.5

Figure 4.21

The y axis is LOAD calculated from MAF. And the x-axis is RPM. **AFR from load** gives the value obtained from interpolation using LAOD and RPM.



Spark Ignition Table

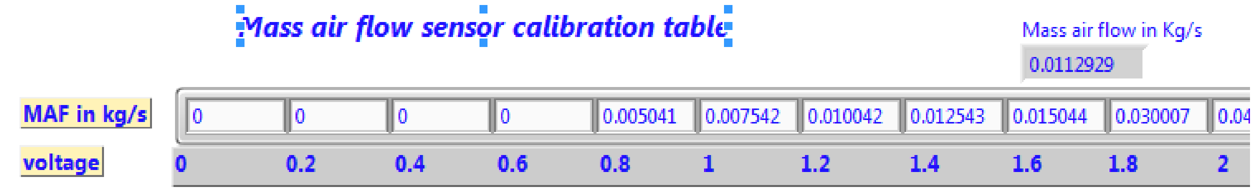
Load: 3.40044

Spark Advance (Load): 26.7

	600	1200	1800	2400	3000	3600	4200	4800	5400	6000
0	24.1	25.1	26.7	28.8	31	34.7	36	36	36	36
2.5	24.1	25.1	26.7	28.8	31	34.7	36	36	36	36
5.0	24.2	25.5	26.5	28.8	31.1	33.9	36	36	36	36
7.5	23.1	25	26.6	27.9	29.7	32	35	36	36	36
10	22	24	25	26	29.6	30	32	35.5	36	36
12.5	20.6	23	24.8	25.8	27.5	29.9	31.9	35	36	36
15	19	22	23	24	25	28	30	34	36	36
17.5	18.5	20.7	22.3	23.5	24.7	27.5	29.5	33.5	36	36
20.0	17	19.5	21	21	23	26	39	32.5	36	36
22.5	16.5	18.5	19.5	20.3	22.6	25	28	32	36	36
25	16	17	18.3	19	21.5	24	27.4	31.3	36	36

Figure 4.22

This is the spark advance table. The y axis is LOAD calculated from MAF. And the x-axis is RPM. It is used to calculate spark advance in degrees from interpolation using LAOD and RPM. The value is displayed in **Spark Advance (Load)**.



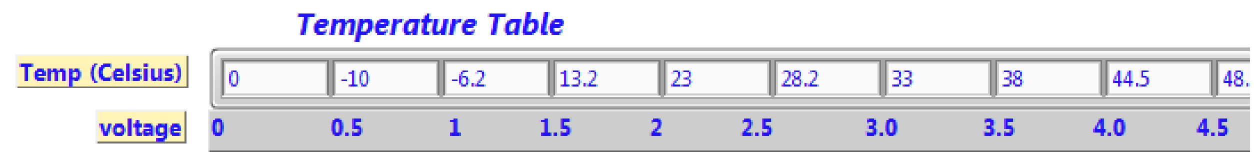
Mass air flow sensor calibration table

Mass air flow in Kg/s: 0.0112929

MAF in kg/s	0	0	0	0	0.005041	0.007542	0.010042	0.012543	0.015044	0.030007	0.04
voltage	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2

Figure 4.23

This table is used to obtain the value of MAF (in Kg/s) from the voltage acquired from the MAF sensor.



Temperature Table

Temp (Celsius)	0	-10	-6.2	13.2	23	28.2	33	38	44.5	48.
voltage	0	0.5	1	1.5	2	2.5	3.0	3.5	4.0	4.5

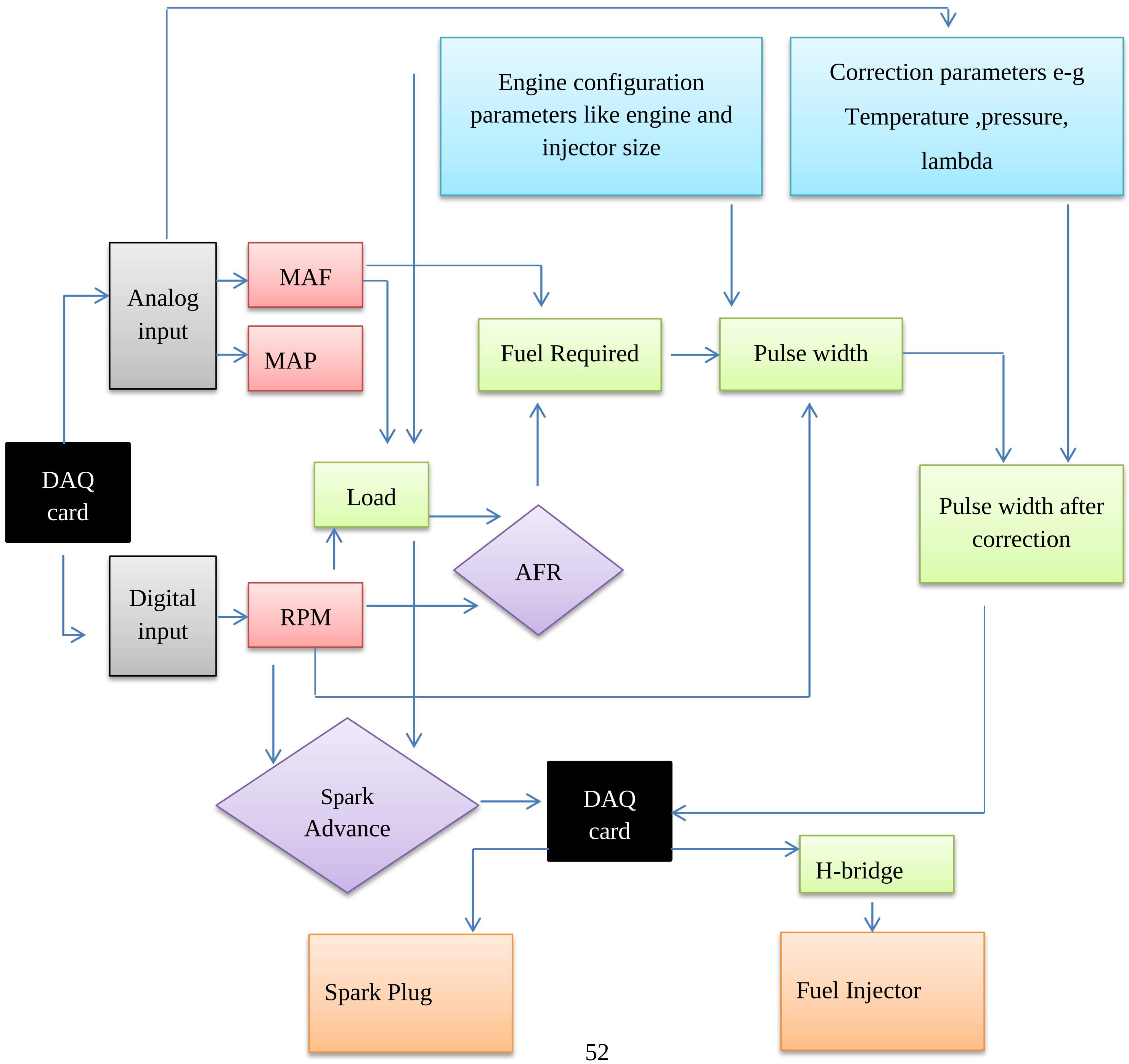
Figure 4.24

This table is used to obtain the actual value of temperature from the voltage acquired by the temperature circuit. The values are in degree Celsius.

CHAPTER 5

Labview Algorithm Explanation:

5.1 Block Diagram:



Engine configuration parameters like engine and injector size
Correction parameters e-g Temperature ,pressure, lambda
Analog input
MAF
MAP
Fuel Required
Pulse width
DAQ card
Load
AFR
Pulse width after correction
Digital input
RPM
Spark Advance
DAQ card
H-bridge
Spark Plug
Fuel Injector

5.2 Code Explanation:

A detail is presented step by step of how the individual values are calculated.

From the main equation of Pulse width (in milliseconds)

5.2.1 Req_Fuel(ms) in the *Working* tab.

Equation:

REQ_FUEL = $^{\textbf{36,000,00 * CID * AIRDENSITY}}/_{\textbf{(NCYL*AFR*INJFLOW)}}$ *** 1/DIVIDE_PULSE**

The numerators and denominators are made into a matrix by using *BUILD MATRIX* (pic below).

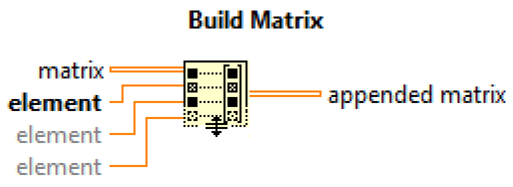


Figure 5.1

First the denominators are input into *BUILD MATRIX*.

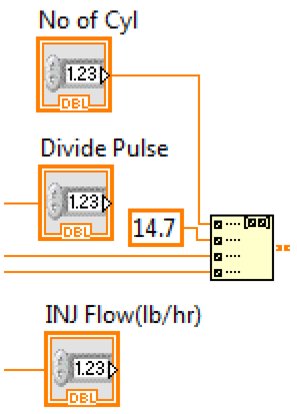


Figure 5.2

Then they are inversed so they can be multiplied by the numerators. This is done by using

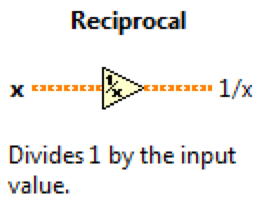
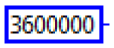
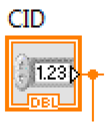


Figure 5.3

Then the numerators
AIR DEN
1.23
DBL
, 

and 

along with the reciprocal of the denominators are build into a matrix by using *BUILD MATRIX*. Then their product is calculated by using

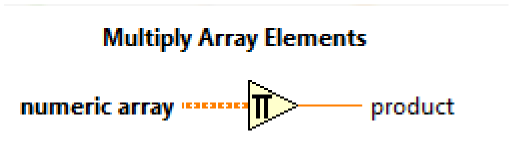
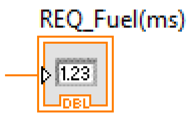


Figure 5.4

Then the result is displayed in 

.

5.2.2 E-Enrichment: in the working tab.

Equation:-

E= EGO-correction val* BARO COR* Temp Enrichment

- **BARO COR** is a constant till now. It can be made a variable if some ambient pressure sensor is used.
- **Temp Enrichment** depends upon the temperature.

The *Thermometer reading* is compared by a constant number (70 in our case). If this is true the LED labelled *Cool* is turned on and the value of *Temp Enrichment* is 1.

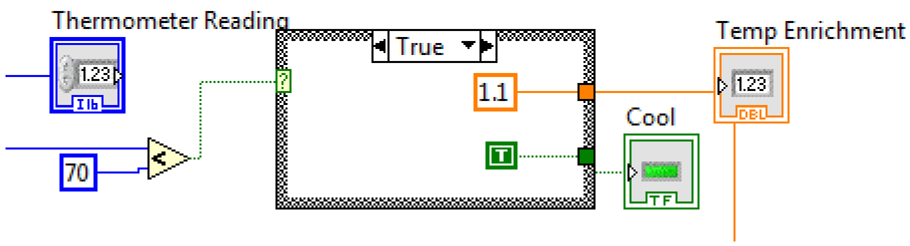


Figure 5.5

If on the other hand the value is more than OR equal to 70, the comparator gives a FALSE signal. And the LED labelled *Cool* is turned off and the value of Temp Enrichment is 0.

