Nelson Mandela Metropolitan University

for tomorrow

The development of a programmable engine management system for a Formula Student race vehicle

by

Hiten Parmar

(Baccalaureus Technologiae: Engineering: Electrical)

A research dissertation submitted in compliance with the requirements for the degree of

Magister Technologiae: Engineering: Electrical

in the Department of Electrical Engineering,

Faculty of Engineering, the Built Environment and Information Technology

of the Nelson Mandela Metropolitan University

Promoter: Professor Theo van Niekerk

January 2012

Declaration

I Hiten Parmar hereby declare that:

- the work in this dissertation is my original work;
- all sources used or referred to have been documented and recognized and
- this dissertation has not been previously submitted in full or partial fulfilment of the requirements for an equivalent or higher qualification at any other recognized educational institution.

…………………………
Hiten Parmar 23/01/2012

Acknowledgments

- My heartfelt gratitude goes to Nazeem Sirkhotte for the knowledge he shared throughout the development of this project.

- Appreciation to Riaan Ehlers and Frank Adlam for their support on the PIC24 development board.

- Thank you to the VWSA-DAAD International Chair of Automotive Engineering for the support offered for my trip to Wolfsburg, Germany.

- My sincere thanks to my mentor, Professor Theo van Niekerk, for his support and motivation during this project.

- Thank you to my parents and friends for all their support and encouragement throughout this project.

- Lastly, to the members of the NMMU Racing team, thank you to all who supported me through this project.

Table of Contents

List of Figures

List of Tables

Abbreviations

AFR – Air to Fuel Ratio

BTDC – Before Top Dead Centre

CAD – Computer Aided Drawing

COP – Coil-on-Plug

CPI – Central Port Injection

DAQ - Data Acquisition System

ECU – Engine Control Unit

EFI – Electronic Fuel Injection

EMS – Engine Management System

MPI – Multi-Point fuel Injection

OEM – Original equipment from manufacturer

RPM – Revolutions per minute

TBI – Throttle Body Injection

TDC – Top Dead Centre

TPS – Throttle Position sensor

Glossary of Terms

Combustion – the burning of the air-fuel mixture in the cylinders of an engine

Engine Management System (ems)/ Engine Control Unit (ecu) – is an embedded system that controls one or more of the electrical subsystems in a vehicle. The ems interprets these parameters in order to calculate the appropriate amount of fuel to be injected and among other tasks controls engine operation by manipulating fuel and/or air flow as well as other variables

Fuel injection – is a means of metering fuel into an engine

Ignition timing – the time occurrence of ignition measured in degrees of crankshaft rotation relative to the extreme highest point of the piston during its stroke i.e. Top Dead Centre (TDC)

Stoichiometry – the air to fuel ratio for perfect combustion, enabling exactly all fuel to burn using exactly all of the oxygen in the air

Abstract

Formula student teams are faced with a challenge to attain peak engine performance under the given rules of the competition. Teams are enforced to use a 20mm circular restrictor in the air intake system. The restrictor serves as a safety measure to limit the power capability of the engine. As a result the original manufacturer's air intake manifolds cannot be used and a new design has to be implemented. Due to the decreased air flow into the engine by the restrictor, the manufacturer's engine management system does not provide adequate control of the engine for peak performance. Teams aim to gain as many points as possible in the engineering design judging as well as on the dynamic events on the track. Due to budget constraints that most student driven projects face, formula student teams are limited to good aftermarket ems's due to this high costs.

This project aims to provide a cost effective solution to the Nelson Mandela Metropolitan University (NUUMU) Racing Team. The master's thesis provides an overview of engine management systems and as well as a solution of a functional prototype to be used on a formula student vehicle. The initial objective of the design to control the fuel injection and ignition timing parameters of the Honda CBR600 engine was achieved. Hardware prototype and complete software code were developed.

Full signal analysis during engine idle conditions illustrated the successful control of the engine. Under the specified load condition, fuel and ignition timing parameters were adjusted via the controller board and the resultant engine responses were recorded. This project forms a base model for future developments for the formula student team to expand into greater vehicle control systems and be recognised for engineering design at the international competitions.

Chapter 1: Introduction

Formula Student is an international competition that promotes careers and excellence in engineering. It challenges university students to market, design, build and compete as a team with a single seat formula race car. Team members go the extra step in their education by incorporating into it intensive experience in building and manufacturing together with considering the economic aspects of the automotive industry. Teams take on the assumption that they are a manufacturer developing a prototype to be evaluated for production [1].

Regulations of the competition, applicable to this thesis, are that a four stroke piston engine with a displacement not exceeding 610cc per cycle be used to power the car. Further to this, a single circular 20mm restrictor must be placed in the intake system to limit the power capability of the engine [1]. The challenges faced by a team are to compose a complete package consisting of a well-engineered race car and a cost effective sales plan that best matches these given criteria. The end result is a great experience for young engineers in a meaningful engineering project with dedicated team effort.

The regulated 20mm air intake restrictor limits the overall engine performance by reducing the amount of airflow into engine. To overcome this, a programmable engine management system (ems) is required to adapt fuel and ignition timing parameters to the reduced amount of air flowing into the engine. The original equipment from manufacturers (OEM) control systems are not able to compensate for this as their functionality is solely based upon the original air intake design. Any aftermarket modifications to the engine and its related components will consequently affect the engine performance.

The purpose of this project is to develop a programmable ems - also referred to as an engine control unit (ecu), for a Formula Student race car. This design will offer a versatile engine management system providing electronic control of the fuel injection and ignition timing system. By using feedback signals from the engine sensors, combined with pre-programmed information, metered fuel and ignition spark will be delivered to the cylinders at the optimal time for peak engine performance.

1.1 Problem statement

This thesis was formulated from the requirements of a programmable ems for the NMMU formula student race car. The team were faced with limited financial resources and were aiming to establish high recognition for engineering design at the competition.

OEM engine management systems employ complex algorithms and as a result the possibility to adjust any fuelling or ignition timing parameters on these is virtually impossible. Some teams use ‘piggy back’ systems to manipulate the OEM input/output signals which in-turn gives some control to adjust engine parameters. However, these units are not favourable to the judges as noted by the researcher’s attendance at two formula student events. The aftermarket programmable ems’s are costly and most require a specialised skill level and training. The conclusion was to develop a programmable ems within the university.

1.2 Objectives

Engine management systems are able to provide a wide range of functions for engine control, each of which results in different output responses. However the base of every ems is essentially the control of the fuel injection and ignition timing parameters. A programmable ems is ideally suited under the formula student application to allow precise fuel and ignition timing quantities for the reduced airflow into the engine.

A microcontroller based system with adequate processing speed is required for all signal processing, calculations and output control. The raw engine sensor signals are required to be filtered and conditioned. The output controlling factors include high current drivers for coil based actuators. Fuel injection and ignition timing are crucial in respect of time and thus a precise software code structure is to be implemented.

Further to a basic control of fuel injection and ignition timing, an ems design is expandable into closed loop engine control, traction control and electronic stability control. Improvements to implement communication platforms such as CAN bus networks are widely possible.

1.3 Hypothesis

The OEM engine management systems fail to provide optimal engine performance under the formula student application. A programmable ems will be developed to control fuel injection and ignition timing to achieve the desired air-fuel ratios for ideal optimal engine performance.

1.4 Delimitations of research

To enable a successful prototype engine management system to be developed, specific conditions are to be outlined. In respect of this thesis the following delimitations were applied:

- Research and development will be within the rules stipulated for the Formula student competition.
- A Honda CBR600 engine (PC37) will be the designated engine used.
- To provide fuel injection and ignition timing control only.
- During all calculations, atmospheric conditions are all kept at a constant.

1.5 Research methodology

This research project primarily focuses on engine control through fuel injection and ignition timing control. To achieve the desired objectives the following procedures were followed in the project research:

- An in-depth literature survey into engine management systems, variations in fuel injection strategies and ignition systems.

- Discussions with the motor sport industry on existing programmable ems's on their basic operation and features of the available systems. Secure a better understanding of the engine controlling parameters.

- A proposed design concept for fuel injection and ignition timing control on a Honda CBR600 engine using available engine components.

- A literature survey into suitable microcontrollers and supporting hardware interface components was carried out. Considerations were taken into account for automotive sensor signals and possible electrical interference.
  A concluding hardware design was then developed.

- The development of the microprocessor software to enable processing of input signals from the sensors, data processing and calculations for desired output control.

- Simulation tests of the hardware representing sensor signals and conditioning for microprocessor input pins. Validation of the software code for the control of the outputs.

- Integrating the complete system onto an engine in a dyno cell. Establishing a condition to be able to carry out tests and acquire data for validation of the prototype functionality.

1.6 Conclusion

From the outlines of the problem statement and stated objectives, this thesis is divided into four chapters.

Chapter 2 is an overview and introduction to engine management systems.
It outlines the basics operation these system, their inputs and output devices. The chapter also shares a small insight into engine testing and the equipment available for the testing purposes in this project.

Chapter 3 explains the experimental setup and build-up of the project. A breakdown description of the hardware and software development phase of the project is discussed.

Chapter 4 gives a basic insight into engine tuning. With a programmable ems there has to be to some knowledge of engine tuning to be able to setup control of the engine. The chapter outlines the general guidelines along lambda and ignition timing values.

Chapter 5 indicates the testing carried out and results obtained after. The chapter concludes with a conclusion on the whole project.

Chapter 2 – Principles of an engine management system

The two main aspects of ems include providing the right amount fuel and delivering the spark at the right time to get perfect combustion in the engine. Over recent years there have not been many new developments on the fuel delivery components since the early applications of electronic control units on engines, however with the ignition system components it is the opposite.

The fuel delivery system consists of a fuel tank, fuel pump, fuel pressure regulator and fuel injectors [2, 3, 9]. The fuel tank is for fuel storage and is normally located at the rear of the vehicle. Within the tank is the fuel gauge sending unit, a filler tube and on most fuel injected vehicles the fuel pump is housed within the fuel tank. The fuel pump ensures a constant supply of fuel to the fuel rail in which the fuel injectors are housed. The average fuel line pressure that the pump must maintain is approximately 3 bars [9]. The fuel pressure regulator is usually mounted at the end of the fuel rail comprising of a mechanical mechanism that is connected to the inlet air intake manifold to ensure a constant differential pressure across the fuel injectors. The fuel injectors are simple solenoid-operated valves designed to operate at high speed and produce a finely atomized spray pattern into each cylinder [2]. The fuel injector is one of the major output components of this system and will be explained further in the later parts of this chapter.

Earlier ignition system components comprised of an ignition coil, distributor, plug wires and spark plugs. The newer systems tend to have fewer components as well as electronic ignition modules integrated into the system to improve the spark delivery. The ignition coil is the unit that takes the relatively weak battery power and turns it into a spark powerful enough to ignite air-fuel mixture. Inside the traditional ignition coil are two sets of coils of wire on top of each other known as windings - primary and secondary windings. The distributor directs the spark from the coil to the respective cylinder in a pre-set sequence known as the ignition firing order. The plug wires, also known as high tension leads, carry the high voltage from the distributor to the spark plugs.

These leads must meet or exceed stringent ignition product requirements some of which include insulation to withstand 40 000V, temperatures from -40°C to +260°C and radio frequency interference suppression [8]. The spark plugs are sealed electrodes which allow the spark to jump across in the cylinder. These components must be able to withstand very high voltages, pressures and temperatures.

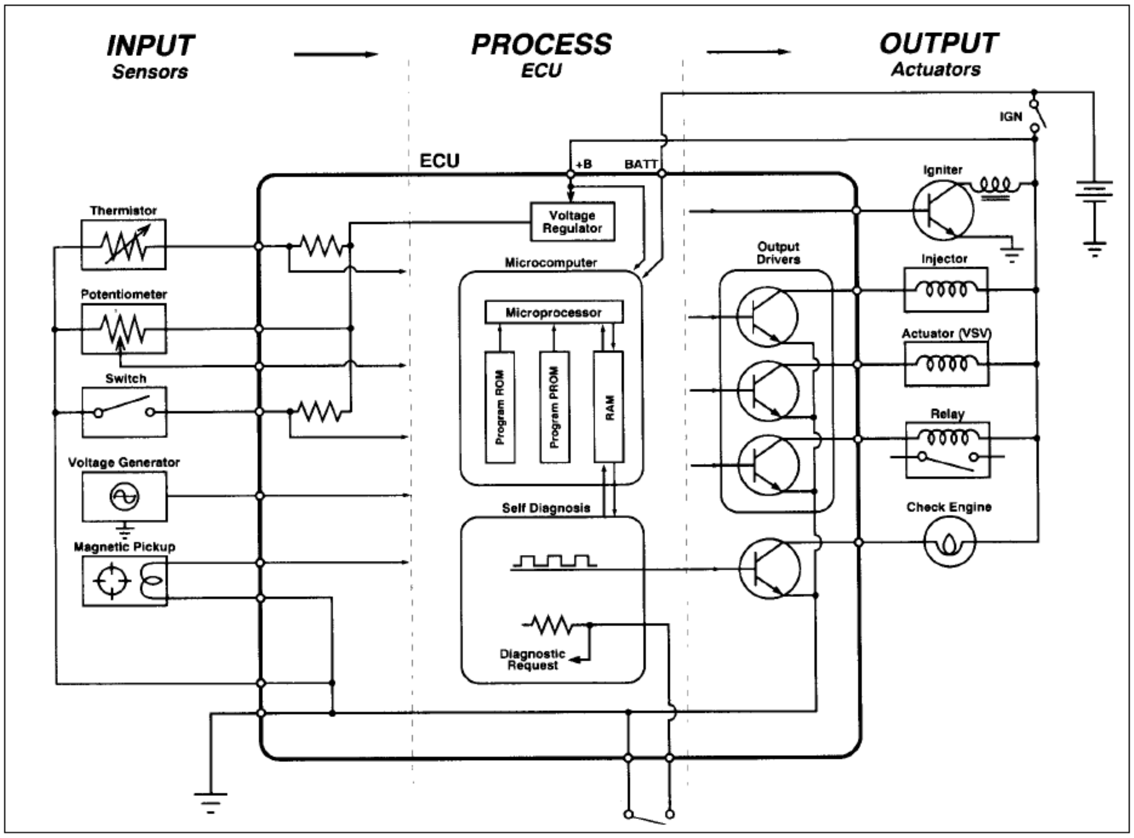


Figure 2.1: Engine management system overview [17]

Figure 2.1 above illustrates an overview of an ems as described earlier. All the components are systematically controlled by the ems which essentially determines the quantity of fuel, ignition timing and other parameters by monitoring the engine through the various sensors. The ems receives a wide number of readings from sensor all over the engine. Built into the ems is a fuelling and ignition map which is basically a gigantic table of numbers. It is like a lookup table that the ems uses to determine injector pulse width and spark timing.

On receiving these set of values from all the sensors, which it then looks up in the fuelling and ignition map, the point which all these numbers coincide is a final number which the ems then uses to set the injector pulse width. These are the manufacturer’s ‘blessed’ fuelling parameters, and for re-mapping aftermarket engine tuners can alter these mapping tables to make the engine behave differently, in most cases where non-factory modifications have been done to the engine. [4, 9]

It has been found that OEM ems’s are sold to be more economy and comfort based than performance based. Most engines have a lot of room for generating more power or torque and aftermarket programmable ems‘s tend to allow for this. These ems’s are required where significant aftermarket modifications have been made to a vehicles engine, or simply if the consumer is after more power out of the standard engine [4, 9]. The modifications include any internal engine component changes,
air intake or exhaust system modifications, or conversions to run on alternative fuel. As a consequence of these changes, the factory fitted ems may not provide appropriate control for the new engine configuration and this leads to a need for a programmable engine control unit.

2.1 Sensors (inputs)

Microcontroller controlled systems continually monitor the operating conditions of today’s vehicles. Through the sensors, computers receive vital information about a number of conditions, allowing minor adjustments to be made far more quickly and accurately than mechanical systems. The sensors convert temperature, pressure, speed, position and other data into either digital or analog electrical signals [3, 11].

A digital signal is a voltage signal that is either ‘on’ or ‘off’ with nothing in between. The signal from the switch could be 0 volts when off or 12 volts when on. Instead analog signals have a variable voltage. For example the throttle position sensor may vary the voltage signal anywhere between 0 volts and 5 volts depending on the actual position of the throttle.

The digital signal is the easiest for the ems to understand since it reads the signal as either ‘on’ or ‘off’. The analog signal must be conditioned or converted to digital for the ems to be able to interpret it [3].

A vehicle has many different sensors, the 3 main categories being voltage-generating, resistive and switches. Voltage-generating sensors generate their own voltage signal in relation to the mechanical condition it monitors. This signal in turn relays to the ems the data about the condition of the system it controls. Resistive sensors react to changes in mechanical conditions through changes in its resistance. The ems supplies a regulated voltage or reference voltage to the sensor and measures the voltage drop across the sensor to determine the data.

Switch sensors toggle a voltage from the ems high or low, or supply an ‘on’ or ‘off’ voltage signal to the ems. This type of sensor may be as simple as a switch on the brake pedal or as complex as a phototransistor speed sensor.

The two types of sensors that will be used in this development system are the crankshaft position sensor and throttle position sensor (TPS).

2.1.1 Crankshaft position sensor

This sensor generates a signal that the ems uses to determine the speed of the engine rotation and position of the crankshaft relative to the position of number one cylinder. With some engines, a separate camshaft position sensor is also used to inform the ems of the correct firing order of the cylinders [2, 4, 9]. The physics behind its operation include the principle of magnetic induction. A toothed gear mounted on the rotating crankshaft passes by the sensor and disrupts this magnetic field. The disruption in the field causes the sensor to produce a sinusoidal voltage signal. Essentially each tooth produces a pulse and as the gear rotates faster more pulses are produced in which the number of pulses in one second is the signal frequency. The frequency and amplitude of the sinusoidal voltage signal are proportional to the speed of the rotating gear [2, 3, 17].

The amplitude of the signal voltage is also directly related to the distance between the sensor coil and the toothed gear. This distance is referred to as the air gap and is critical to ensure signal output at lower speeds where the greater the distance the weaker the signal.

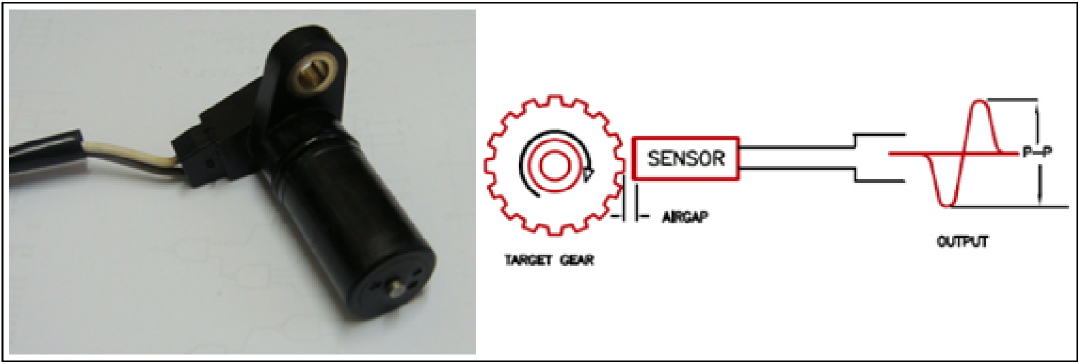


Figure 2.2: Crankshaft position sensor [16]

Figure 2.2 above illustrates the sensor located in the Honda CBR600 engine as well as the typical signal waveform. There is generally an approximate 1.25mm air gap between the sensor and the toothed gear. The gap is critical as the signal output is in relation to engine cranking speed and air gap spacing. Air gaps smaller than 0.75mm will create higher than normal voltages resulting in some high speed abnormalities and potential circuit damage. Air gaps larger than 1.8mm will thus result in weaker voltage signals and some engine starting problems due to weak and erratic signal [3, 16].

The frequency of the output signal is necessary to calculate the speed of the engine, ignition timing and the operation of the fuel injectors. This is a critical sensor and an engine with a malfunctioning sensor will not operate correctly.

There are two variations of crankshaft position sensors, magnetic and hall-effect, in which the output waveforms are illustrated in figure 2.2 and figure 2.3 respectively.

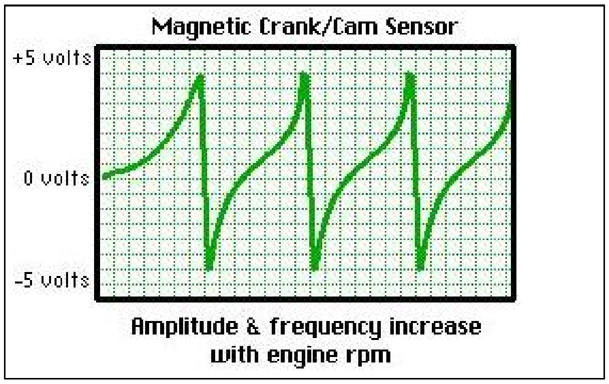


Figure 2.3: Magnetic crankshaft position sensor waveform [18]

The magnetic type uses a magnet to sense notches in a gear mounted at the end of the crankshaft. As a notch passes, it causes a change in the magnetic field that produces an alternating voltage signal out of the sensor. The sensor signal varies with engine speed and air gap spacing in which the signal strength can vary from an approximate +200mVAC to -200mVAC at very slow cranking speeds to +150V to -120V at high speeds. The frequency of the signal supplies the ems with the information it requires to calculate engine speed as well as to control ignition timing. These sensors have 2 wires which carry the generated signal voltage and do not need an external power source. The wires are twisted and shielded to prevent electrical interference from disrupting the signal.

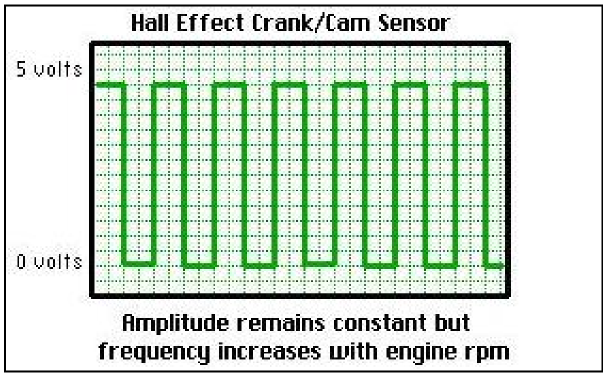


Figure 2.4: Hall-effect crankshaft position sensor waveform [18]

The hall-effect type of sensor uses notches or shutter blades on the crankshaft to disrupt a magnetic field in the hall-effect sensor window. This causes the sensor to switch on and off producing a digital signal that the ems reads to determine crankshaft position and speed. In principle, a controlled voltage from the ems is routed to a semi-conductor wafer in the hall sensor. A permanent magnet positioned beside the semiconductor induces a hall voltage across the semiconductor. The crankshaft sensor is positioned in such a manner that its metal blades pass between the semi-conductor and the permanent magnet. As the blade rotates through the vanes in the sensor, the signal voltage oscillates from a low to high state, generating a square wave signal [3].

These sensors typically have three terminals; one for input voltage, ground and the output signal. The sensor must have a voltage and ground source to produce an output signal. A hall-effect sensor is similar to a magnetic sensor in that it uses a stationary sensing unit and a rotating trigger wheel. The hall sensor has a distinct advantage over magnetic sensors, especially at lower speeds. Full output voltage is present at lower speeds and it is not changed by the speed of the trigger wheel.

2.1.2 Throttle Position Sensor (TPS)

The accelerator pedal in a vehicle is connected to a throttle valve which regulates how much air enters into the engine. In the normal state the throttle valve is in a closed position and when the pedal is pressed the throttle valve opens up more letting in more air into the engine. The TPS sensor is mounted at the one end of the throttle valve. This sensor is a potentiometer (variable resistor type) which has three terminals, one for power input, ground and a variable voltage output. These are mechanical devices whose resistance can be varied by the position of the movable contact on a fixed resistor [3].

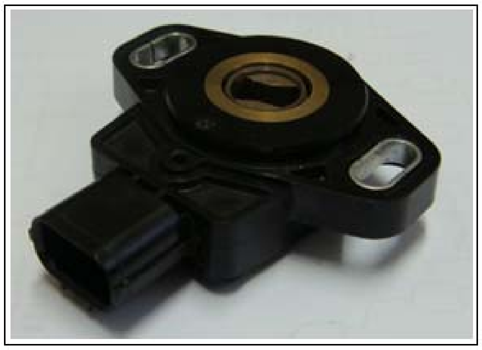


Figure 2.5: Throttle Position Sensor

Figure 2.5 illustrates the TPS sensor on the Honda CBR600 engine. The movable contact slides across the resistor to vary the resistance and as a result varies the voltage output of the potentiometer. The output becomes higher or lower depending on whether the movable contact is near the resistor's supply end or ground end. In essence, at closed throttle the voltage signal will usually be below 1 volt and at wide open throttle the voltage will be above 4 volts.

Figure 2.6 is an illustration of the typical output signal waveform from a TPS sensor.

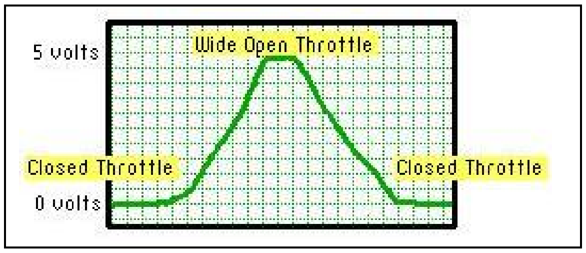


Figure 2.6: Throttle Position Sensor signal waveform [18]

2.2 Actuators (outputs)

The ems uses sensor data to control different systems on the vehicle through the use of actuators. These are electromechanical devices such as a relay, solenoid or motor.

Solenoids are essentially digital actuators, being either ‘on’ or ‘off’. One terminal is attached to the battery voltage while the other is attached to the ems which opens and closes the ground circuit as needed. When energized, the solenoid may extend a plunger or armature to control functions such as fuel injection. Most actuators are solenoids [3, 4].

Solenoids are controlled by two ways, pulse width or duty cycle. Pulse width control is used when the frequency is not consistent. An example of a pulse width is a fuel injector which is turned on for a determined period of time and then switched off. Duty cycle control is used when the frequency remains constant. A duty cycle solenoid in an anti-lock braking system is designed to be ‘on’ and ‘off’ for a specific time [2].

Actuators can adjust engine idle speed, change suspension height or regulate fuel metered into an engine. The two types of actuators that will be used in this development system are the fuel injector and ignition coil which will both be reviewed in detail.

2.2.1 Fuel Injector

The purpose of the fuel injector is to precisely inject the correct amount of fuel at the correct time. Based on the input signals, the ems will decide when and how much fuel to inject. There is usually one fuel injector per cylinder mounted on the intake manifold of the engine – multiport injection [8]. Fuel injectors are basically electro-mechanically operated needle valves. Figure 2.7 below illustrates fuel injector in the Honda CBR600 engine.

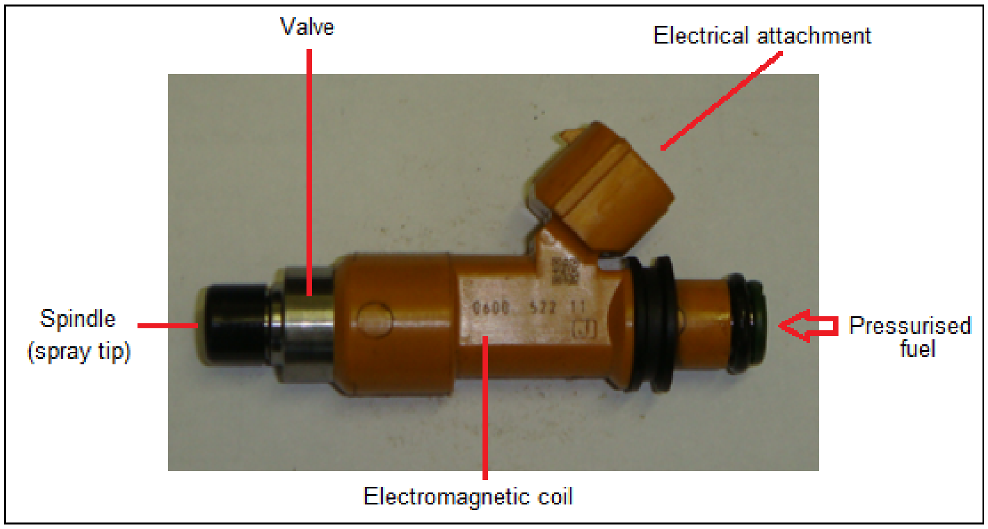


Figure 2.7: Fuel Injector

When a current is passed through the electromagnetic coil, the windings are energized and the armature is attracted due to magnetism and compresses a spring. The valve then opens, and the pressurized fuel is forced through the spray tip. The front end of the injector spindle is shaped as an atomizing pintle with a ground top to atomize the injected fuel. When the current is removed, the combination of a spring and fuel back-pressure causes the needle valve to close [3].

The controlled voltage operating the fuel injector is a binary pulse train of on and off pulses. For the pulse train signal, the ratio of on time to the period of the pulse is called the duty cycle and the time interval between the leading edge and trailing edge of the pulse is known as the pulse width. Varying the pulse width determines how much fuel can flow through the injectors and the typical injector pulse width is between 1.5 and 10 milliseconds [9].

2.2.2 Ignition coil

The ignition coil is the heart of the ignition systems providing the high voltage required by the ignition system to fire the spark plugs which in turn ignites the air-fuel mixture in the cylinder completing the combustion process. The ignition coil serves as a high voltage transformer, stepping up the ignition system's primary voltage from 12 volts up to many thousands of volts. The key principle that makes transformers work is electromagnetic induction. The actual firing voltage needed to create a spark across the electrode gap of a spark plug depends on the width of the gap, the electrical resistance in the spark plug and plug wires, the air-fuel mixture, the load on the engine and the temperature of the spark plug [8]. The voltage required is constantly changing and varies from as little as 5,000 volts up to 25,000 volts or more. Figure 2.8 below illustrates an ignition coil.