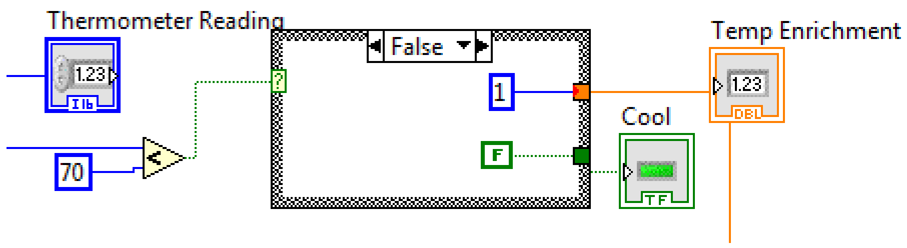
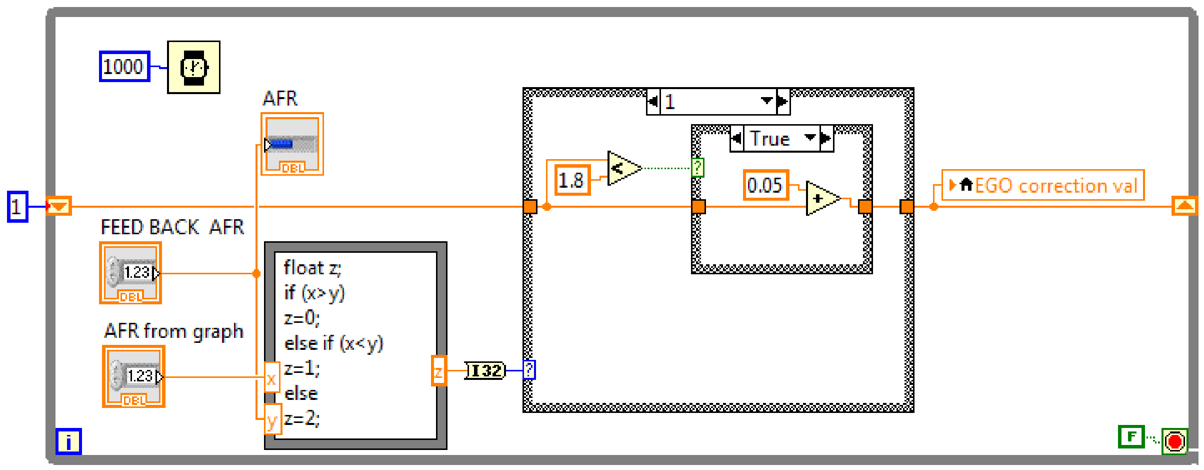


Figure 5.6

- **EGO-correction val** depends on FEED BACK AFR and AFR from graph



```
float z;
if (x>y)
z=0;
else if (x<y)
z=1;
else
z=2;
```

Figure 5.7

A new while loop is created which continues for infinity. A *Wait* is used and its time set to 1000 milliseconds i.e. 1 second. A *shift register* is created and it is initialized by 1. A *shift register* is used to store the value of one loop and transfer it into the next. Two variables: FEED BACK AFR and AFR from graph are input into the *Formula Node*. AFR from graph is stored as **x** and FEED BACK AFR is stored as **y.** Output is named **z.** In the *Formula Node* these values are compared and accordingly an output is generated.

If x>y output is 0

If x<y output is 1

Otherwise output is 2

Then this output is fed into a case structure with three cases: 0, 1 and 2.Case 2 is *default case*.

Case 0:

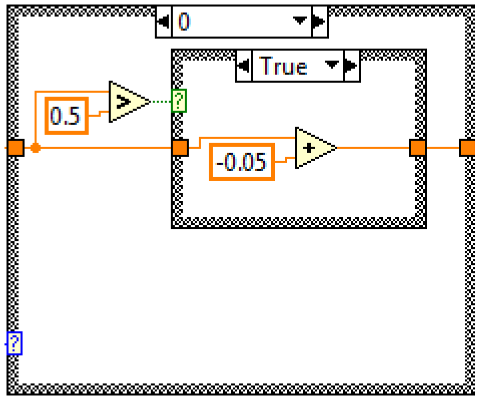


Figure 5.8

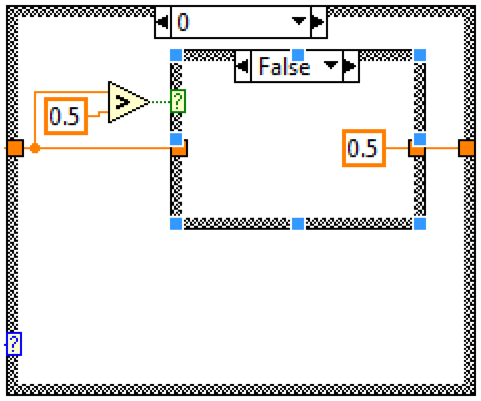


Figure 5.9

The value from shift register is compared to 1.8 using *Less than*. The output is fed into another *Case structure*. This case structure has two cases: True and False. If it is true the value is incremented by 0.05 and if it is false the value is fixed to 1.8.

Case 1:

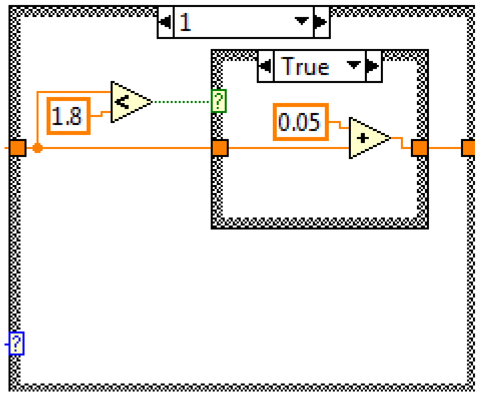


Figure 5.10

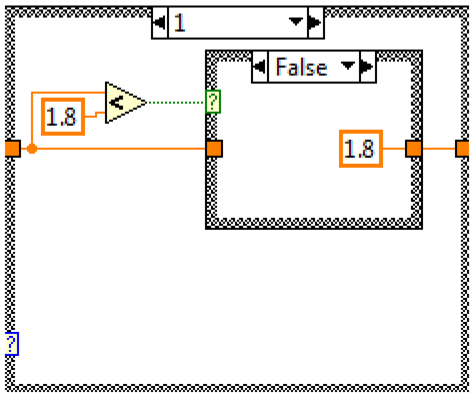


Figure 5.11

The value from shift register is compared to 0.5 using *greater than*. The output is fed into another *Case Structure*. This *Case Structure* has two cases: True and False. If it is true the value is decremented by 0.05 and if it is false the value is fixed to 0.5.

Case 2:

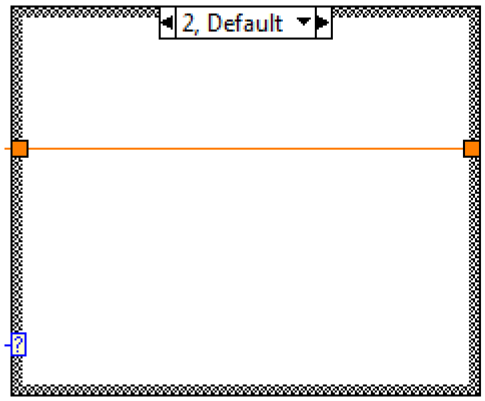
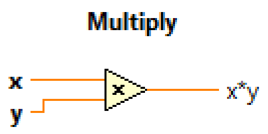


Figure 5.12

Whatever the value from the shift register is, it is outputted.

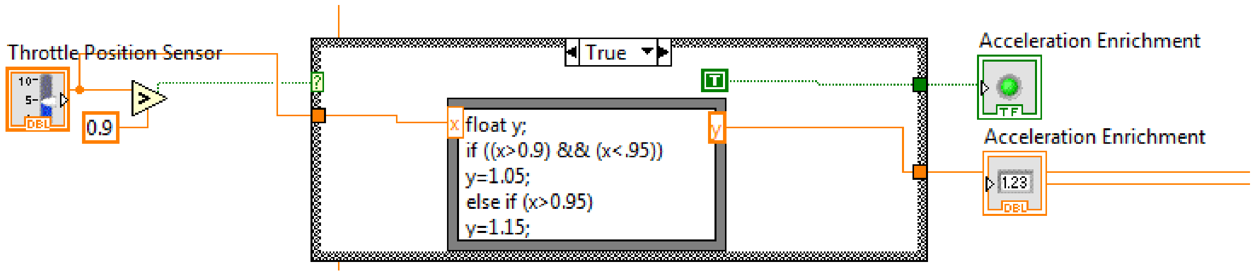
The output from this *Case Structure* is input into a *Local Variable* which indicated to *EGO correction val* in the main while loop. It is also fed into the other shift register.

Finally E-Enrichment is calculated by multiplying EGO-correction val, BARO COR andTemp Enrichment. This done using two of these in series:  .

5.2.3 Acceleration Enrichment in the *Working* tab.

The value of Acceleration Enrichment depends upon the *Throttle Position Sensor* value. The Throttle Position Sensor value is compared to 0.9 using a *Greater?* Box. The result is input into a *Case Structure*.

True:



```
float y;
if ((x>0.9) && (x<.95))
y=1.05;
else if (x>0.95)
y=1.15;
```

Figure 5.13

The value of *Throttle Position Sensor* is checked again to see where exactly does it lie. If it is more than 0.9 and less than 0.95 it gives an output of 1.05 and the *Acceleration Enrichment* LED is turned on.

False:

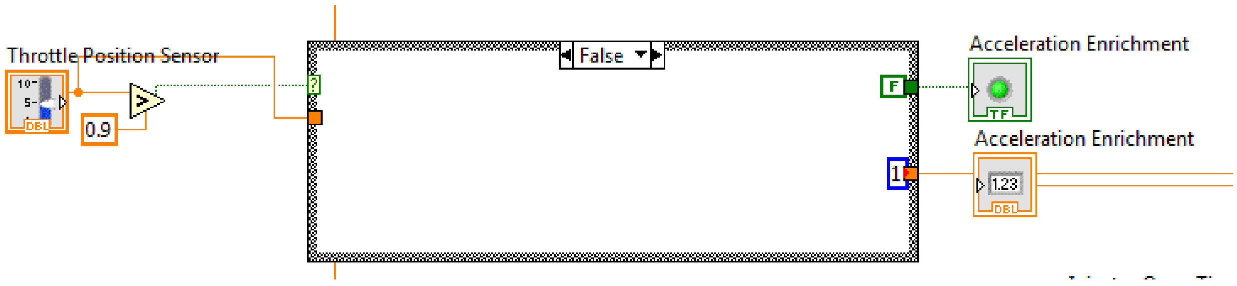


Figure 5.14

The output is 1 and the *Acceleration Enrichment* LED is turned off.

5.2.4Actual duty cycle in the *Working* tab.

This depends on RPM as well as Pulse Width. First RPM is divided by 60 to convert it into frequency and then it is multiplied by 0.5 so that it is halved. It is halved because the frequency of Injector is half that of crankshaft for a four stroke engine. Then it is multiplied by Pulse

Width. After that it is divided by 10. This is done to convert Pulse Width into milliseconds and to convert duty cycle in percentage.

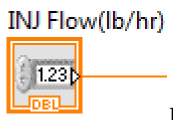
Formulae:

Duty Cycle= (Injector open time (in milliseconds)*100)/ (Time period*1000)

Time period=1/Frequency

Frequency= RPM/60

5.2.5 Inj Flow kg/s

This: 

multiplied by **0.000125997881** gives the value of *Inj Flow kg/s*.

5.2.6 MFR value kg/sec

It is obtained from the *Mass air flow sensor calibration table* using the actual input from the MAF sensor.

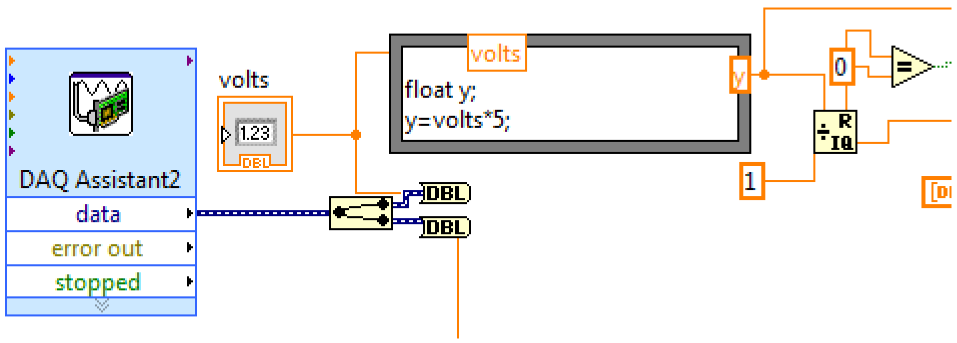


Figure 5.15

The data from the sensor is acquired using a DAQ assistant. The card we are using is USB 6009. Analog input-Voltage is acquired from the card. The Acquisition Mode is set to 'Continuous Samples'. Since we are acquiring more than one analog input (Temperature circuit and MAF sensor) which means we are using more than one channel so we need to use:

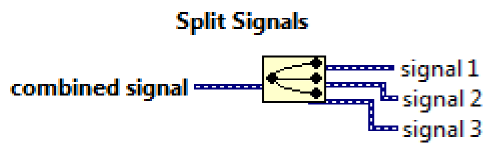


Figure 5.16

And as the data is continuous and analog we first convert it to a discrete number. We convert it into Double Precision Float. This value is displayed in *volts*. Then this value is inputted to the *Formula Node*. The input to *Formula Node* is named 'volts'. The output 'y' is the multiplication of ‘volts’ by 5. After that this output is checked whether it is a whole number or not. This is done by dividing ‘y’ by 1 and acquiring its remainder and quotient. The remainder is compared with a 0. This value is input to the case structure. The case structure has 2 cases: True and False.

True:

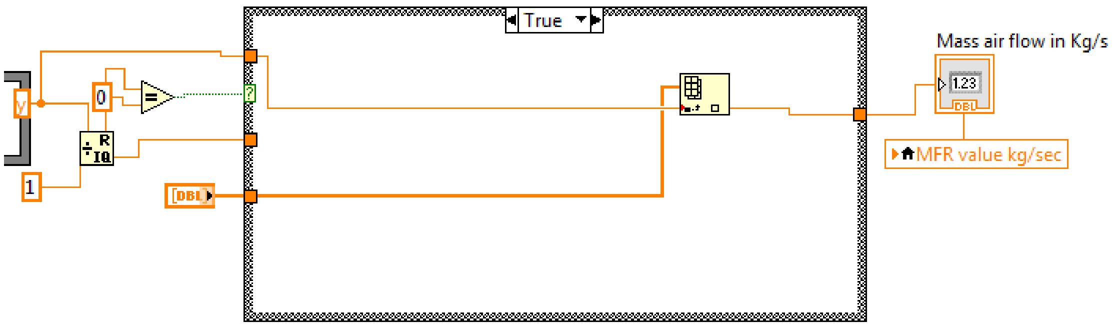


Figure 5.17

Index Array is used to get the required value from the table (*Mass air flow sensor calibration table)*. The table is 1-D array of type Double. The value ‘y’ is the other input to *Index Array*. The output is displayed in the indicator, *Mass air flow Kg/s*. The value is also forwarded to the local variable *MFR value kg/sec*.

False:

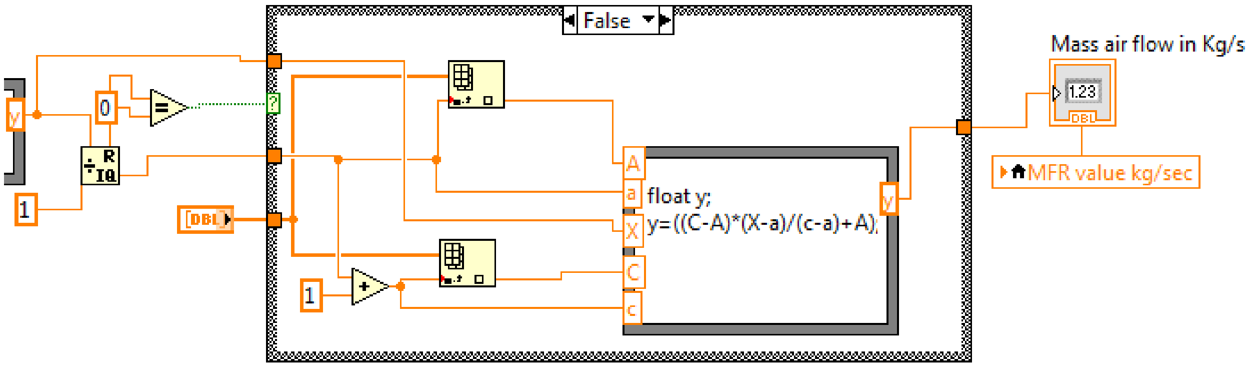


Figure 5.18

The quotient of *y/1* is input into the case. Two values are obtained from the *Mass air flow sensor calibration table.* This is done by two separate indexing. One with the input quotient and the other with (1+the input quotient). Then five variables are input to the *Formula Node*.

A: the value indexed from the table using quotient

a: the input quotient

X: ‘y’ (i.e. the output from the previous *Formula Node: y=volts*5)*

C: the value indexed from the table using quotient+1

C: the input quotient+1

Then the equation inside the *Formula Node* is used to get an output which is named ‘y’. This value is displayed in the indicator, *Mass air flow Kg/s.* The value is also forwarded to the local variable *MFR value kg/sec*.

(Note: The above method is actually **interpolation** between two points in a table to get any value between them. It will be called **1D Interpolation** for later referencing This interpolation is also done in calculating ***Thermometer Reading***, ***AFR from load graph, AFR from graph, SPARK ADVANCE MAP (degrees)*** and ***Spark Advance (load)*** **)**

5.2.7 Temperature sensor Reading

The value from temperature circuit is obtained via DAQ card. We use DAQ Assistant to add channels. The signal is split and converted to double float as described in “MFR value kg/s” section.

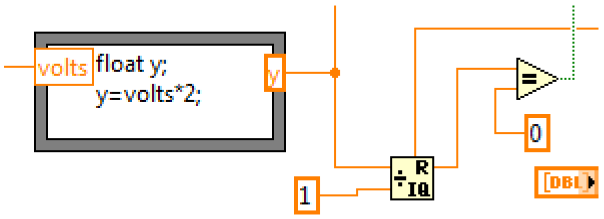


Figure 5.19

After that the data is input into the *Formula Node*. Than output is calculated by multiplying volts by 2. This output is called ‘y’. After which this output is checked whether it is a whole number or not. This is done by dividing ‘y’ by 1 and acquiring its remainder and quotient. The remainder is compared with a 0. This value is input to the case structure. The case structure has 2 cases: True and False.

True: ‘y’ is used to get the value from the *Temperature Table.*

Flase:

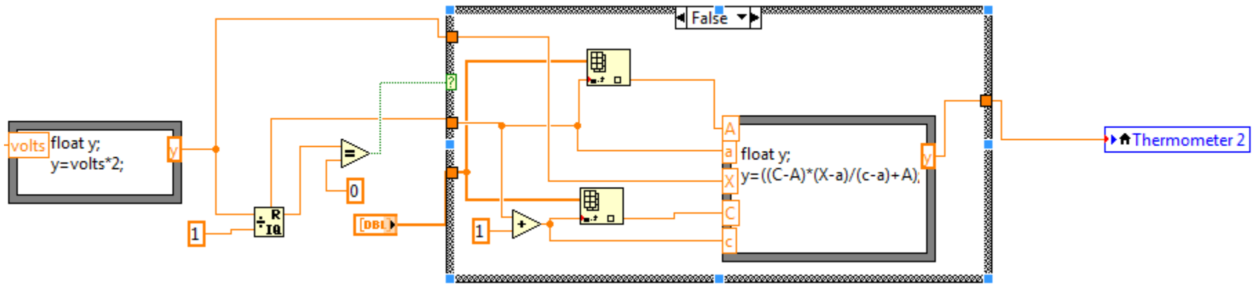


Figure 5.20

Just like in *MFR value in Kg/s*, i**nterpolation** is done to get the required value from the *Temperature Table*. The method is same as **1D Interpolation.** The output is sent to a local variable which is pointed to *Thermometer 2* which is connected to *Thermometer* in the *Main View* tab.

5.2.8 PULSE WIDTH MAP

Formula:-

PW = Req_Fuel(ms) * VE * MAP * E + Acceleration Enrichment + Injector Open Time

Req_Fuel(ms) has already been discussed.

E is actually *E-Enrichment* which has also been discussed.

Acceleration Enrichment has also been discussed.

Injector Open Time is the constant that is input by the user.

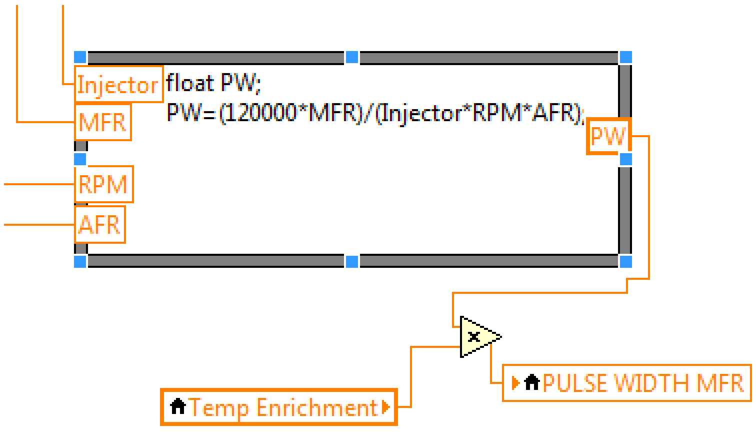
MAP is controlled by the user from the USER INTERFACE. It is a moving bar.

VE is controlled by the user from the USER INTERFACE. It is a moving bar.

First MAP is divided by 100 to make it a fraction of 100. Then this result along with *VE*, *E-Enrichment* and *Req_Fuel(ms)* are made into an array using *Build Matrix*. This matrix is inter-multiplied. The output is inserted into another *Build Matrix*. The other two inputs of this *Build Array* are *Acceleration Enrichment* and *Injector Open Time*. The appended matrix is then added to give the final output i.e. *PULSE WIDTH MAP* (which is displayed in the *working* tab).7

5.2.9 PULSE WIDTH MFR

It depends upon *Inj Flow Kg/s*, *AFR from load graph*, *RPM* and *MFR value Kg/sec*.



```
float PW;
PW=(120000*MFR)/(Injector*RPM*AFR);
```

Figure 5.21

All these variables are input into a *Formula Node*. The output, *PW*, is calculated by the formula shown. Then this value is outputted from the loop and multiplied by a correction factor/enrichment factor i.e. *Temp Enrichment* which is mentioned in early sections.