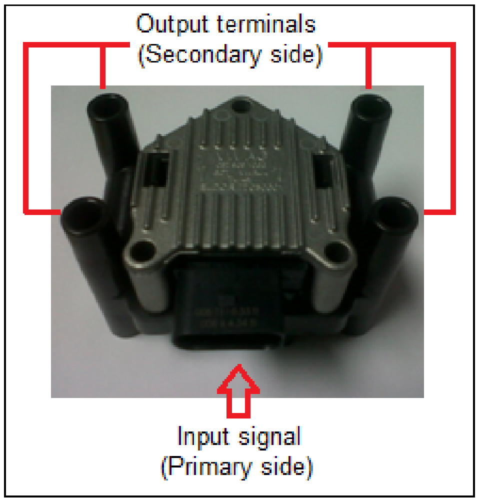


Figure 2.8: Ignition coil (distributor-less)

Older engines that have a distributor ignition system have a single coil and distributor-less ignition systems have multiple ignition coils. The circuit on the primary side of the coil has evolved from the basic contact breaker points and condenser to the distributor-less and coil per cylinder systems in common use today.

Inside every ignition coil are two sets of windings around a laminated iron core. The primary windings are a few hundred in number and the secondary windings have thousands of turns. The ratio of the secondary to primary windings is typically around 80 to 1, the higher the ratio the higher the potential output voltage of the coil [3]. High performance ignition coils typically have a higher ratio than factory fitted coils.

When the ignition module closes the coil’s primary circuit and provides a ground, current then flows through the primary windings. This creates a strong magnetic field around the iron core and charges up the coil. It takes about 10 to 15 milliseconds for the magnetic field to reach maximum strength [8]. The ignition module then opens the coil's ground connection and turns the primary coil windings off causing the magnetic field to suddenly collapse. The energy stored in the magnetic field has to go somewhere and so it induces a current in the coil's secondary windings. Depending on the ratio of turns of wire, this multiplies the voltage up to a 100 times or more until there is enough voltage to fire the spark plug. Although the number of turns within the coils primary is pre-set from manufacture, the strength of the magnetic field is proportionate to the current within the circuit and the speed of the switching of the ignition module.

With the technological advancement of ignition systems, soon after distributor-less ignition the coil-on-plug (COP) configuration of ignition coil was released. COP ignition is implemented on the Honda CBR600 engine used for this development system and a brief insight into this technology will now be carried out.

2.2.3 Coil-On-Plug (COP)

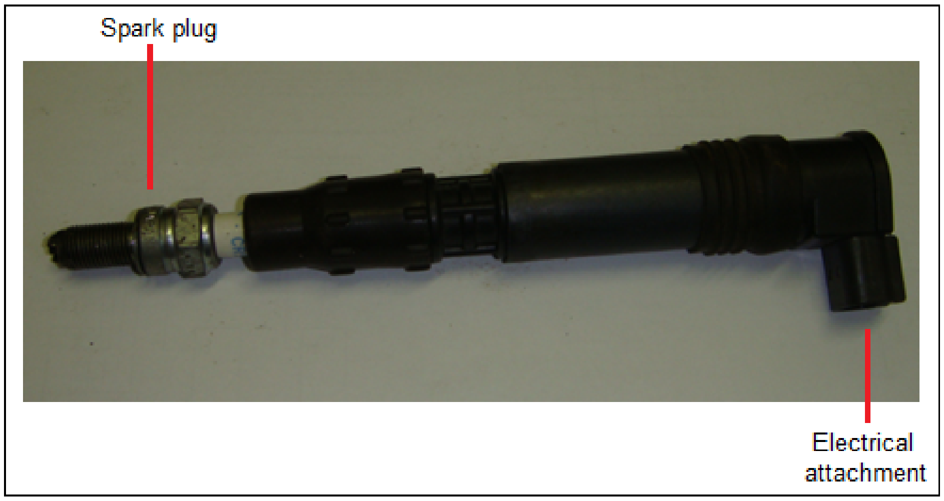


Figure 2.9: Coil-On-Plug (COP) ignition coil

The main difference between COP and other ignition systems is that each COP coil is mounted directly above the spark plug and so the voltage is directly delivered to the plug electrodes without having to pass through a distributor or high tension spark plug wires. It is a direct connection that delivers the hottest spark possible.

Vehicle manufacturers continuously aim to reduce costs and improve overall performance and reliability. High tension spark plug wires are often a source of ignition as well as vehicle assembly line problems. These plug wires must carry anywhere from 5 000 up to 40 000 or more volts to fire the plugs. This requires heavy insulation plus the ability to suppress electromagnetic interference. The wires must also be coated with a tough outer jacket to withstand high temperatures in the engine compartment and chemical attack. As reliable as today's plug wires, there is always the potential for trouble. Even the toughest insulation can burn if a wire rubs up against a hot exhaust manifold. The connection inside the spark plug boot between the wire and plug terminal can also be damaged if handled incorrectly when changing spark plugs. Plug wires can also radiate magnetic fields that may affect nearby sensor wires or other electronic circuits. Attaching the ignition coils directly to the spark plugs eliminates the need for separate high voltage wires along with their potential for trouble.

Eliminating the individual plug wires also eliminates the need for wire looms and heat shields and this is the reason why coil-on-plug ignition systems are being used on a growing number of newer model engines.

Along with the cost saving, improvement on the durability of the ignition system was noticed with COP. Using individual coils for each spark plug also means the coils have more charge time between each firing. Increasing the coil saturation time increases the coil output voltage at high rpm when misfire is most likely to occur under load. This also improves combustion and reduces the risk of misfire with lean air-fuel mixtures in which lean mixtures actually require more voltage to ignite
reliably [9]. The end result is fewer misfires, cleaner combustion and better fuel economy.

2.2.4 Spark plug

The spark plug is the electric component generating the spark to ignite the air-fuel mixture inside an internal combustion engine. As illustrated in figure 2.10 below, the spark plug comprises of a ceramic insulator forming a pottery support for the parts that conduct electricity, the terminal is where a current-conducting wire is attached, the spline is a hollow channel and the resistance is the device that controls the strength of the current. The ground electrode is a current device that unites the electrodes where the plug gap is the space separating the current conductors. The centre electrode is the central current conductor.

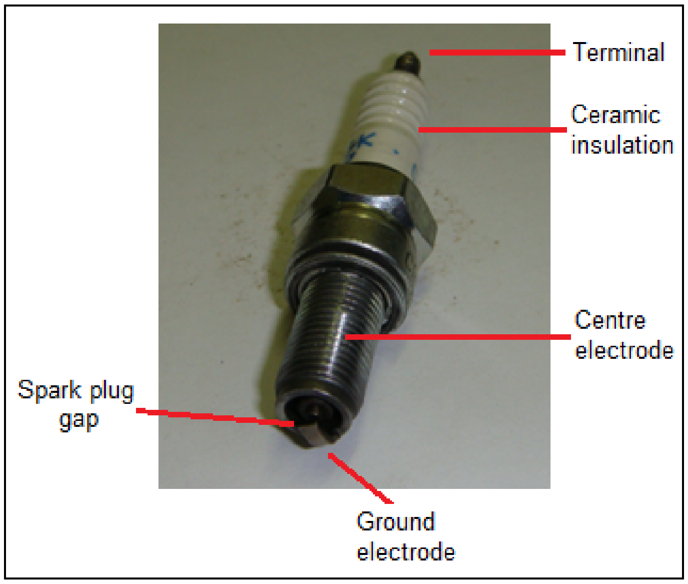


Figure 2.10: Spark plug

Spark plugs are mounted in the cylinder head where the end of the plug is sitting at the top of the cylinder. At the predetermined point of ignition when the intake valves have allowed the right amount of air-fuel mixture into the cylinder, the spark plug then ignites the mixture and creates combustion [3, 8]. The high voltage from the ignition coil is fed into the terminal at the top of the spark plug. It travels down through the core of the plug and arrives at the centre electrode at the bottom where it jumps to the ground electrode creating a spark. The ceramic insulator basically keeps the high-tension charge away from the cylinder head so that the spark plug does not ground before it generates the spark.

The main aspects that affect a good delivery of the actual spark on the spark plug are the electrode material and configuration, spark plug gap and the heat range. In the mid 1980s spark plug manufacturers started making plugs with copper core centre electrodes. Copper being an excellent conductor of heat, allows plugs to run hotter without causing preignition - a form of abnormal combustion. This improved plug fouling resistance, ignition reliability and the plug life. It also reduces the number of plugs needed to cover a range of engine applications because each plug has a broader heat range. In 1985 a big improvement in spark plug technology came when the first generation long life plugs with platinum or gold-palladium electrodes were developed and with this electrode wear was greatly reduced. Long-life spark plugs drastically reduce the need for maintenance while helping the engine maintain like-new performance and emissions. Spark plugs with platinum or platinum-tipped electrodes use a conventional electrode configuration with a small platinum plug welded to the tip of one or both electrodes. Increasing the number of side electrodes gives the spark more paths to ground and reduces the risk of a misfire.

A spark jumps more easily from a sharp edge than a rounded blunt edge and so the sharper the edges it has to jump to, the better the possibility of the plug firing under all types of driving conditions.

Spark plug gapping is the process of ensuring the gap between the two electrodes is correct for the specific type of engine. With a large gap the spark will be weak and with a smaller gap the spark could possibly jump across the gap too early.

The spark plug "heat range" refers to the relative temperature of the tip of the spark plug when it is working. The term actually refers to the thermal characteristics of the plug itself, specifically its ability to dissipate heat into the cooling system. A cold plug can get rid of heat very quickly and should be used in engines that run hot and a lean air-fuel mixture [2]. A hot plug takes longer to cool down and should be used in lower compression engines where the heat needs to be retained to prevent combustion by-product build-up.

2.3 Engine test facility

Every ems has to be initially calibrated and tuned to the specific engine configuration. To enable full potential of any engine the system must be initially setup on a dynamometer (dyno). The dyno essentially places a load on the engine and measures the amount of power that the engine can produce against the load set. The most common types of dyno's that are available are the engine and rolling road dyno's. The NMMU has an engine test facility with two engine dyno cells, specifically the eddy current type as illustrated in figure 2.11 below.

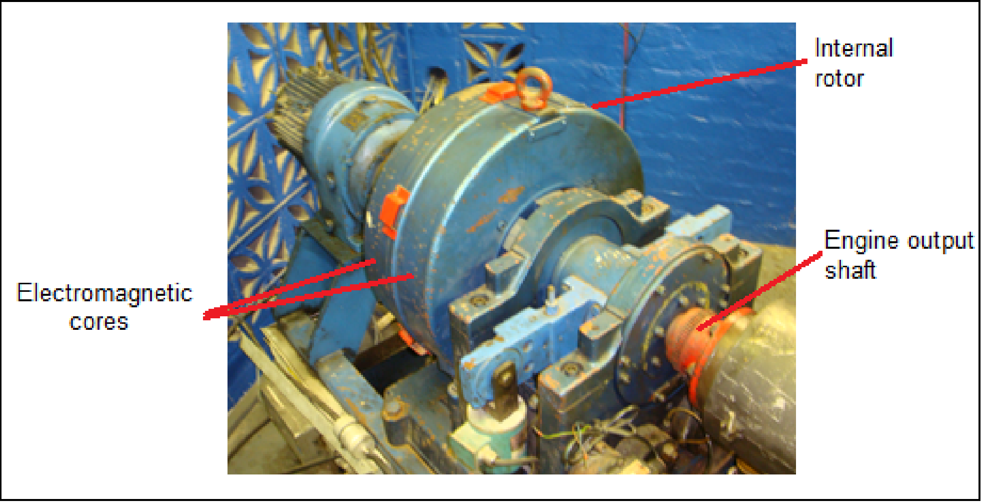


Figure 2.11: Eddy current engine dynamometer

To test engine performance in the thermodynamic laboratory at the NMMU, the engine is coupled to an engine dyno which is connected to the engine output shaft. The eddy current dyno has an electrically conductive disc rotor moving across a magnetic field to produce resistance to movement. The variable electromagnets change the magnetic field strength to control the amount of braking. The electromagnet voltage is controlled by the computer using changes in the magnetic field to match the power output being applied. As the operator increases the current supply to the coils, the rotor becomes harder for the test engine to turn. The braking resistance is continued until the engine's maximum turning force is measured at the desired engine speed. The operator can then obtain accurate readings of the engines torque and then use specific calculations to derive the power output for the engine. Various atmospheric conditions are factored into the calculations. The eddy current brake advantage is its fast response to the computers loading instructions. The overall principle of the dyno is to measure and compare power transfer at different points on the engine allowing the engine controlling parameters to be altered to get the desired output form the engine – power, fuel efficient, emissions control.

2.3.1 Key Measurements

For the purpose of this research the important performance measurements are output power, torque and speed. The engine's brake power (output power) is determined from the engine speed and torque.

Speed is measured with a pickup that monitors the shaft spinning off the dyno. Typically, a toothed steel wheel is placed on the shaft and an inductive sensor is placed close enough to the wheel such that it generates pulses as the teeth pass by. This time-based pulse signal is analysed in order to measure the period between the pulses and then converted to frequency. Shaft speed is directly proportional to the measured frequency but since the output shaft is coupled to the gearbox the dyno speed is multiplied by the gearing ratio to determine the engine speed.

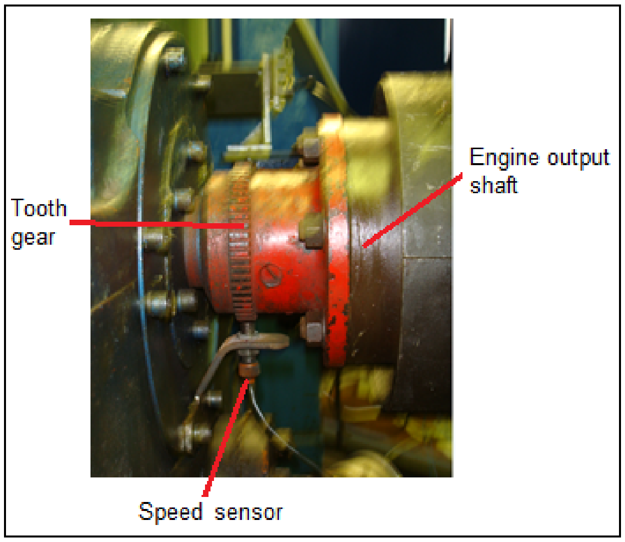


Figure 2.12: Dynamometer speed sensor

The torque is measured by the load cell which is a force transducer. The load cell strains slightly with applied load and the strain is measured using strain gauges connected to a preamplifier that outputs a voltage proportional to the load. According to Newton’s first law, the measured torque on the dyno housing equals to the torque on the dyno shaft.