5.2.10 RPM 2

Due to the limitation of our DAQ card (USB 6009) we cannot get frequency as an input. So we have used DAQ card as a counter using DAQ Assistant.

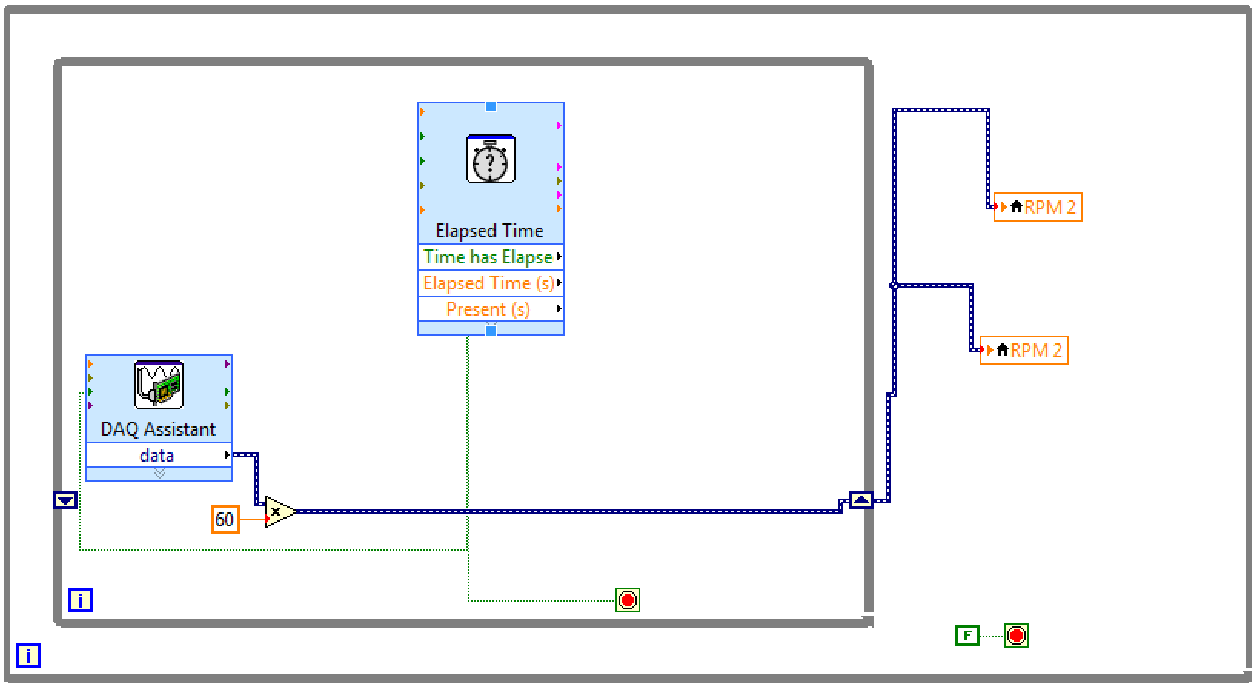


Figure 5.22

Two while loops are created. The outer while loop is made to continue till infinity. The inner while loop has shift registers to store value and transmit it outside at the termination of the inner while loop. Elapsed Time is used to control the inner loop. Elapsed time is set to 1000 milliseconds i.e. 1 second. The Elapsed Time is also used to reset DAQ Assistant. The value from the DAQ Assistant is continually multiplied by 60 and stored in a shift register. The values starts from 0 and increases until it reaches some number at the end of the time (i.e. 1 second). So the output is actually the value of RPM of the incoming signal as RPM can also be thought of as the number of counts in a second multiplied by 60. When the time elapses the shift registers lets the value out of the inner loop and it is transferred to a local variable which indicates to *RPM 2*.

5.2.11 AFR from graph in the *Tables* tab

The value of AFR from graph is obtained from the Air Fuel Ratio Table in Map Tuning. It depends upon *RPM 2* and *input MAP*.

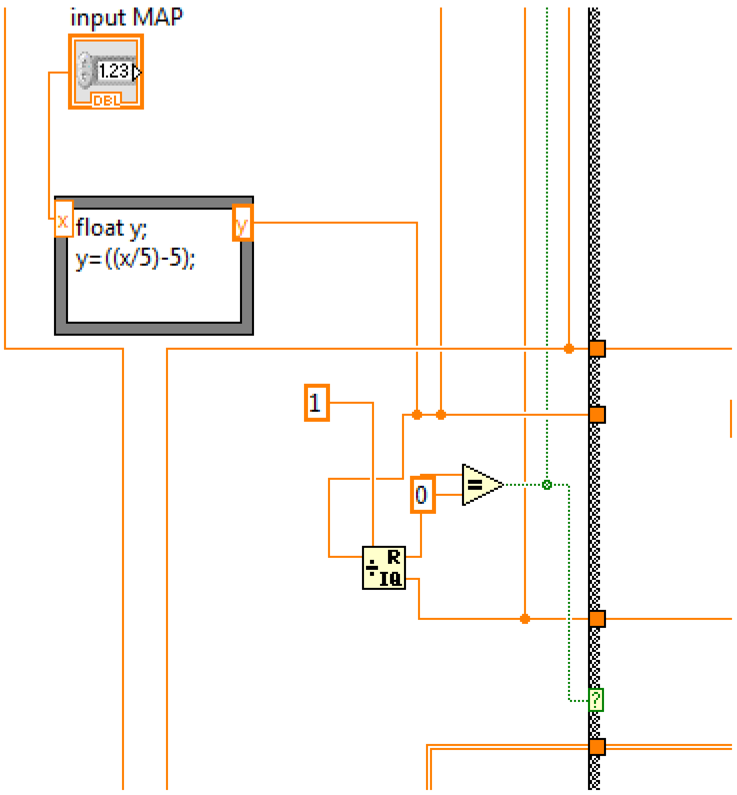


Figure 5.23

First input MAP is input into a *Formula Node*. Then this value is divided by 5, after which it is subtracted by 5. This is used to scale the table and start it from 25 with the interval of 5 units e.g. 25, 30, 35 and so on. Another point to be noted is that the minimum value of *input MAP*, according to the *MAP* bar in *Main View* tab, is 25 and the highest value is 100. That makes 16 steps in the Pressure axis/y-axis.

Then this value is checked whether it is a whole number or not. This is done by first dividing the output with 1 and getting the remainder, which is called P-remainder, and then comparing it with 0. The result of this comparison is inputted into main *Case Structure*. The quotient is also input into the *Case Structure*. We will name this quotient P-quotient to avoid confusion in this explanation. The *Case Structure* has two cases: True and False

True:

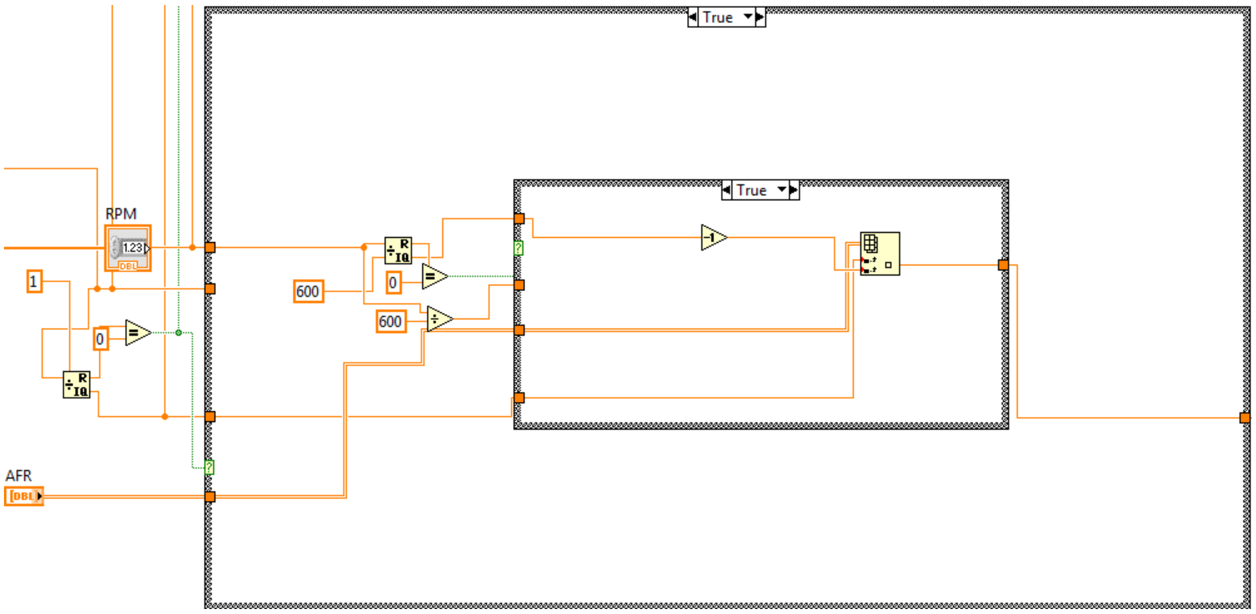


Figure 5.24

Now the RPM is divided by 600. This is done to scale the table in multiples of 600. The quotient of this arithmetic is called R-quotient to avoid confusion. And the remainder, which is called R-remainder, is compared to 0 to check whether it is a whole number. The answer to comparison is inputted into another *Case Structure*.

If the remainder is 0 i.e. the logic gives true output then the value of R-quotient is subtracted by 1. This is done because the first element of array is 0 and not 1. This value coupled with the P-quotient is used to index the array and get the corresponding output from the table. P-quotient is fed into the y-axis and R-quotient is fed into the x-axis. The fetched value is outputted from the loop.

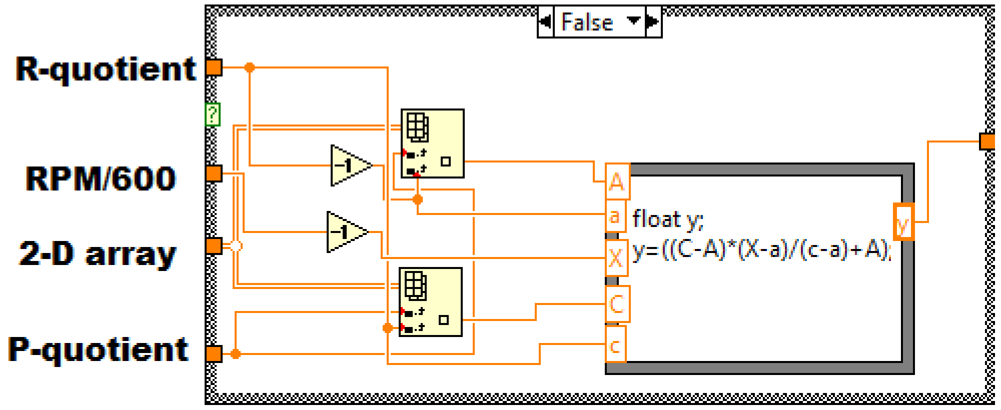


Figure 5.25

If however the remainder is not 0 and the case is False. Then the following two value are fetched from the 2-D array i.e. *Air Fuel Ratio Table* in *MAP tuning*:

1) Indexing using (R-quotient - 1) on x-axis and P-quotient on y-axis. This value is input into the *Case Structure* and is named A.
2) Indexing using R-quotient on x-axis and P-quotient on y-axis. This value is input into the *Case Structure* and is named C.

Then just like **1-D Interpolation,** which is described before, the final value from the graph is fetched and is output from the secondary as well as primary *Case Structure*.

False:

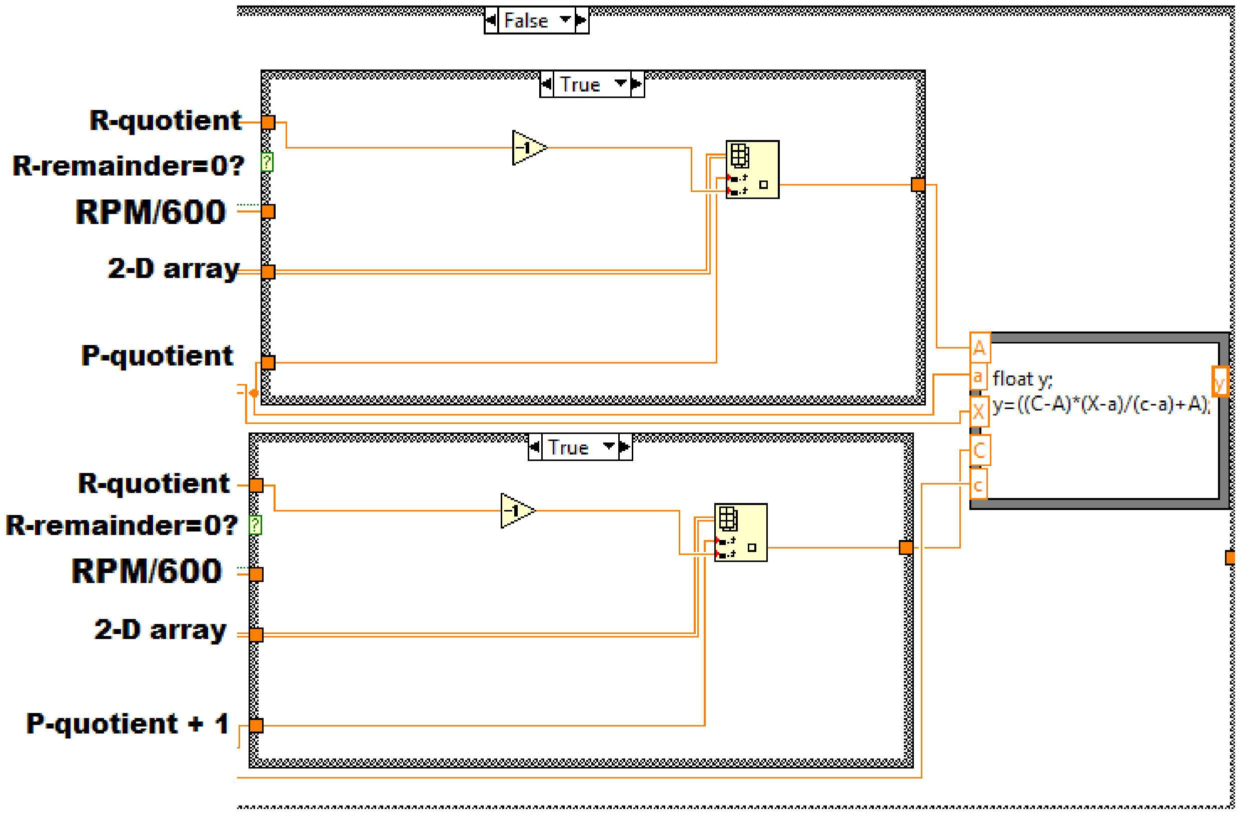


Figure 5.26

There are further two *Case Structures* in this Case. If R-remainder is 0. Then two values are fetched from the table.

1) Indexing using (R-quotient - 1) on x-axis and P-quotient on y-axis. This value is input into the *Case Structure* and is named A.
2) Indexing using (R-quotient - 1) on x-axis and (P-quotient + 1) on y-axis. This value is input into the *Case Structure* and is named C.

Then simple **1-D Interpolation** is done with respect to P-quotient.

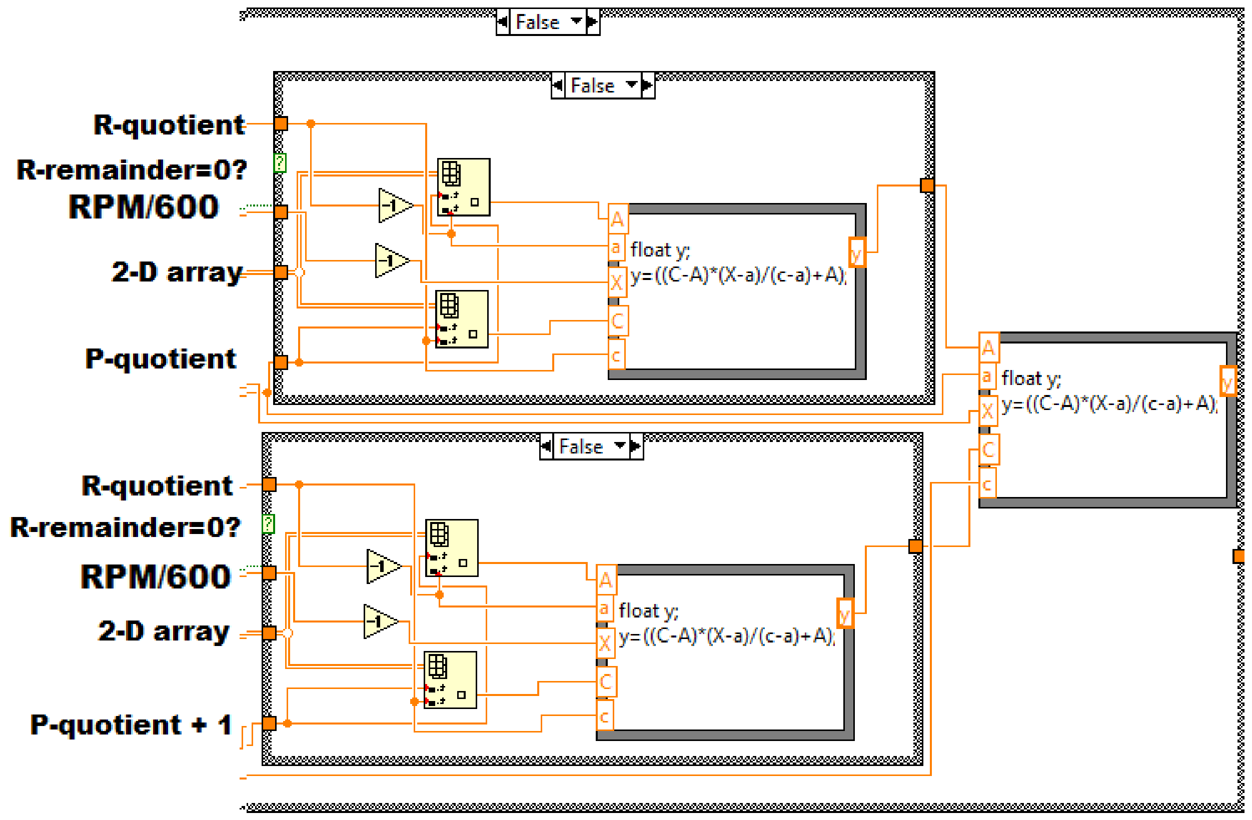


Figure 5.27

But if R-remainder is not equal to 0 then four values are fetched independently.

1) Indexing using (R-quotient - 1) on x-axis and P-quotient on y-axis.
2) Indexing using R-quotient on x-axis and P-quotient on y-axis.
3) Indexing using (R-quotient – 1) on x-axis and (P-quotient + 1) on y-axis
4) Indexing using R-quotient on x-axis and (P-quotient + 1) on y-axis

Values 1 & 2 are linearly interpolated i.e. **1-D Interpolation** with respect to x-axis/RPM. This value is called **x1.**

Values 3 & 4 are linearly interpolated i.e. **1-D Interpolation** with respect to x-axis/RPM. This value is called **x2.**

Now **x1** & **x2** are linearly interpolated i.e. **1-D Interpolation** with respect to y-axis. This is the final output of the loop.

(Note: This whole procedure of getting the value from 2-D array by using two variables: *RPM* and *Pressure* is called **2-D Interpolation** for referencing later on.**)**

5.2.12 SPARK ADVANCE MAP (in degrees) in the *Tables* tab.

The procedure of obtaining spark advance value from the *Spark Ignition Table* in *MAP Tuning* is almost the same as the method of getting *AFR from graph* i.e. **2-D Interpolation**. The only difference is that the 2-D array referenced is not the *Air Fuel Ratio Table* and instead is *Spark Ignition Table*.

5.2.13 AFR from load graph in the *Tables* tab.

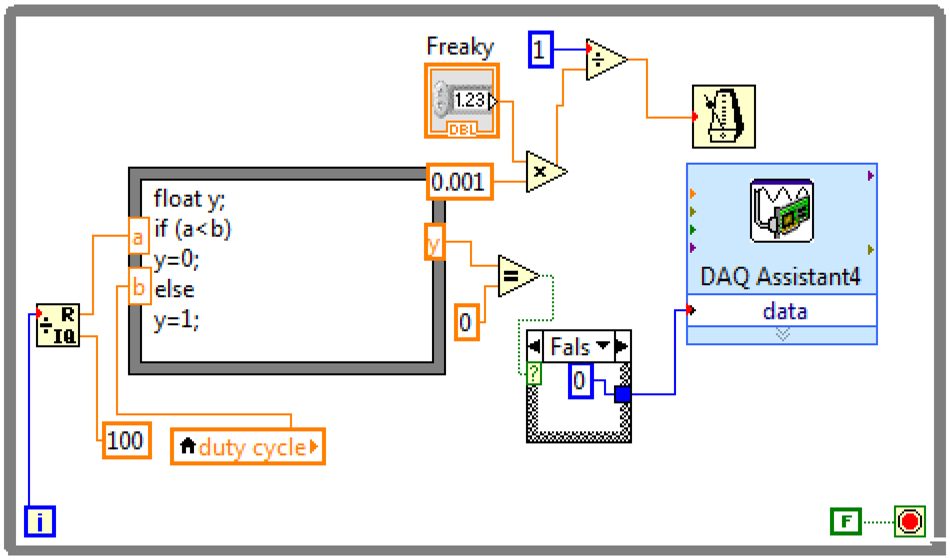
The AFR from load graph is also obtained by **2-D Interpolation**. However there are some differences from the method used in getting *AFR from graph*:

1) The y-axis is now *Load*. And *Load* is first multiplied by 0.25 to scale it according to our values inputted in the table: Our table values start from 0 and are incremented by 0.25 in every step. (The x-axis is still RPM)
2) The table used is now *Air Fuel Ratio Table* in *MAF Tuning*.

5.2.14 Spark Advance (Load) in the *Tables* tab.

The *Spark Advance (Load)* is also obtained by **2-D Interpolation**. It is similar to *AFR from load graph*. The only difference is that the 2-D array referenced is not the *Air Fuel Ratio Table* and instead is *Spark Ignition Table* in *MAF Tuning*.

5.2.15 Actual Output to Injector via DAQ (running at back end)



```
float y;
if (a<b)
y=0;
else
y=1;
```

Figure 5.28

The DAQ card that we are using does not support continuous analog output. So we setup DAQ Assistant to get *Sample on Demand* in analog output.