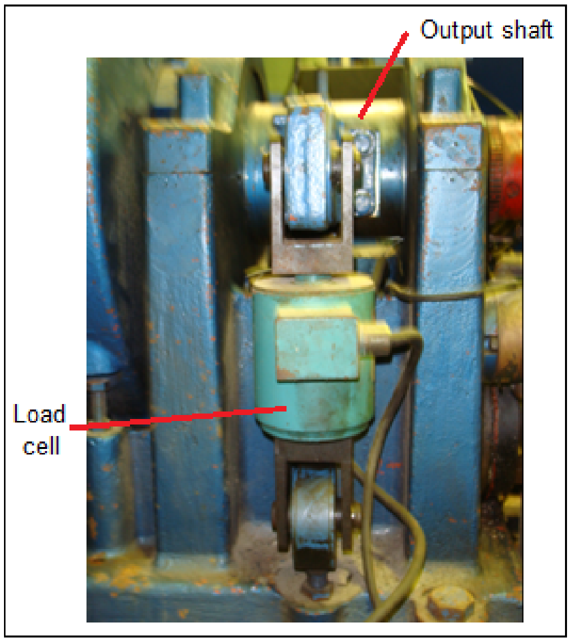


Figure 2.13: Dynamometer load cell

The throttle position is another useful parameter to measure. The TPS is basically a potentiometer configured as a voltage divider that rotates with the throttle. A stepper motor is connected to the throttle linkage allowing control of the throttle position.

A typical test might first set the dyno controller to a specific speed and then measure torque variation as a function of throttle setting. Alternatively, the throttle may remain fixed (for example, wide open) and the speed control set point varied. The resulting speed and torque values are measured, and power is calculated.

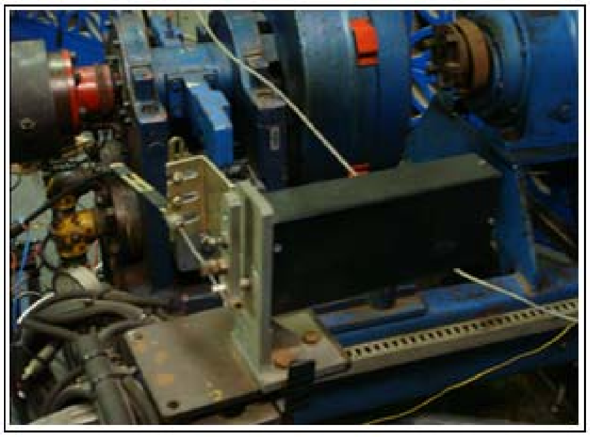


Figure 2.14: Dynamometer throttle control

During the testing required by this research the important parameter to monitor is lambda. This provides a measure of how efficiently the engine is running based on the air-to-fuel ratio in the exhaust gas. The temperature measurements are useful in monitoring if the engine is being operated safely or is overheated.

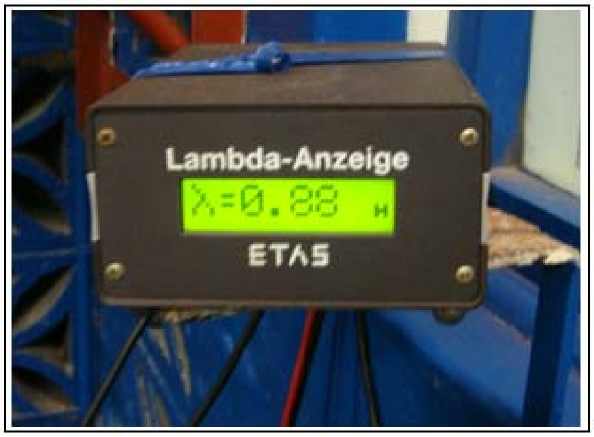


Figure 2.15: Lambda measurement display

2.3.2 The measurement system

Given the volume of data and the number of real-time calculations required a computer-controlled data acquisition (DAQ) system is used to calculate, display, and archive test data from tests. These devices are very helpful in engine testing that require measurements of speed, torque, throttle position, and temperature of cylinder and exhaust. The speed, torque, and throttle position signals are measured using the analog input ports. Shaft speed is calculated from the period of the sinusoidal speed. Torque is the product of the spring load or weight and the distance from the axis of rotation. Dyno's also measure the torque produced by an engine in order to reveal important information about its performance. Figure 2.16 below illustrates the DAQ hardware in the NMMU dyno cell and the user interface on the computer.

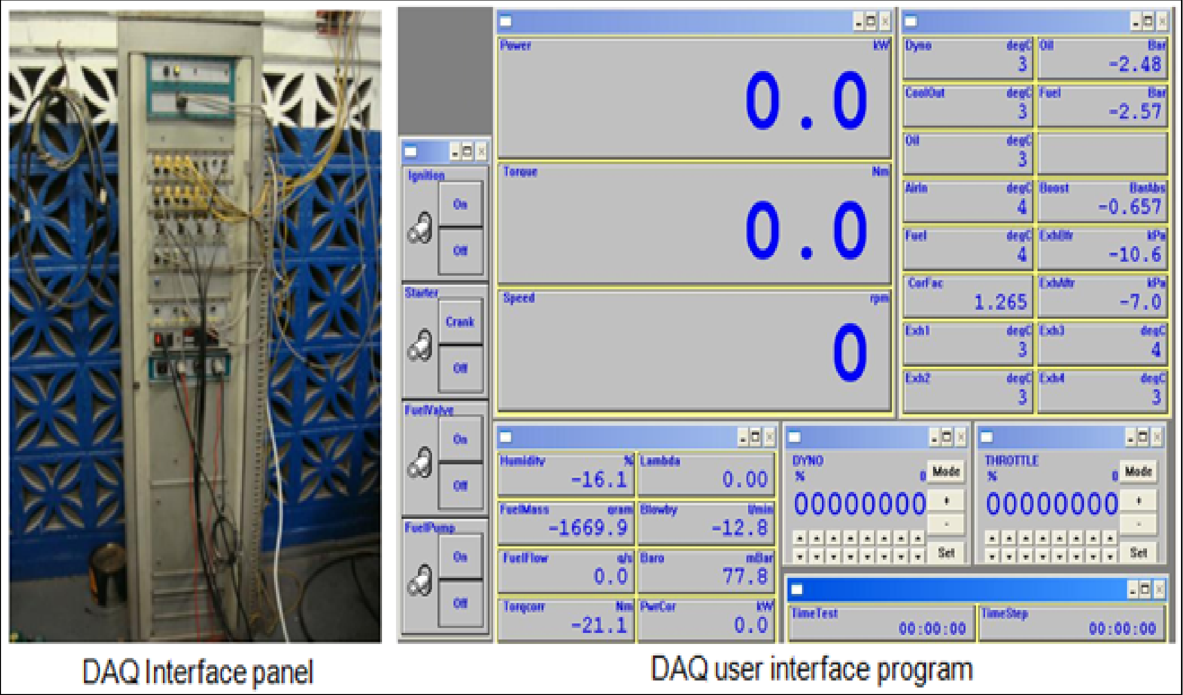
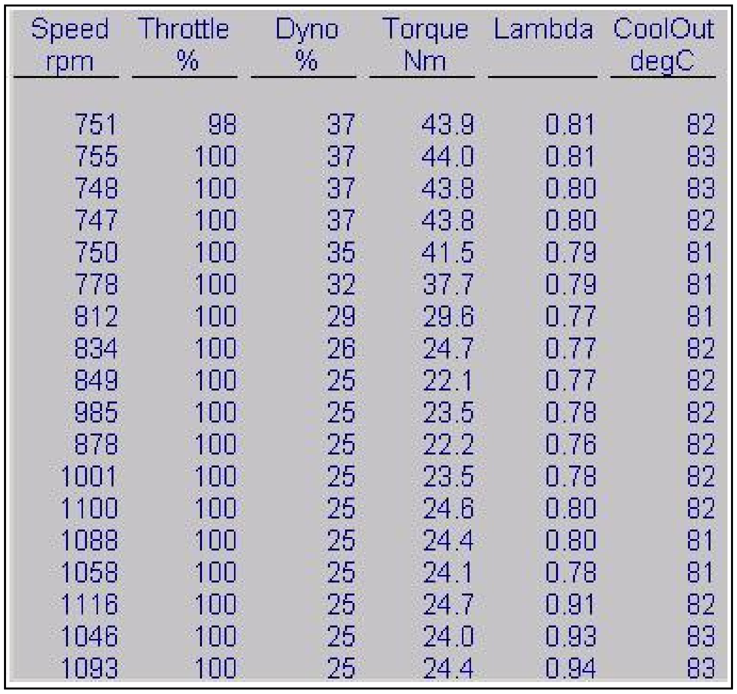


Figure 2.16: Data acquisition system

2.3.3 The Testing Sequence

A typical test will consist of running the engine at a number of speeds and loads under varied engine conditions. For example, when investigating the effect of ems tuning, it is important to test at both low and high engine speeds as well as low and high throttle openings.



Speed rpm	Throttle %	Dyno %	Torque Nm	Lambda	CoolOut degC
751	98	37	43.9	0.81	82
755	100	37	44.0	0.81	83
748	100	37	43.8	0.80	83
747	100	37	43.8	0.80	82
750	100	35	41.5	0.79	81
778	100	32	37.7	0.79	81
812	100	29	29.6	0.77	81
834	100	26	24.7	0.77	82
849	100	25	22.1	0.77	82
985	100	25	23.5	0.78	82
878	100	25	22.2	0.76	82
1001	100	25	23.5	0.78	82
1100	100	25	24.6	0.80	82
1088	100	25	24.4	0.80	81
1058	100	25	24.1	0.78	81
1116	100	25	24.7	0.91	82
1046	100	25	24.0	0.93	83
1093	100	25	24.4	0.94	83

Figure 2.17: DAQ User Interface

At each test point, the system software sets the dynamometer speed and then opens the throttle until the desired torque is achieved as illustrated in Figure 2.17 above. The engine is then operated at the desired condition long enough to consume a significant amount of fuel. This generally takes one or two minutes, which is enough time to obtain results from exhaust-gas analysis. Once the test point is finished, the system proceeds to the next test point. Once all of the test points have been run, ems tuning may be adjusted and a new test performed using the entire set of test points. Trends in fuel consumption, lambda, or engine temperature can then be seen as a function of the ems settings**,** making it possible to determine the optimal settings.

Chapter 3 – Experimental setup

The foundation of the project revolves on the type of engine that the control system will need to work on. Various engine manufacturers use different types of sensors, control systems and actuators and so once a specific engine has been identified then the following design steps can take place. Research was carried out to find out what was the most widely used engine in the formula student competition and the conclusion was that the Honda CBR600 engine was best suited for this project as they had good overall performance as well as its proven reliability [10]. The engine configuration was studied and the appropriate development was carried out to control the fuel and ignition timing parameters.

3.1 Engine hardware setup

In the early parts of the research a lot of time had been spent on investigating various exiting engine management systems on their hardware components and overall operation. The important aspect is to have clear filtered signals into the microprocessor so that accurate calculations can be carried out and the corresponding outputs signals can be in synchronization with what is required by the engine for optimal performance. Various filtering circuits were tried and tested for the range of voltages and frequencies of the magnetic sensors. The Honda engine used in this project has magnetic sensors for both the crankshaft sensor and camshaft sensor.

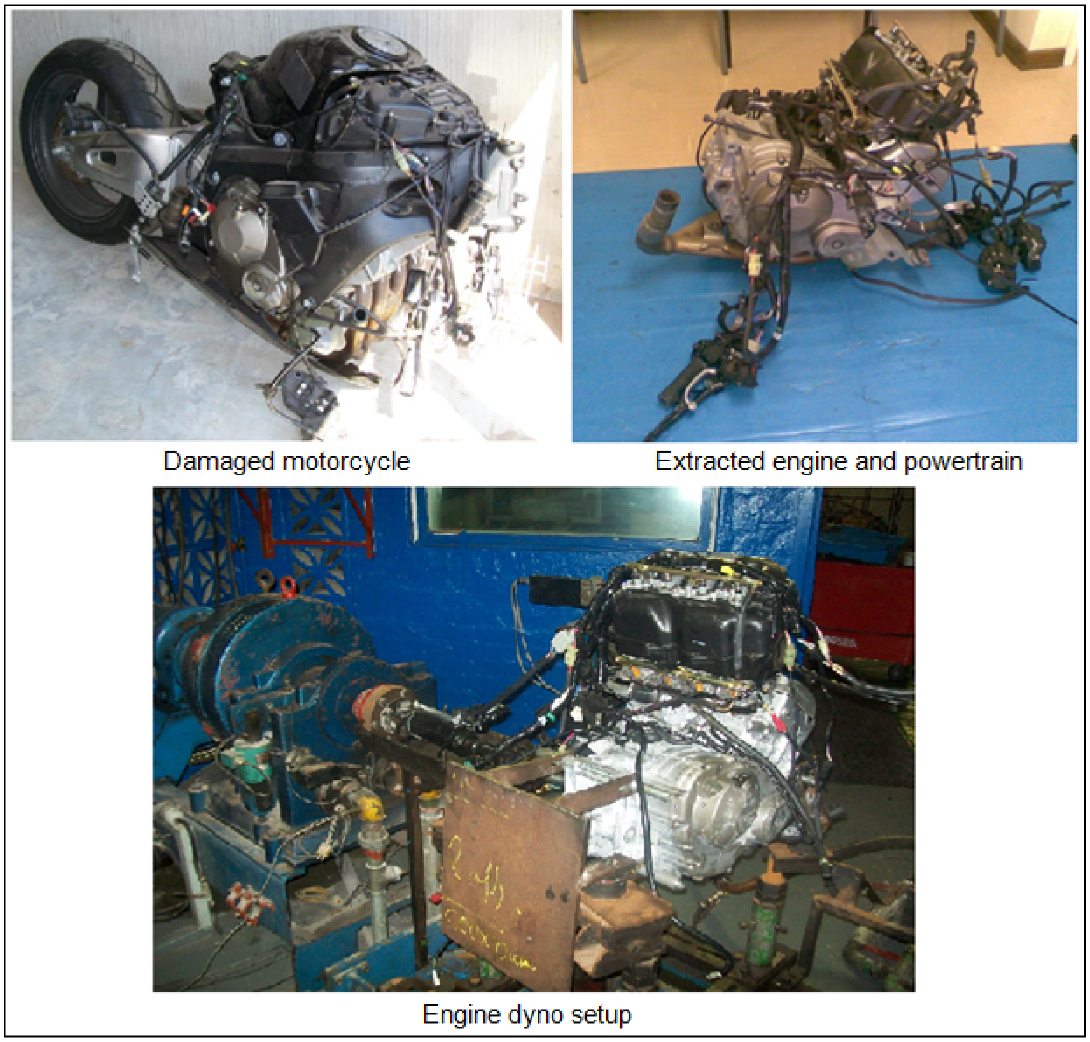


Figure 3.1 Honda CBR 600 engine

An accident damaged motorcycle was purchased and the engine disassembled from the main frame and specific brackets were made for the engine to be adapted and mounted onto the engine dyno. The existing ems and electronics were then coupled up to the engine. There was some difficulty in getting the engine started as there are many safety features on motorbike, some of which include a bank angle sensor which switches off the engine when the bike exceeds a tilt angle of 45 degrees or greater. All these features had to be examined and then by-passed in order to get the engine started. The start-up of the engine was important to make sure that the mechanical operation of the engine is fully functional.

The existing Honda ems uses both crankshaft and camshaft sensors for control of the engine. These sensors and sensor triggers were removed from the engine to investigate how the factory ems controls the fuelling and timing of the engine. At first the technical documentation of the engine was extracted from the workshop manual as illustrated in Table 3.1 below.

Engine Configuration	Inline Four, 4-Stroke
Engine Displacement	599 cc
Engine Cooling System	Liquid
Compression Ratio	12.0:1
Combustion Chamber Design	Pentroof
Valves Per Cylinder	4
Intake Valves Per Cylinder	2
Exhaust Valves Per Cylinder	2
Bore x Stroke	67.0 x 42.5 mm
Combustion Chamber Volume	8.0 cc
Intake Valve Diameter	27.5 mm
Exhaust Valve Diameter	23.0 mm
Intake Valve Stem Diameter	4.0 mm
Exhaust Valve Stem Diameter	4.0 mm
Intake Valve Maximum Lift	8.3 mm
Exhaust Valve Maximum Lift	7.2 mm
Intake Valve Timing	
Open BTDC	22 degrees
Close ABDC	43 degrees
Duration	245 degrees
Exhaust Valve Timing	
Open BBDC	40 degrees
Firing order	1-2-4-3
Primary Drive	Gear
Primary Drive Gear Teeth (Ratio)	76/36 (2.111:1)
Final Drive Sprocket Teeth (Ratio)	46/16 (2.688:1)
Transmission Gear Teeth (Ratios)	
6th	28/24 (1.166:1)
5th	29/23 (1.261:1)
4th	31/22 (1.409:1)
3rd	29/18 (1.611:1)
2nd	31/16 (1.937:1)
1st	32/12 (2.666:1)

Table 3.1 Honda CBR 600 Engine Specification

There is no particular information related to the ems control except for the firing order of the cylinders which is 1 – 2 – 4 – 3. The information for the exact control of the fuel injectors and ignition coils is confidential and so this type of information is not easily available and will not be released under any request to Honda. An investigation into the trigger patterns of the factory crankshaft and camshaft trigger patterns was carried out where specific degree wheels were drawn up in CAD and then printed and mounted on the trigger discs.

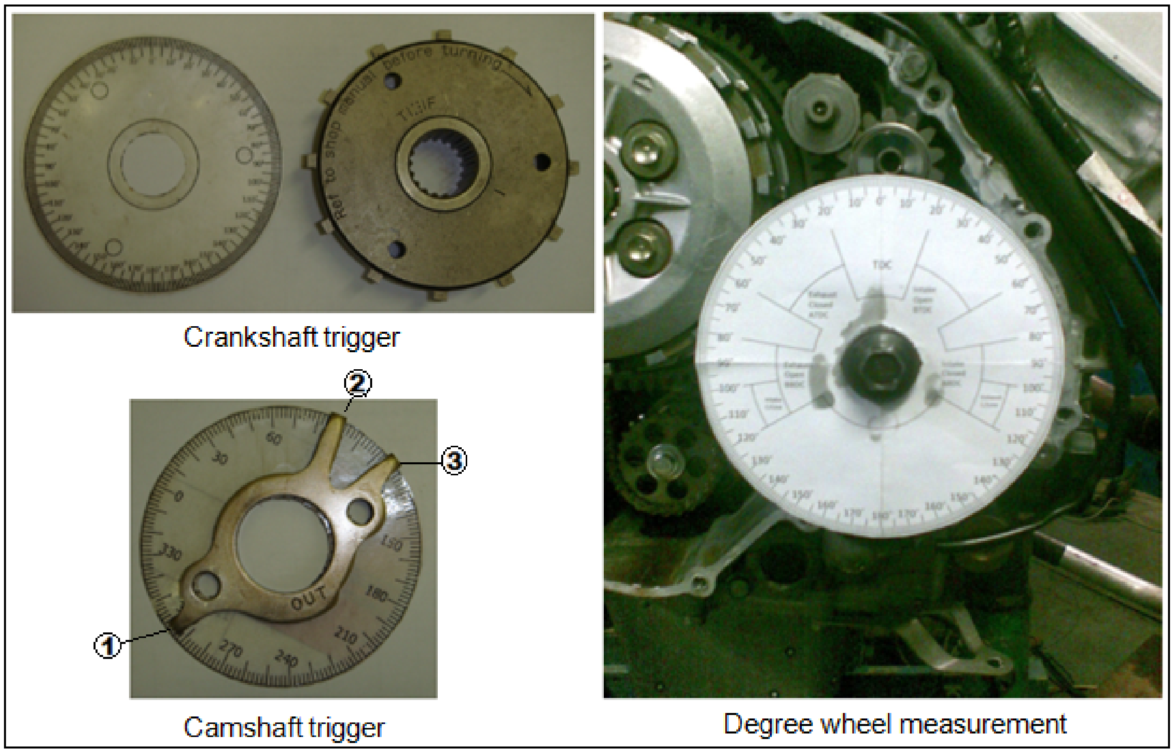


Figure 3.2 Honda CBR 600 crankshaft and camshaft sensor triggers

The results concluded that the crankshaft trigger disc has 12 teeth, each at 30 degrees apart.

The camshaft trigger disc has 3 teeth, and the analysis from starting with cylinder 1 at TDC (power stroke) was as follows,
Trigger 1 takes place after 270 degrees of crankshaft rotation, all pistons at the same level with cylinder 3 in the power stroke.

Trigger 2 takes place after 570 degrees of crankshaft rotation, with cylinder 3 at TDC.

Trigger 3 takes place after 630 degrees of crankshaft rotation, all pistons at the same level with cylinder 2 in the intake stroke. With no further information available from Honda on the model behind these trigger patterns it can only be assumed that the crank sensor trigger is used for engine speed since the conventional multiple teeth will not work under the high operational speeds of these engines. The multi-tooth trigger wheels would end up giving a continuous high level signal at high engine speeds which would not be useful for any calculations by the ems.

The cam sensor trigger is assumed to be for cylinder detection for the ems to know the specific position of the cylinders for sequential injection and ignition. There could be corresponding trigger patterns between the crank and cam triggers that could be used for certain information as well.

Figure 3.3 below shows the waveform pattern for a 60-2 crankshaft trigger wheel for a Bosch ecu. The term 60-2 (sixty minus two) refers to a trigger wheel with 58 teeth and two missing teeth. The missing tooth acts as a reference for cylinder 1 recognition.

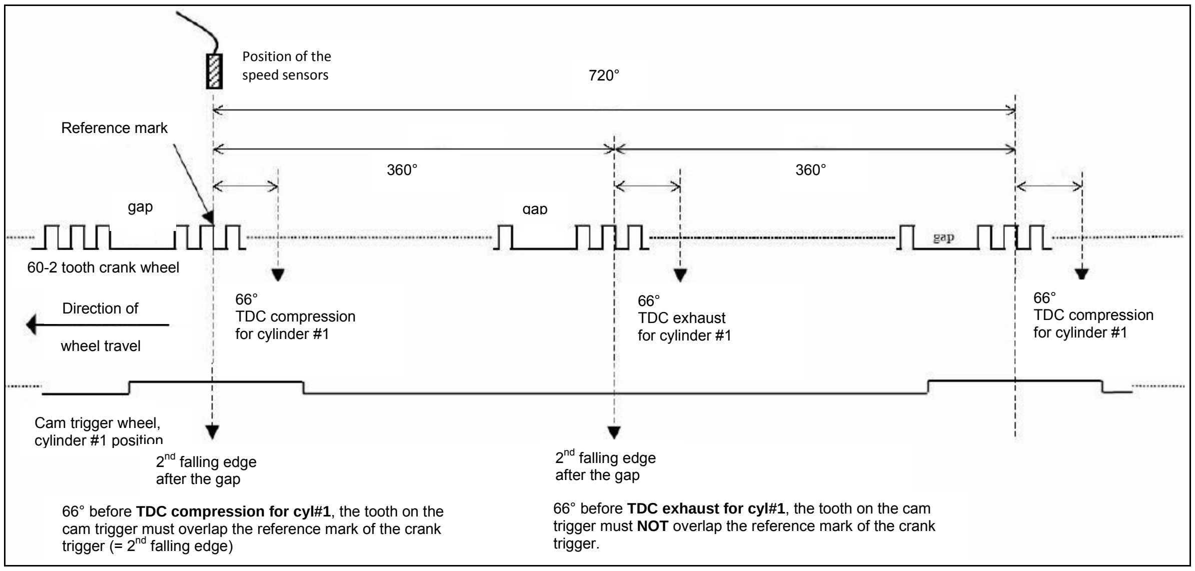

Position of the speed sensors
720°
Reference mark
360°
360°
gap
gap
gap
60-2 tooth crank wheel
Direction of wheel travel
66° TDC compression for cylinder #1
66° TDC exhaust for cylinder #1
66° TDC compression for cylinder #1
Cam trigger wheel, cylinder #1 position
2nd falling edge after the gap
2nd falling edge after the gap
66° before TDC compression for cyl#1, the tooth on the cam trigger must overlap the reference mark of the crank trigger (= 2nd falling edge)
66° before TDC exhaust for cyl#1, the tooth on the cam trigger must NOT overlap the reference mark of the crank trigger.

Figure 3.3 Bosch engine control unit signal reference for 60-2 trigger wheel [13]

Any further interpretation on the result obtained from the Honda engine was not possible as there was no access to any of the Honda ems information. No model could be really drawn up to explain the operation of the factory control of the fuel and ignition delivery into the engine which then concluded that specific trigger disks had to be developed for this project. In analysing the valve timing system from the extract of the technical document in Table 3.1, the following waveforms were developed for a complete combustion cycle of one cylinder.

Intake valve:		**Exhaust valve:**	
Opens at 1 mm lift	22° BTDC	Opens at 1 mm lift	40° BBDC
Closes at 1 mm lift	43° ABDC	Closes at 1 mm lift	5° ATDC
Duration	245°	Duration	225°

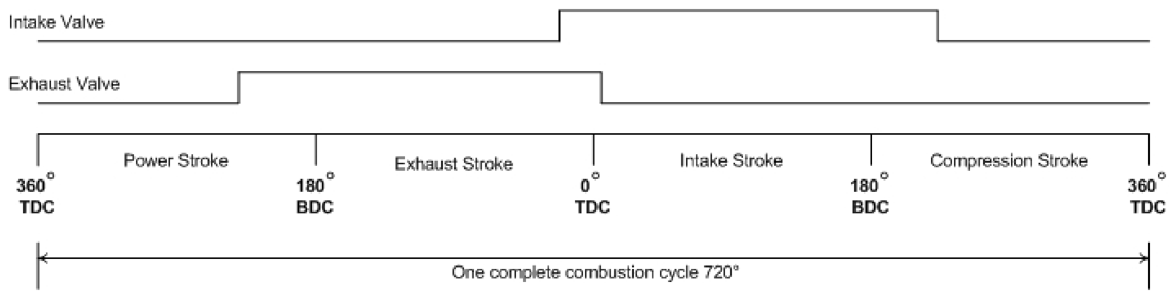


Figure 3.4 Valve timing diagram for one combustion cycle

Different types of ems configurations require different input triggers and at the same time give different output controlling parameters. For injection there is batch (simultaneous) injection, grouped injection and sequential injection, which is more common on the newer model engines. Batch injection is where the fuel is injected into all cylinders at the same time and only enters the combustion chamber when the specific cylinder’s intake valve opens. Grouped injection is when two injectors are operated at the same time. With sequential injection the fuel is injected into a specific cylinder at specific times of the opening of the intake valve, each injector is staged to inject at different times of the crankshaft rotation. The different injection timing gives different output responses from the engine, this however requires a lot more trigger inputs from a hardware side as well as time when setting up and tuning the engine.

A vast majority of ems's open and close the fuel injectors before the intake valve actually opens. Atomized fuel is then drawn into combustion chamber with the intake of air. If the injector opens when the intake valve is open, uneven mixture distribution can occur in the combustion chamber.

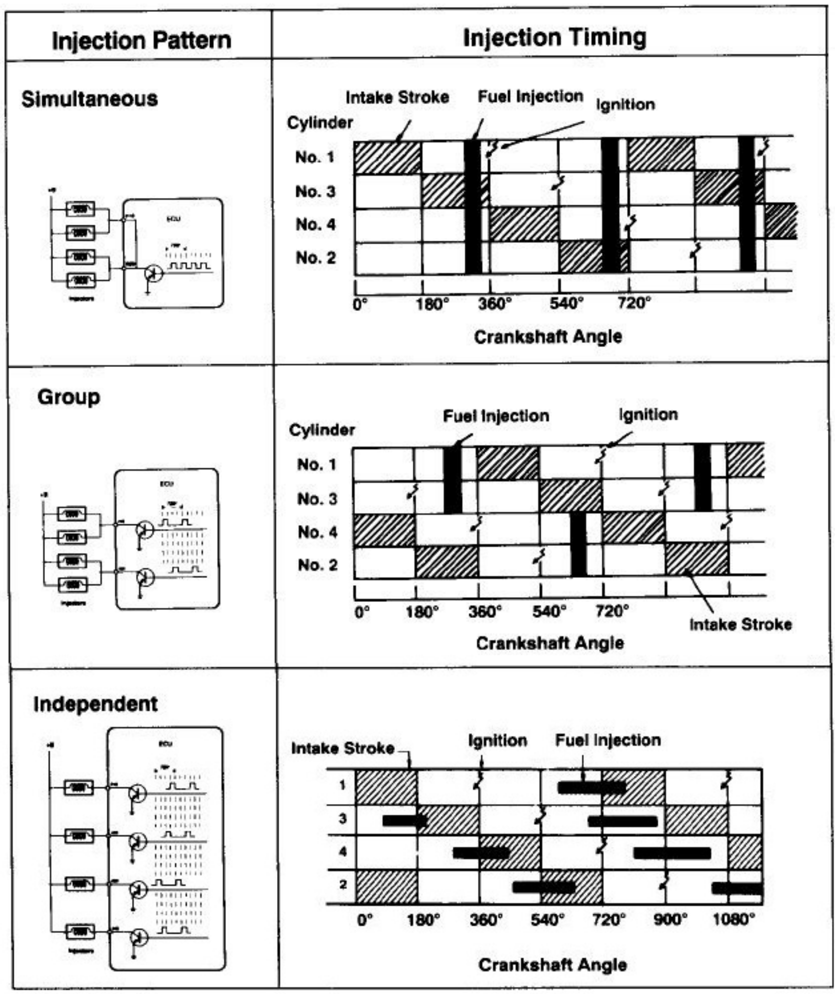


Figure 3.5 Toyota fuel Injection pattern and ignition pattern [17]

Figure 3.5 above explains Toyota's method of injection relative to crankshaft angle. It can be noted that their injection in the simultaneous pattern takes places every 360°. Although the fuel is delivered to some cylinders while the intake valve is closed, the engine will still use up the fuel once the intake valve opens, this happens so fast with higher engine speeds that the fuel does not always settle.