First we make a *while* loop to continue for infinity. Then we calculate *Freaky*. It is a variable which has the value of half the frequency of engine crankshaft i.e. RPM/120. *Wait until next ms multiple* is used to pause the loop for some time before restarting again. The time to wait is 1/(*Freaky**0.001). This 0.001 is a constant used to slow down the frequency in order for the user to observe the change in frequency of injector. This is solely for demonstrational purposes. The *i* from the *while* loop is divided by 100 and the remainder is fed into the *Formula Node*. The *duty cycle* (calculated in previous sections) is also inputted into the *Formula Node*. In the *Formula Node* it is checked whether the remainder for *i* is less than **b** (*duty cycle*). If it is so then *Formula Node* gives an output of 0. Otherwise the output is 1. The output is compared by 0 and if it is *True* then 0 is fed into the DAQ Assistant. And if it is *False* 5 is forwarded to the DAQ Assistant.

Logic: The time period of the output frequency becomes *Freaky*/100 (in seconds). The loop is repeated after every 100 values of *i*. And after every b or *duty cycle* vales of *i*, the value to DAQ Assistant shifts from 5 to 0.

5.2.16 Piston Displacement and Spark Advance Graph in the *Main View* tab

For piston displacement we have used:

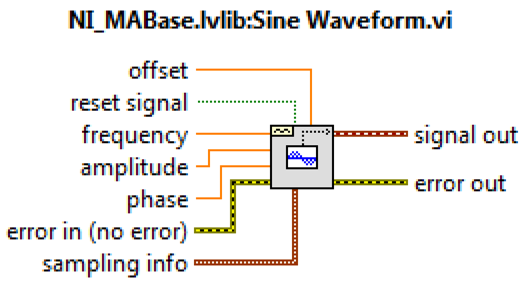


Figure 5.29

RPM is divided by 60 and then it is inputted into frequency. A constant is created at *amplitude* and its value is set to 5. Another constant is created at *phase* and its value is set to 90.

For Spark Advance we have used:

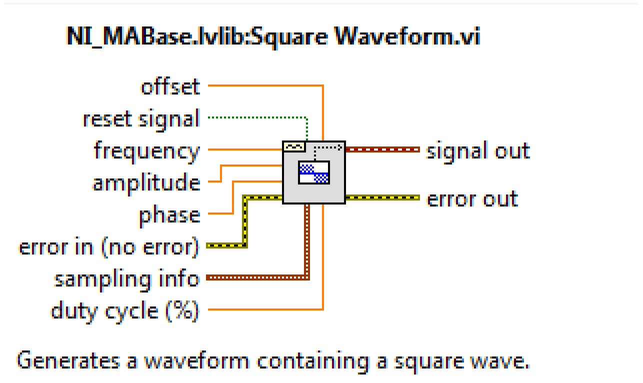


Figure 5.30

RPM is divided by 120 and then it is inputted into frequency. A constant is created at amplitude with a value of 5. Another constant is created at *duty cycle* and its value is also set to 5. *Spark Advance MAIN SCREEN* is divided by 2 and inputted to *phase.*

Both the signals are combined into an array using *Build Array*. The output is then displayed on *Piston Displacement* graph on the *Main View* tab.

5.2.17 **Injector Pulse** graph in *Main View* tab

We have used:

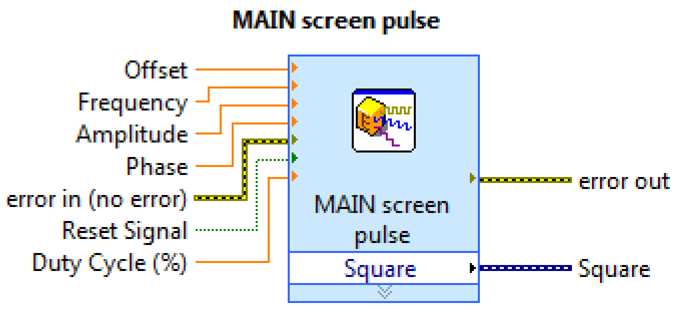


Figure 5.31

Duty cycle is inputted to *Duty Cycle (%).* A constant is created at *Phase* with a value of -45. Amplitude is set to 5. And *RPM* is divided by 120, after which it is inputted into frequency. The square wave output is connected to *Injector Pulse* graph.

5.2.18 **MAP / MAF toggle button** in *Main View* tab

There are two conditions of the toggle button:

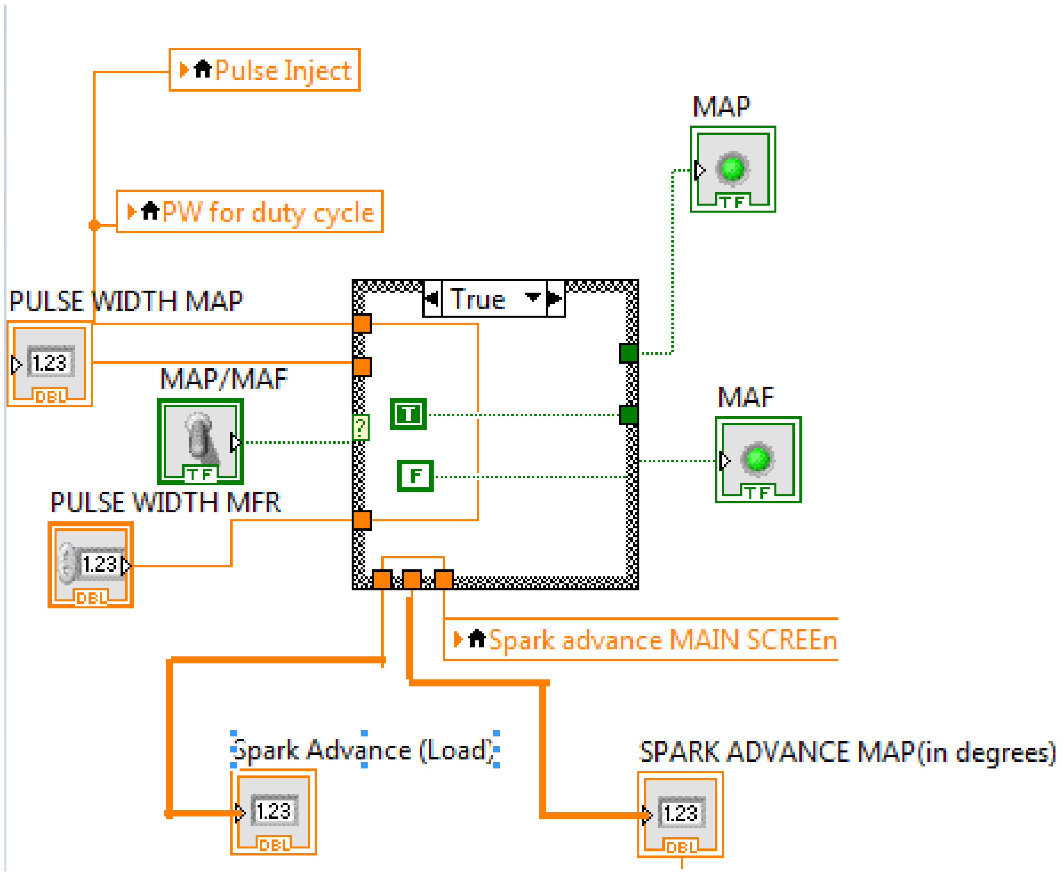


Figure 5.32

- *MAF* LED is on and *MAP* LED is off.
- *PULSE WIDTH MFR* is connected to the local variables which point to *Pulse Inject* and *PW for duty cycle*.
- *Spark Advance (Load)* is connected to the local variable which points to *Spark Advance MAINSCREEn.*

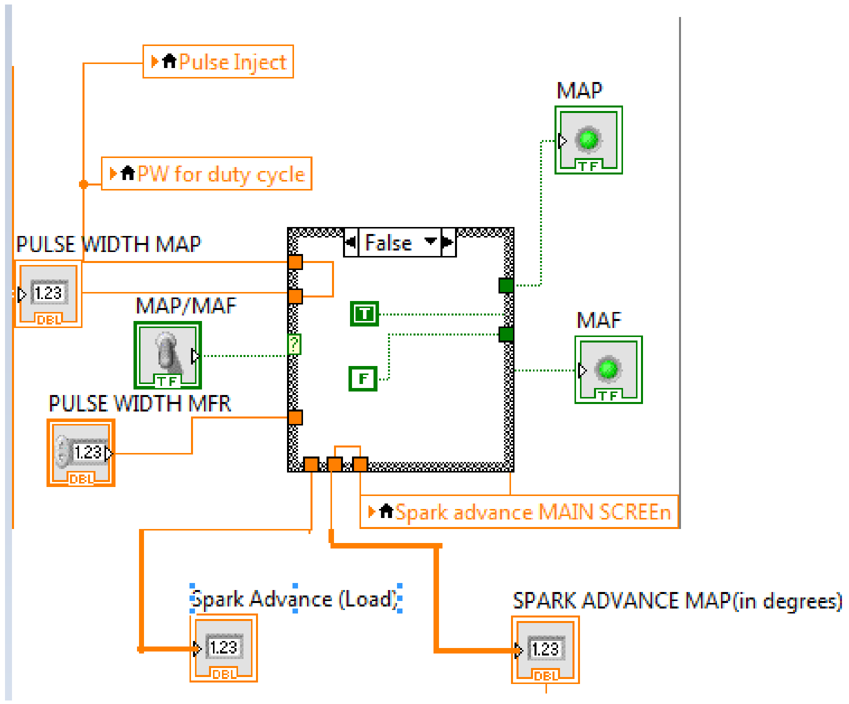


Figure 5.33

- *MAF* LED is off and *MAP* LED is on.
- *PULSE WIDTH MAP* is connected to the local variables which point to *Pulse Inject* and *PW for duty cycle*.
- *SPARK ADVANCE MAP(in degrees)* is connected to the local variable which points to *Spark Advance MAINSCREEn.*

CHAPTER 6

Engine Testing and Mapping:

6.1 Engine Testing:

Most engine testing is performed on an engine dynamometer. An engine dynamometer is able to very accurately control the rotational speed of an engine and therefore able to capture data at any engine speed. The whole range of possible engine speeds needs to be mapped, so the engine speeds that are analyzed range from idle to the engine's red line speed. This range of engine speeds is discritized so that data is taken about every 50 or 100 rpm.
27
At each engine speed, the load is varied by controlling the throttle's position.
Again the whole range of loads must be analyzed, so data is collected at throttle positions ranging from closed throttle to wide open throttle (WOT). All of the remaining parameters such as spark timing must be tested at each coordinate on the two dimensional space of engine speed and load. Depending on the operating condition one value of each parameter is chosen. For example at idle speed with no load, the spark timing that produces the lowest emission and most stable combustion would be chosen.

6.2 Engine Mapping:

"*Engine Maps*" refer to a wide set of one-dimension or two-dimension parameter tables loaded into the ECU to control all the engine parameters, that are, throttle opening, injection and ignition timings (duration, phase, etc...). The main two-dimensions maps take the throttle opening and the engine speed as inputs and return a proper parameter as output. All maps are filled with proper numbers during hours of engine calibration done at the test rig and/or at track..