“Tau” is the term used to describe wall wetting where there is settling of the fuel on port walls before intake valve is open opened [4]. On the independent/ sequential pattern, injection takes place before-top-dead-centre (BTDC) on the exhaust stroke during the valve overlap.

Most engines have a certain amount of valve overlap time where the exhaust and intake valve are both open at the same time, with the exhaust on its closing cycle and the intake on its opening cycle. This then concludes that injection can take place as the intake valve is opening or has just opened.

For engines with coil-on-plug ignition systems there is wasted spark ignition and sequential ignition. Just like on the injection side, the sequential ignition requires more trigger inputs and well as tuning time to find the optimal timing of ignition for improved results. The wasted spark system produces two sparks at the same time, one spark on compression and one on the exhaust stroke of each cylinder. It is referred to as wasted because only the spark on the compression stroke is useful, the spark on the exhaust stroke is wasted with no combustion taking place. This type of ignition also allows the cylinder in the exhaust stroke to be cleaned out of any unburned fuel/air mixture from the previous combustion cycle and thus allowing for a clean induction of mixture on the next cycle. It is found that less voltage is needed to fire the ‘wasted spark’ cylinder as there is no compression. The wasted spark ignition systems are commonly found on motorcycle engines.

In this project batch injection and wasted spark ignition will be used. The trigger inputs are critical for the microprocessor to be able to receive the signal, do all the necessary calculations and then carry out the required output task within the specific time.

Table 3.2 below shows the theoretical calculation for the time duration available for different engine speeds. In a typical 4 cylinder ignition system there would be 2 input pulses per revolution.

RPM	RPS	Frequency Ignition pulses (4cyl)	Period (sec)	msec
1000	16.667	16.667	0.0600	60.00
2000	33.333	33.333	0.0300	30.00
3000	50.000	50.000	0.0200	20.00
4000	66.667	66.667	0.0150	15.00
5000	83.333	83.333	0.0120	12.00
6000	100.000	100.000	0.0100	10.00
7000	116.667	116.667	0.0086	8.57
8000	133.333	133.333	0.0075	7.50
9000	150.000	150.000	0.0067	6.67
10000	166.667	166.667	0.0060	6.00
11000	183.333	183.333	0.0055	5.45
12000	200.000	200.000	0.0050	5.00

Table 3.2 Engine speed calculations

It can be noted that as the engine speed increases there is less time available to carry out the necessary tasks by the microprocessor. In analysing some existing ems on the market there is a common trend for crankshaft trigger references to be between 90° and 60° BTDC, this varying between manufacturers. For the selected type of ignition system in this project the microprocessor would require input signals for each cylinders top-dead-centre (TDC) position. This is not as simple as there needs to be some form of identification of the specific cylinder to get the right ignition timing delivered to the correct cylinder at the correct time. The configuration of the Honda engine crankshaft is as per the normal 4 cylinder configuration where cylinder 1 and 4 follow the same movement but in different cycles of combustion, and the same pairing for cylinder 2 and 3 in different combustion cycles.

The decision was then made to use the crankshaft trigger for cylinder 1 and 4 recognition, and the camshaft trigger for cylinder 2 and 3 recognition. In the wasted spark ignition system the pairing of the firing cylinders would be cylinder 1 and 4 together and cylinder 2 and 3 together.

For ease of mechanical work in developing a new crankshaft trigger, the existing 12 tooth Honda crankshaft trigger wheel was modified to suit the application of this project. This trigger disc has 12 equally spaced teeth over 360° which means one tooth every 30°. It would be possible to use the pattern of 60° BTDC trigger for cylinder reference as the 60° time would allow the microprocessor enough time to carry out the necessary calculations based on the input triggers received. A worst case scenario analysis was carried out in a spreadsheet to calculate the available time at maximum rpm and the 60° allowance BTDC proved sufficient enough.
Figure 3.6 below illustrates the exact relation of the reference to the combustion cycle.

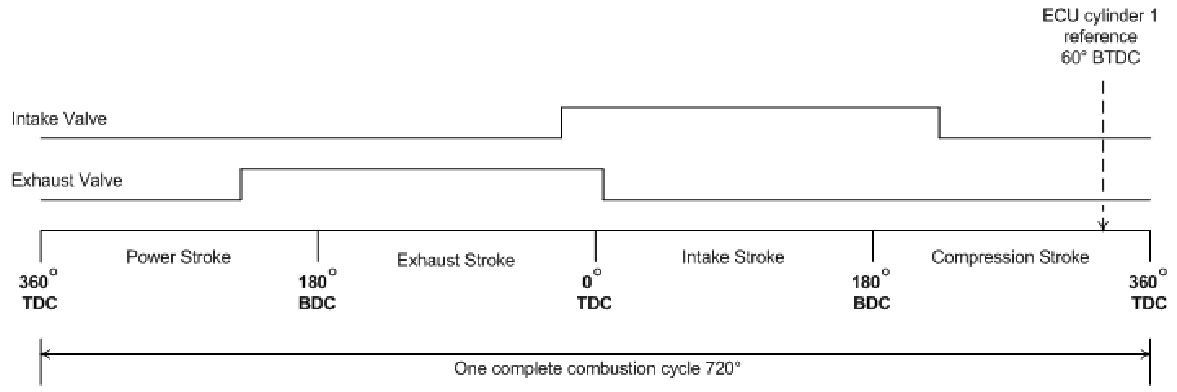


Figure 3.6 Timing diagram for one combustion cycle

The concept then followed that it was required to have two triggers on the crankshaft trigger, one for cylinder 1 and one for cylinder 4. However this proved not to be the case during the testing on the dyno as there were double ignition triggers taking place when the timing light was used to verify the actual ignition timing on the engine.

The problem came about in being misled that the crankshaft needed two triggers, this was not the case because in one complete combustion cycle the crankshaft would rotate through two revolutions and the camshaft through one revolution.
Since the crankshaft would be rotating twice in one combustion cycle, the first trigger would represent the first cylinder of the pair (1 and 4) at 60° BTDC and then on the next 360° rotation the opposite cylinder of the pair would be at 60° BTDC.

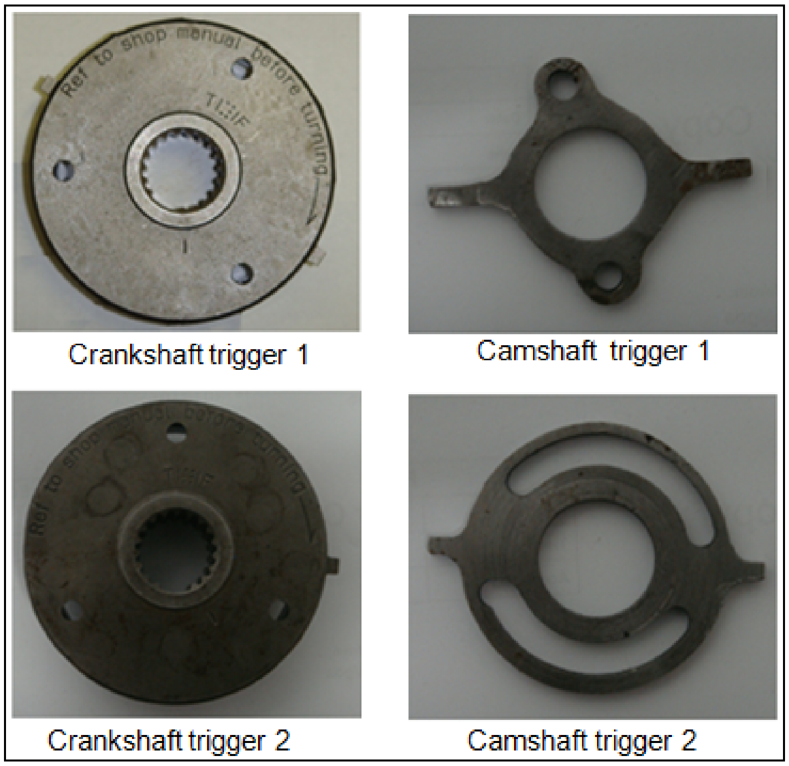


Figure 3.7 Crankshaft and Camshaft triggers used in this project

It was not possible to modify the existing camshaft trigger and so a custom design was implemented. The first prototype that was made proved some difficulty in synchronizing the timing of cylinder 1 and 4 to cylinder 2 and 3's ignition firing. The later improvement of the design had a slotted section to allow adjustment of the trigger angle to synchronize the cylinder firing as illustrated in Figure 3.7 above. The ignition timing light was used to verify each bank of ignition timing. The two triggers on the cam trigger represented cylinder 2 and 3 60° BTDC, this being 180° out of phase to the crankshaft trigger.

3.2 Electronic Hardware development

The sine wave output signals from the crankshaft and camshaft sensors need to be converted into a square wave to be interpreted by the microcontroller. There are various circuits that are capable of doing the signal conversion but the problem is that the sine wave from the magnetic sensors has varying amplitude with the varying frequency. A few circuit configurations were tried, discrete components and integrated components, and further research led to a stable circuit using operational amplifiers.

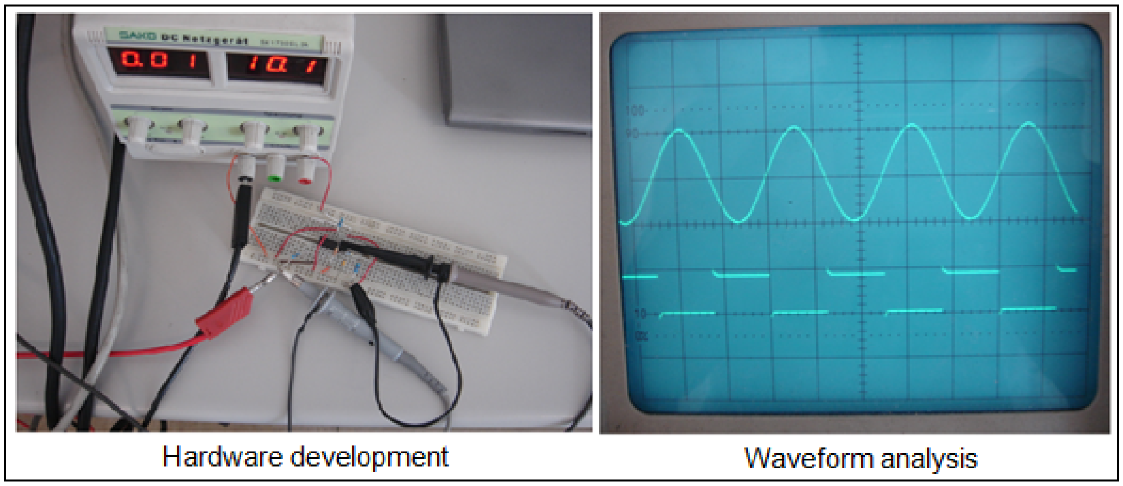


Figure 3.8 Crankshaft sensor circuit development

Figure 3.8 above illustrates the first generation of the signal conversion circuit. This section of the project developments described here was carried out in Wolfsburg, Germany as part of the exposure to the formula student project and access to dedicated facilities of the project. The circuit above included the LM1815 integrated circuit which is purposely manufactured for magnetic sensor signal conversion but the problem was the transition time delay between input and output signals. This is not very suited for automotive ignition applications as any delay in the signal transitions affects the ignition timing and can affect the performance of the engine.

A latter design shown in Figure 3.9 incorporating filtering components was built which provided a stable input signal into the microcontroller. The design is capable of converting sine wave signals within the operating band of frequencies and voltages of the magnetic sensors. The non-inverting operational amplifier operates on a single power supply eliminating the need for the negative power supply, which is common on most operational amplifiers. Zener diodes were used to isolate the high voltages associated with high engine speeds. The gain of the operational amplifier was biased with the resistor configuration to allow the small enough signals during engine cranking to be able to be recognized by the microcontroller.

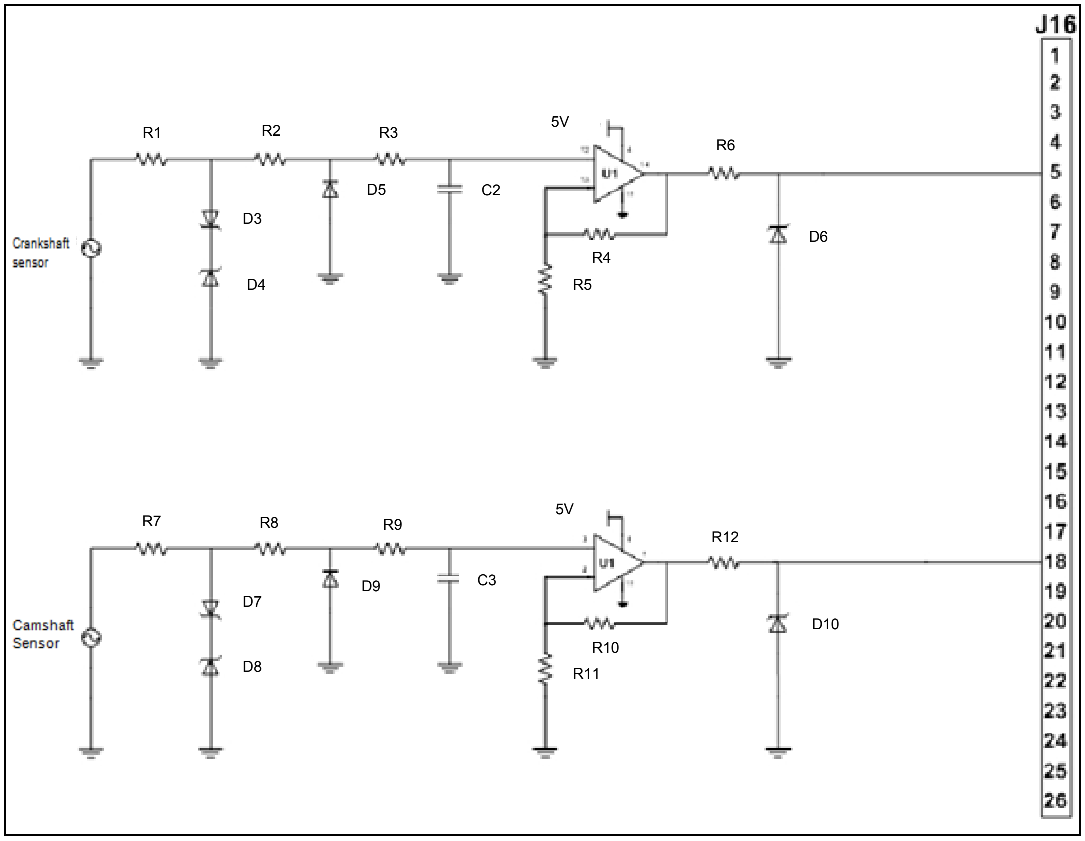


Figure 3.9 Signal conditioning circuit for crankshaft and camshaft sensors

The main heart of all processing of the input signals, calculations and output signals is done by a PIC 24FJ128GA010 microcontroller which features a high performance 100 pin controller operating at 32Hhz oscillator speed.

The demo board developed using this microcontroller was done in-house at the NMMU which provides an extended scope of applications to any project [23].