While the engine is running its requirements for fuel and ignition timing will vary according to certain engine conditions, the main two being engine speed and engine load. A "map" is no more than a look up table by engine speed and load, which gives the appropriate fuel or timing setting for each possible speed and load condition. There will normally be a map for the injector timings (fuel map) and a separate map for the ignition timing settings (ignition map) within the Engine Management System.

Each map has entries for a pre-determined range of engine speeds (called speed sites) and a predetermined range of engine load conditions (called load sites) which generally indicate how far open the throttle is. The Engine Management System knows the engine speed (derived from the crank sensor or distributor pickup) and the engine load (from the Throttle Position Sensor or airflow meter) and will use these two values to "look-up" the appropriate fuel and timing settings in each map.

6.3 Types of driver maps:

The required maps are the following:

1. **The engine torque map**. This is a 2-dimension table with engine speed and throttle as inputs and torque as output. This map is defined point-by-point or by ramps at the test rig with the fired engine and the torque meter. Sometimes is trimmed on track if the car is equipped with torque meters on the transmission.

2. **The inverse engine torque map**. It is calculated by the engine torque map. It is a 2-dimensions table with torque and engine speed as inputs and throttle as output. It is computer calculated.
3. **The driver demand torque map**. This is the 2-dimension table with engine speed and driver normalized torque as input and engine torque as output. This is a reshaping of the engine torque. This is the core of the drivability because allow the engineers to completely reshape the engine torque in function of the engine speed, within the boundaries of maximum and minimum torque available (which is the torque at off-throttle – engine brake - and at wide-open-throttle).
4. **The pedal map**. This is a 1-dimension map where the input is the normalized accelerator pedal position (0-100%) and the output is the normalized torque (0-100%). It can be considered as a "gain" on the driver demand torque map.

6.4 How the driver uses the Engine Maps

On the practical way, the engineers provide a selection of pedal maps and torque maps, driver selectable by the driver using rotary selectors on the steering wheel. In addition to a manual selection, if the regulation permits, pedal maps and torque maps can be automatically selected by the space information on the ECU or selected by gears. In this way, the driver could have the optimal pedal map in function of the corner (different between low speed corners or high speed corners) or have negative slope torque map for low gears and standard map for high gears - all selected automatically.

The rotary selector for the pedal map is labeled somewhat like PEDAL and the driver demand torque map is labeled TRQ. Sometimes the two functions can be overlaid on single rotary selector, to save one rotary selector for other uses. It is recommended to always have a WET configuration on the rotary selectors, which can be used when grip is low. Pedal map for wet is usually soft-pickup shaped.

Everyday work at track is to modify the pedal maps to give the best support to the driver. Driver demand torque maps are usually developed on telemetry analysis overnight at track.

6.5 Types of driver torque maps:

With different pedal maps, the torque driver torque map change as follows:

1. Soft pickup – driveability for low speed corner or wet conditions
2. Standard linear map
3. Smooth top end – for high speed corner

Changing the pedal map is an easy way to change the sensitivity of the pedal while conserving the characteristic shape of the driver demand torque map. You notice that the negative slope is conserved while the distance of constant pedal torques change.

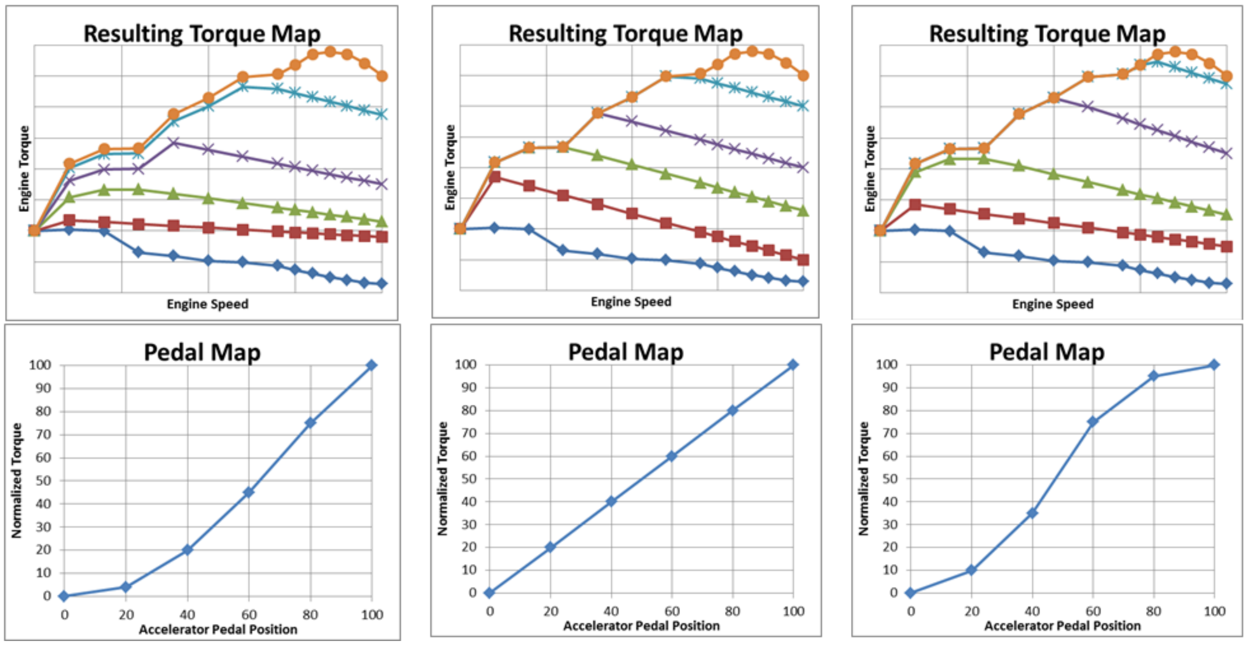


Figure 6.1

CHAPTER 7

Hardware Apparatus to test the engine control model:

Our Engine control module is capable of controlling pulse width as well as spark timing, but for demonstration we made assembly which demonstrates control of fuel injection. Main components of our hardware are:

7.1 Main components:

- MAF Sensor
- Thermistor
- Fuel injector
- Fuel pump
- Fuel Rail
- Fuel regulator
- Fuel Tank
- Support structure
- 5V, 12V Batteries
- H-bridge

7.2 MAF sensor and Pipe:

It measures the total engine load by measuring the amount of air flowing into the engine. These load values then used by Engine control module for calculation of pulse width and injection timing.

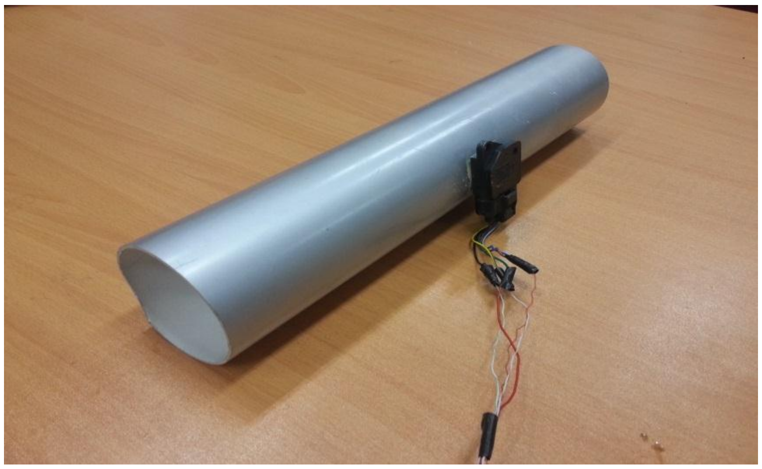


Figure 7.1

The greater the amount of air injected more heat thus current will be required to keep platinum wire at constant temperature gradient. Electronic circuit will then give voltage output according to the current required to keep platinum wire at constant temperature.

7.3 Thermistor:

It is used for measuring coolant temperature. This temperature is then used by control module to determine that if enrichment correction is needed or not.

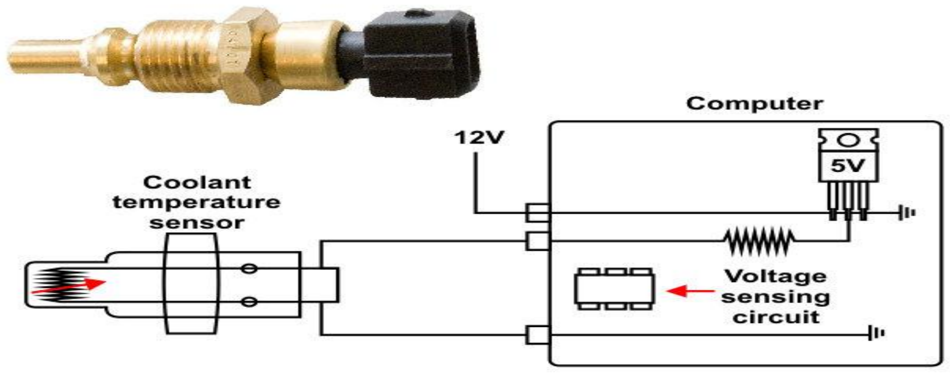


Figure 7.2

7.4 Fuel injector:

The Fuel Injector is an electronically controlled valve that is supplied with pressurized fuel by the fuel pump and when energized atomizes the fuel into a fine mist so that it can burn easily by the vehicle’s engine. It normally works with 12V and in our case it was taking input from computer via DAQ card so we required H-bridge circuit which increases voltage from 5 to 12 V and also the current to energise the fuel injector.

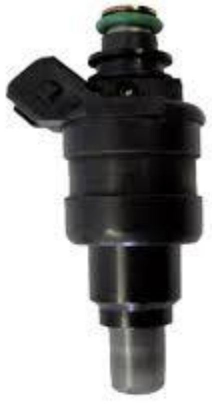


Figure 7.3

7.5 Fuel Pump:

As mentioned earlier Fuel injector required pressurized fuel, Fuel pump is required to do this job. A high-pressure pump running at around 6 bar which supplies fuel to the injectors. Our Fuel pump was immersed in fuel tank. The fuel pressure regulator regulates pressure between 3 and 4 bar .The extra fuel is bleed back into the fuel tank by pressure regulator.

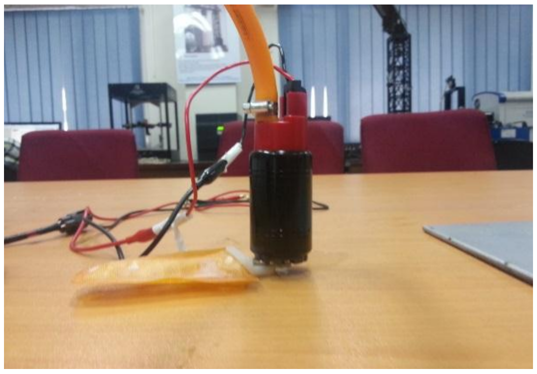


Figure 7.4

7.6 Fuel Rail:

It is used to deliver high pressure fuel to the individual injectors.

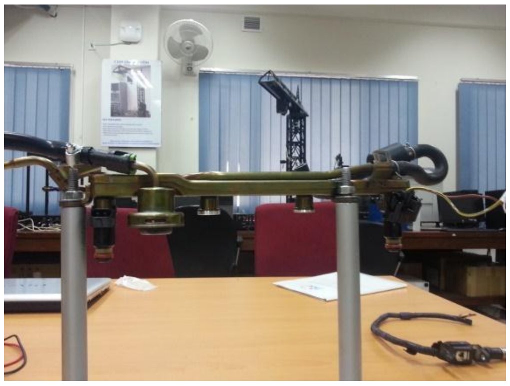


Figure 7.5

7.7 Fuel Regulator:

It maintains the pressure of fuel in fuel rail and bleeds extra fuel back into the fuel tank.

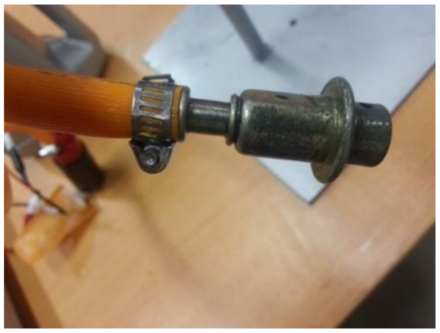


Figure 7.6

7.8 Fuel delivery Assembly layout:

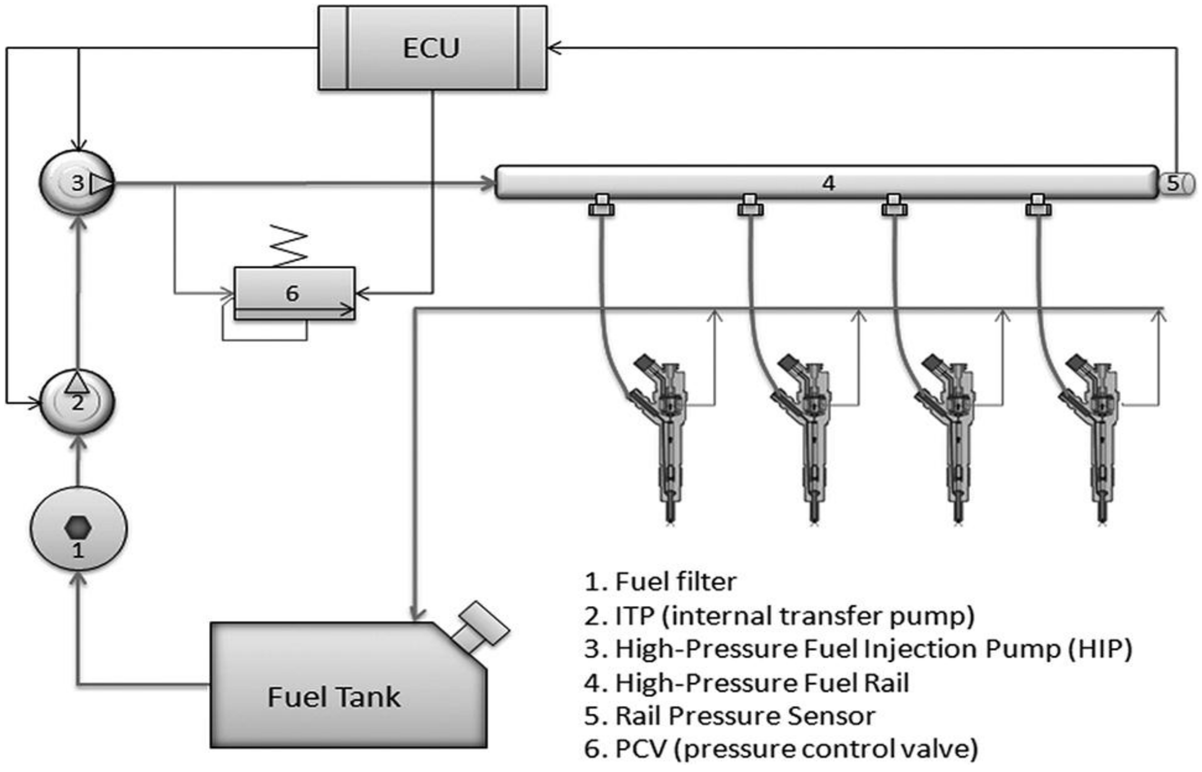

ECU
3
4
5
6
2
1
Fuel Tank
1. Fuel filter
2. ITP (internal transfer pump)
3. High-Pressure Fuel Injection Pump (HIP)
4. High-Pressure Fuel Rail
5. Rail Pressure Sensor
6. PCV (pressure control valve)

Figure 7.7

7.9 Batteries:

5v and 12 v batteries were used for entire hardware 12 V battery with current rating of 8 ampere was used to power fuel pump. Air flow sensor required both 5 and 12 V. Coolant temperature required 12 V. Injectors required 12 V but as we were controlling injector using DAQ card which is unable to give more than 5V so we required H Bridge between DAQ card and injector for voltage and power increase.

Figure 7.8

7.10 H Bridge:

H Bridge circuit was used to give 12 V to the injectors. DAQ card which we used to control the injector can give maximum 5V so we needed an H Bridge to power injectors. We have used L298N IC. It has current rating of 2 Amperes which is ideal for our application.

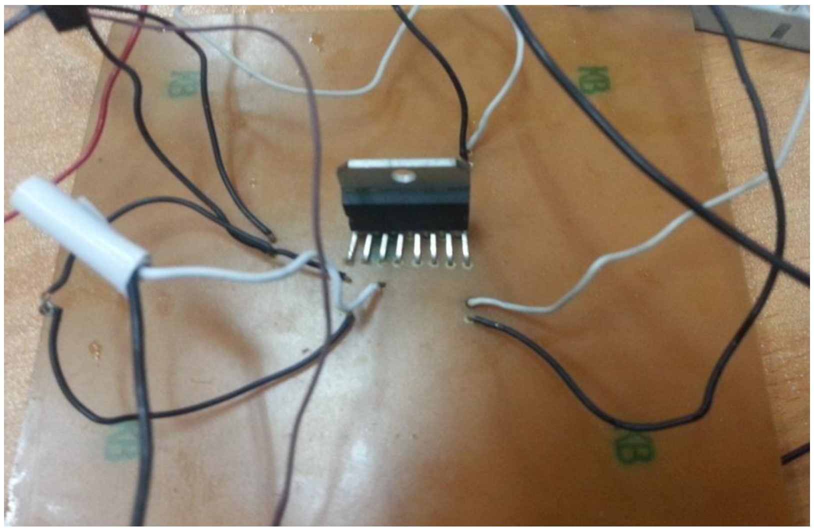


Figure 7.9

7.11 Hardware Assembly

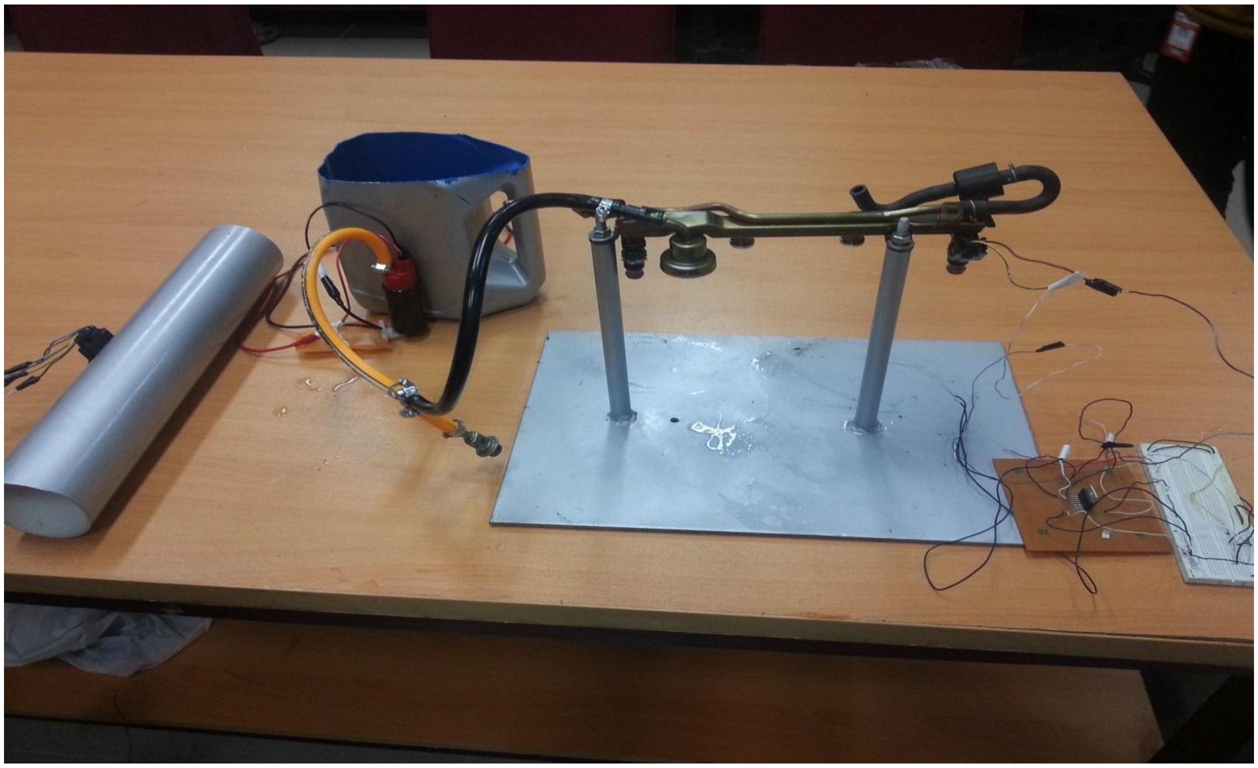


Figure 7.10

CHAPTER 8

Future prospects

1. More efforts are required to run module using microprocessor instead of Laptop to increase its practical application.
2. Latest DAQ card can be used to get accurate high frequency output using *continuous sampling*
3. Lambda and crankshaft sensors can also be integrated with the software to complete the model.
4. Barometric correction can also be implemented with a proper sensor.

References

http://www.gmhightechperformance.com/tech/0704gm_top_10_tuning_tools/viewall.html

http://www.enginelogics.com/engine-management-basics/

http://f1framework.blogspot.com/2013/03/f1-engine-maps_28.html

http://www.importtuner.com/tech/0612_impp_engine_tuning_basics/fuel_injection.html

http://www.enginelabs.com/engine-tech/dyno-testing/taking-your-engine-to-a-dyno-facility-the-first-time/

http://www.howstuffworks.com/ignition-system1.htm

http://www.ni.com/white-paper/3312/en/

http://en.wikipedia.org/wiki/Crankshaft_position_sensor

http://www.autoshop101.com/forms/h20.pdf

http://www.gerrysap.com/efi.html

http://www.edelbrock.com/automotive/mc/efi/

https://www.diyautotune.com/

http://www.omnitekcorp.com/ems.htm

http://www.flyefii.com/EFII_desc.htm

http://www.megamanual.com/v22manual/mfuel.htm

http://www.megamanual.com/begintuning.htm

http://www.epi-eng.com/piston_engine_technology/volumetric_efficiency.htm

http://books.google.com.pk/books?id=Mr-93vKGa-cC&pg=PA49&lpg=PA49&dq=how+to+calculate+load+from+MAF+sensor&source=bl&ots=1TUnBKvkBx&sig=IdA3Df_fotbhQ0Lj5E6OAswutRg&hl=en&sa=X&ei=Ev18U9yLNqj24QTX1oDgBg&ved=0CCUQ6AEwADgU#v=onepage&q=how%20to%20calculate%20load%20from%20MAF%20sensor&f=false

Books:

Automotive Engine testing 3rd edition by A.J Martyr

Bosch Electronic Automotive Handbook

Tune to win by caroll smith

Engine Management Advanced Tuning by Greg Banish