Output signals from the microcontroller need to be amplified in order to drive the ignition coils which provide the spark to ignite the air and fuel mixture in the combustion chamber of the engine. The important criteria for the selection of these components would be the high current delivery and nominal power dissipation of the heat. Earlier research into existing ignition systems provided a good selection of components for an application suited to this project as well as compatibility with the PIC microcontroller. Figure 3.10 below illustrates the injector driver circuit.

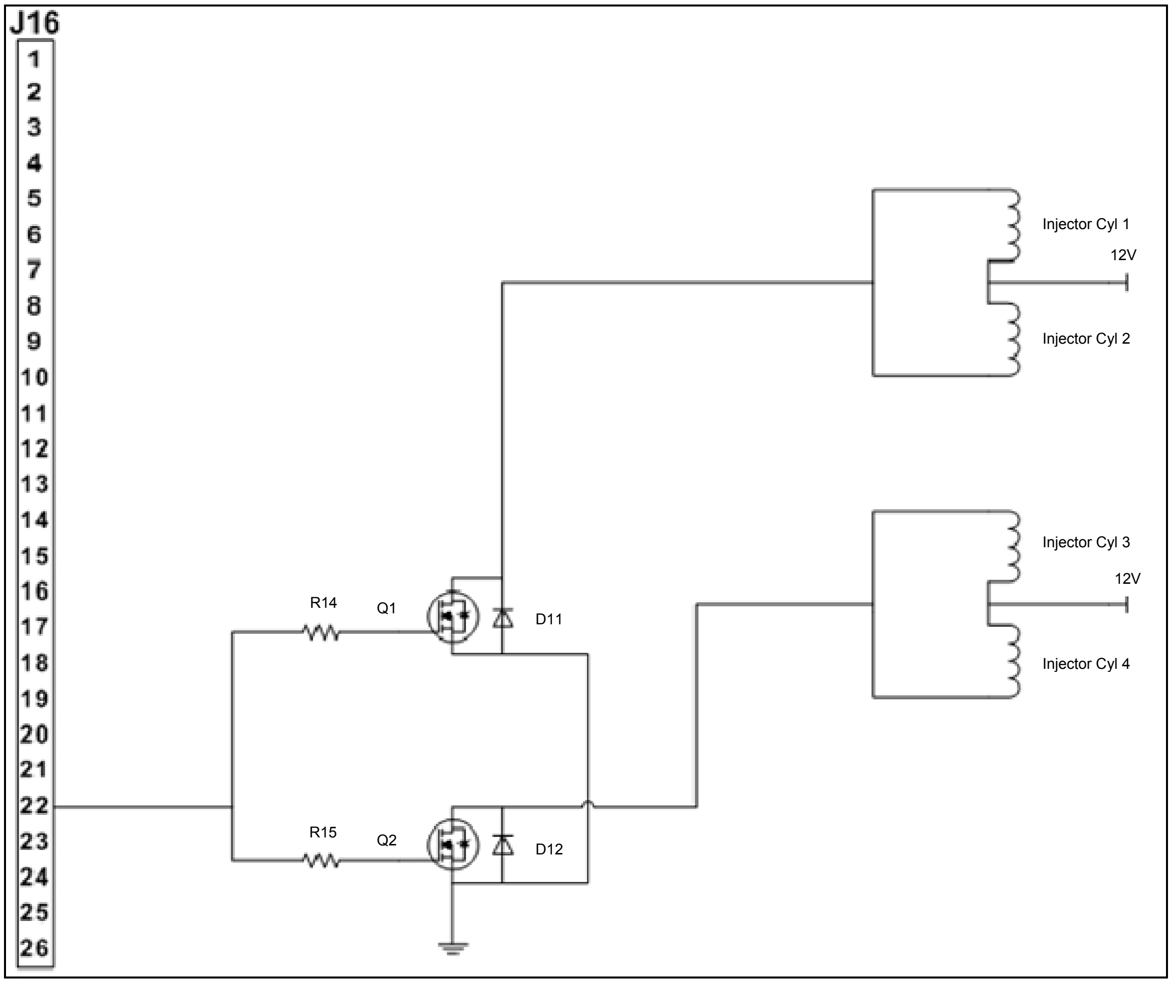


Figure 3.10 Injector Driver circuit

BUZ10 mosfet transistors provide a good application for automotive high current drivers and were used to drive the injectors. The existing Honda fuel injectors which are high impedance injectors were used in this project. The choice of using the existing injectors was motivated by the spray pattern of the injectors which was suited for the design of the cylinder head intake ports as well as from a cost saving point of view. Since batch injection was the choice for fuel delivery two injectors can be connected in parallel without overloading the transistor driver.

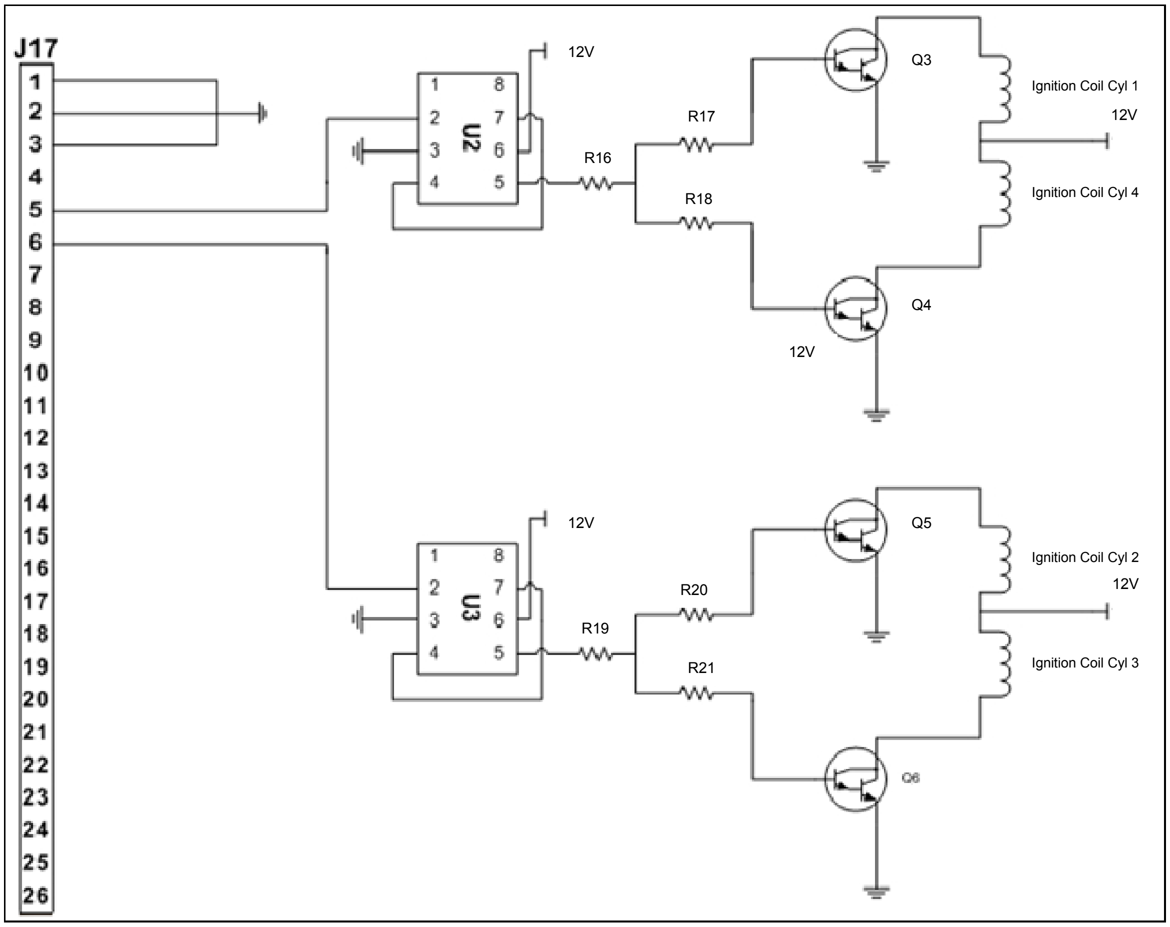


Figure 3.11 Ignition coil driver circuit

The MC34151 integrated component provided an ideal mosfet driver coupling the microcontroller output pin to the ignition module which amplifies the signal to drive the ignition coil. Figure 3.11 above illustrates the ignition coil driver circuit. The complete hardware schematic is illustrated in Appendix B [22].

An existing ignition module from the motorsport market was used for the ignition driver as this had the adequate heat sink design and construction for automotive application. This unit uses a BUZ941Z mosfet transistor as the main component.

Once all the input and output circuitry was built and configured, a signal generator and oscilloscope was used to simulate and test the entire system under normal operating conditions of an engine as illustrated in Figure 3.12 below.

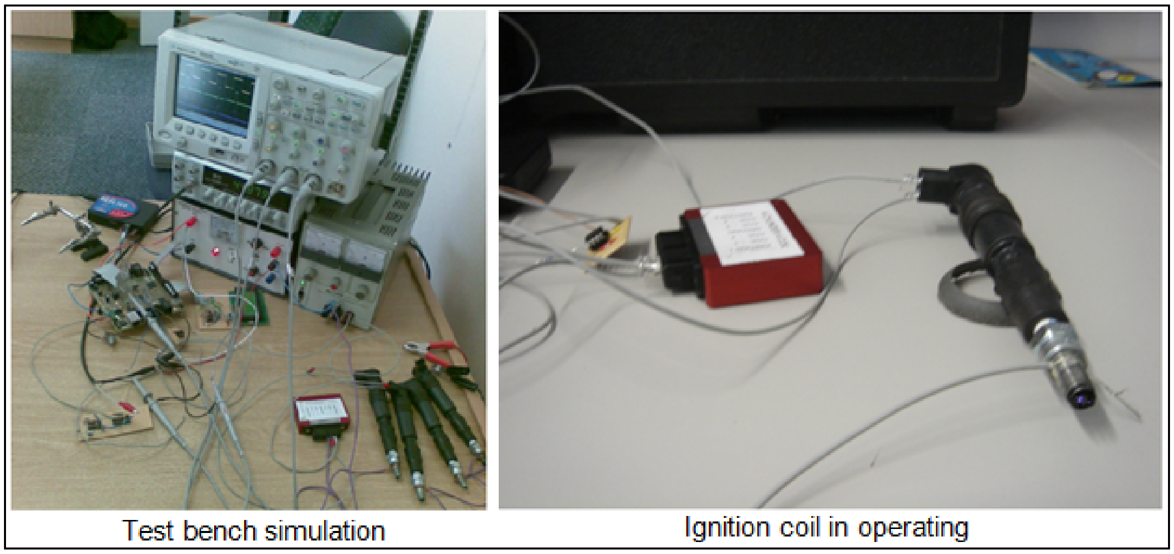


Figure 3.12 Hardware simulation and bench testing

During this simulation and testing it was possible to generate sine wave inputs into the microcontroller and be able to test output signals to the driver circuitry to verify the spark discharged on the spark plug itself. In Figure 3.12 above, on the right shows the actual spark discharge from the spark plug. This then proved that the hardware was functional.

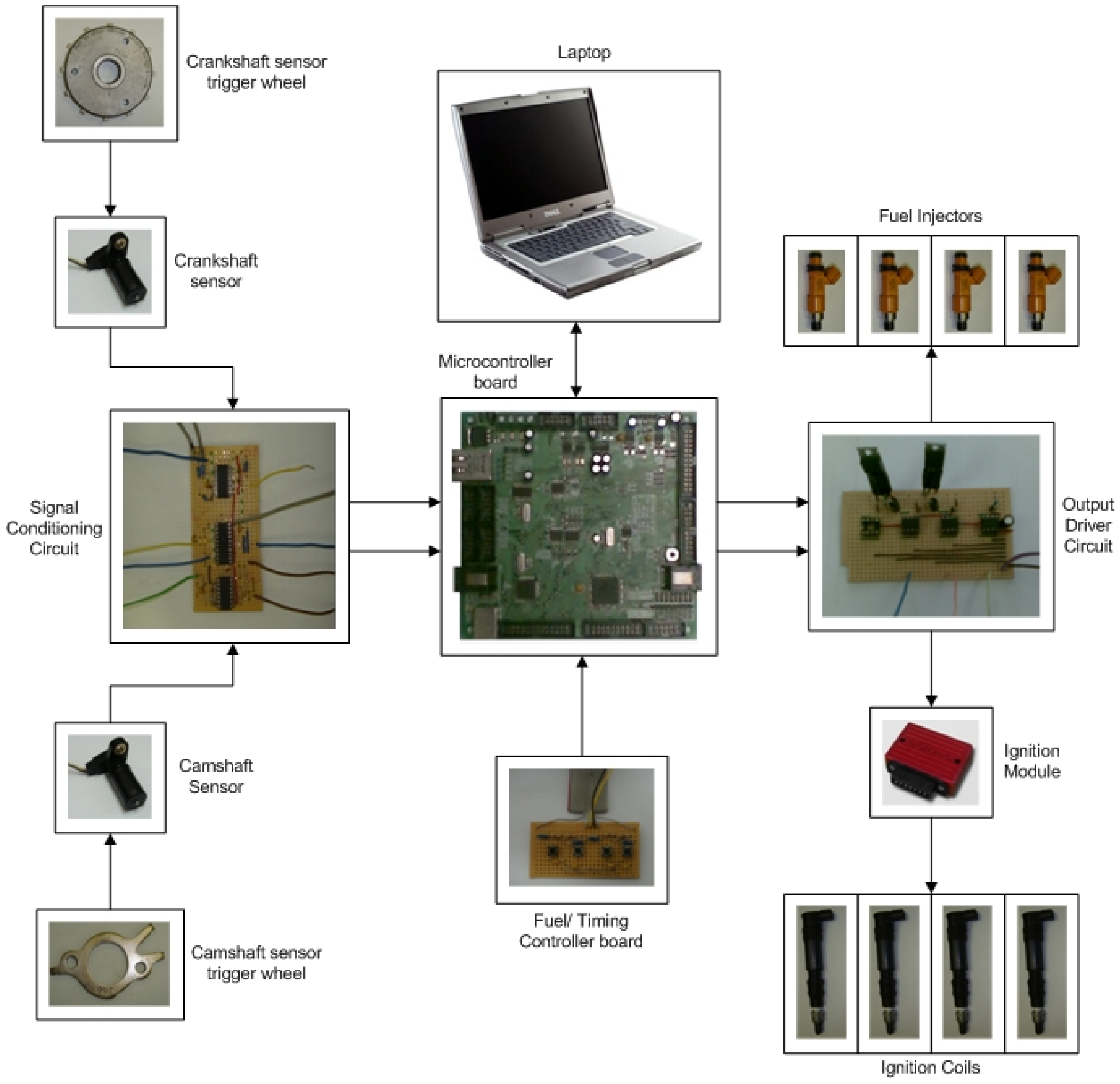


Figure 3.13 Complete overview of the hardware

Figure 3.13 above illustrates the complete hardware structure of the project. The input sensors and conditioning circuitry are shown on the left, with the main controller board, laptop interface and fuel timing board for adjustment of the fuel and ignition timing parameters. The output components and circuitry are given on the right hand side of the diagram.

3.3 Software Development

Appendix C provides the overall software flowchart. This section will now explain the main section of the software code listed in Appendix D. The core processing of the software lies in the interrupts as every input from the crankshaft and camshaft sensor is considered vital information. Both crankshaft and camshaft sensors were connected to the interrupt pins of the microcontroller as this will allow responsive action for all calculations and output control.

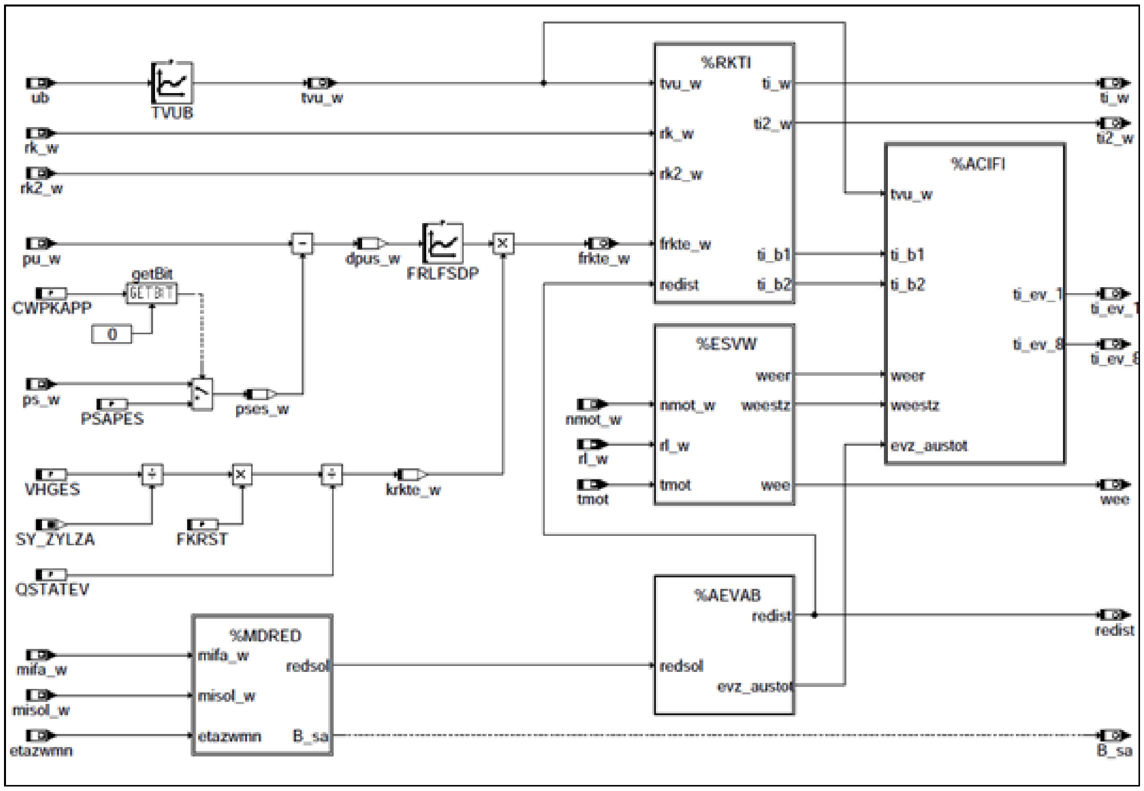
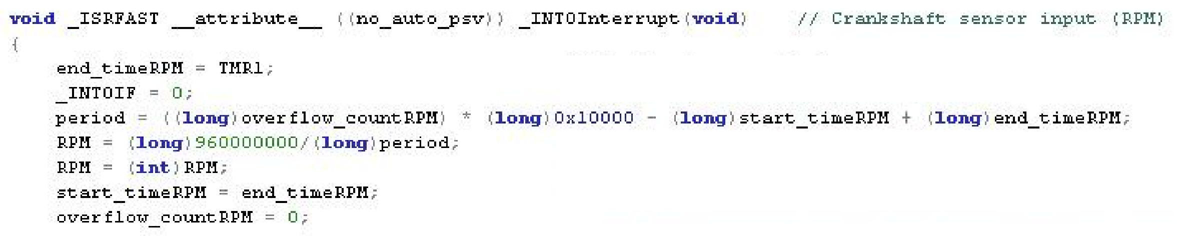


Figure 3.14 Bosch MS3.1 injection time function definition [12]

Figure 3.14 above is an extract showing just one function of calculation fuel injection quantity in a Bosch Motorsport engine control unit. It can be noted that these systems employ complicated functions and algorithms. A feature of this project is to develop simpler yet functional control of fuel injection and ignition timing for a formula student project.

The crankshaft sensor input is used for engine speed calculation, injector pulsing, ignition timing calculations, and firing of cylinder 1 and 4 ignition coils. The camshaft sensor is used for ignition firing of cylinder 2 and 3 coils.

The breakdown of the crankshaft sensor interrupt routine is now explained.

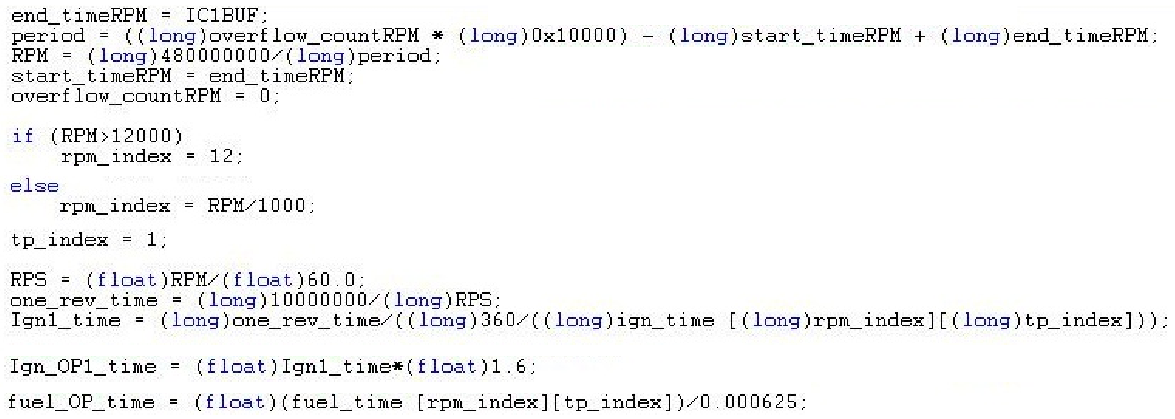


```
void _ISRFAST __attribute__ ((no_auto_psv)) _INT0Interrupt(void)     // Crankshaft sensor input (RPM)
{
    end_timeRPM = TMR1;
    _INT0IF = 0;
    period = ((long)overflow_countRPM) * (long)0x10000 - (long)start_timeRPM + (long)end_timeRPM;
    RPM = (long)960000000/(long)period;
    RPM = (int)RPM;
    start_timeRPM = end_timeRPM;
    overflow_countRPM = 0;
```

The section of code above deals with capturing the contents of Timer 1 register on each crankshaft trigger to get an update of the engine speed. Timer 1 is a dedicated timer for rpm calculation only. The interrupt flag is then cleared so as to initiate another interrupt as soon as the next trigger takes place [19, 20].

There was a great deal of time spent to optimize the rpm and ignition timing calculation code so as to do away with the floating point variables and calculations. These variables take up more space and thus also increase the execution time of the code [19]. Execution timing tests were carried out to find a faster coding structure as this interrupt involved all the important information and had to be processed quickly.

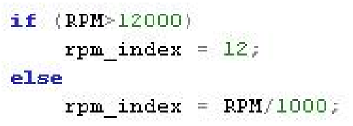
The initial code, as below, had floating point variables and later the values were compensated by multiplying out the decimal places and then later diving by the multiplication factor used.



```
end_timeRPM = IC1BUF;
period = ((long)overflow_countRPM * (long)0x10000) - (long)start_timeRPM + (long)end_timeRPM;
RPM = (long)480000000/(long)period;
start_timeRPM = end_timeRPM;
overflow_countRPM = 0;

if (RPM>12000)
    rpm_index = 12;
else
    rpm_index = RPM/1000;

tp_index = 1;

RPS = (float)RPM/(float)60.0;
one_rev_time = (long)10000000/(long)RPS;
Ign1_time = (long)one_rev_time/((long)360/((long)ign_time [(long)rpm_index][(long)tp_index]));

Ign_OP1_time = (float)Ign1_time*(float)1.6;

fuel_OP_time = (float)(fuel_time [rpm_index][tp_index])/0.000625;
```

The method of rpm calculation is done by first calculating the period of the signal over the last two pulses received, the difference between these is the pulse width of the signal which is then converted into a frequency. To make the interrupt handler much quicker casting of variables was used. Once the rpm value is converted into the correct format, it is stored into the suitable variable and the registers are cleared for the next rpm calculation.

A two dimensional array was created for ignition timing and injection time which stores the value of either ignition angle or injection time based on the specific rpm and throttle position. Rpm and throttle position values need to be calibrated into an index for the arrays.



```
if (RPM>12000)
    rpm_index = 12;
else
    rpm_index = RPM/1000;
```

The indexing for rpm is done as shown in the code above, where the rpm index is split into ranges of 1000rpm, index 1 representing 1000rpm and 2 representing 2000rpm and so on….

The throttle position index is configured in the same manner in the section of code for the analog-to-digital conversion with index values ranging from 0 to 9 representing closed to full throttle opening. The array allows specific values of ignition timing or injection time to be used based on specific engine speeds against throttle position opening. This strategy of throttle position vs engine speed is known as Alpha-N. Engine load is assumed on the basis of throttle angle instead of direct airflow. Alpha-N is mostly used on racing applications where the throttle is operated primarily at wide-open-throttle as well as the engine having very long duration camshafts and tuned air intake systems. These modifications affect the amount of pressure in the intake manifold where-by almost the same amount of pressure is measured at idle as at wide open throttle. By using a manifold pressure sensor under the Speed Density strategy, there would be no difference on engine load measured due to the engine configuration in race engines with no major in intake pressure. Another strategy that is implemented is Mass Air Flow (MAF) system using a sensor mounted in front of the throttle body directly measuring the amount of air entering into the engine [15].

The important part of the ignition time is the accurate measurement of ignition angle before TDC. Earlier or delayed ignition has a huge influence on engine response which can have a detrimental effect to the engine especially at higher engine speeds.

Ignition angle is dependent on engine speed and load. An equation was generated for use for the ignition angle calculation which is given as a measure of degrees out of a complete revolution, where θ is ignition angle,

$$\frac{\theta_{spk}}{360} \qquad \text{Eq. 1}$$

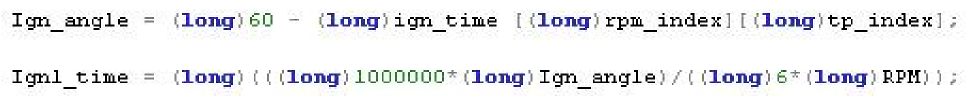
The time taken for one revolution is

$$\frac{1}{rpm/60} \qquad \text{Eq. 2}$$

Therefore the time *t* in seconds for ignition angle calculation is given as

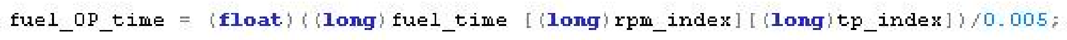
$$t = \frac{\theta_{spk}}{360} * \frac{60\sec s}{rpm} \qquad \text{Eq. 3}$$

Equation 3 is implemented in software as,



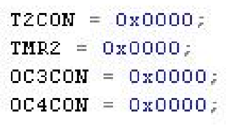
```
Ign_angle = (long)60 - (long)ign_time [(long)rpm_index][(long)tp_index];

Ign1_time = (long)(((long)1000000*(long)Ign_angle)/((long)6*(long)RPM));
```

In the code above, the section `(long)ign_time [(long)rpm_index][(long)tp_index]` deals with the array of the ignition angle based on the rpm and throttle position value. The additional multiplication and division is to compensate for use of floating point variables.
The code for the initial calculation of the duration of the fuel injections is given as



```
fuel_OP_time = (float)((long)fuel_time [(long)rpm_index][(long)tp_index])/0.005;
```

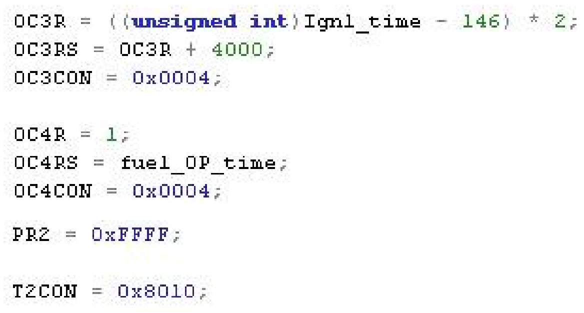
It was found to be good software practice to clear the control registers before reassigning new values [6, 7, 21]. This is implemented in the software as follows



```
T2CON = 0x0000;
TMR2 = 0x0000;
OC3CON = 0x0000;
OC4CON = 0x0000;
```

Once all the appropriate registers have been allocated with the required values, the timer and output compare modules are then reconfigured. The output compare module of the microcontroller was used for the output of the ignition timing and fuel injection. These prove ideal as once the timer is configured, on a match of the output compare register and the associated timer register, the output will be enabled.

The crankshaft trigger 60° BTDC will start Timer 2 which is run in 32 bit mode and will be used as the base timer for the fuel injection and ignition timing control. There is a difference in the way that the injection and ignition output is controlled. The ignition output is pulsed high for a fixed time duration with just the variation of the starting of the pulse. The injection has a fixed time of when it is pulsed high, but the pulse width is varied according to how much fuel is required. The code for the registers for the ignition and injection is given as

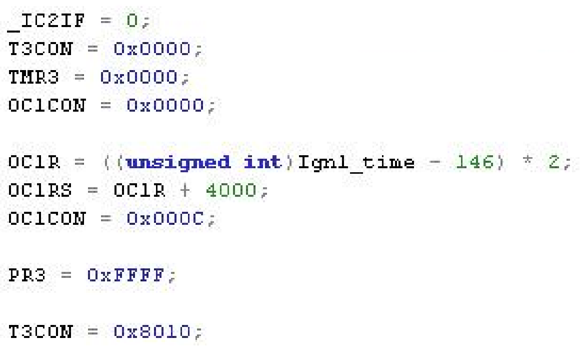


```
OC3R = ((unsigned int)Ign1_time - 146) * 2;
OC3RS = OC3R + 4000;
OC3CON = 0x0004;

OC4R = 1;
OC4RS = fuel_OP_time;
OC4CON = 0x0004;

PR2 = 0xFFFF;

T2CON = 0x8010;
```

A spreadsheet table listed in Appendix E was created to analyse worst case scenarios for the main registers, this being for the timer registers and specific registers for output control. The table shows typical register values at high rpm to see if there is enough time to carry out all the necessary calculations and still do the output task in the available time. It can be noted that from Table 3.2 that at
12 000 rpm there is only 2.5 milliseconds available until the next input pulse is received. This indicates that all calculations and the output controlling code must be completed within this time.

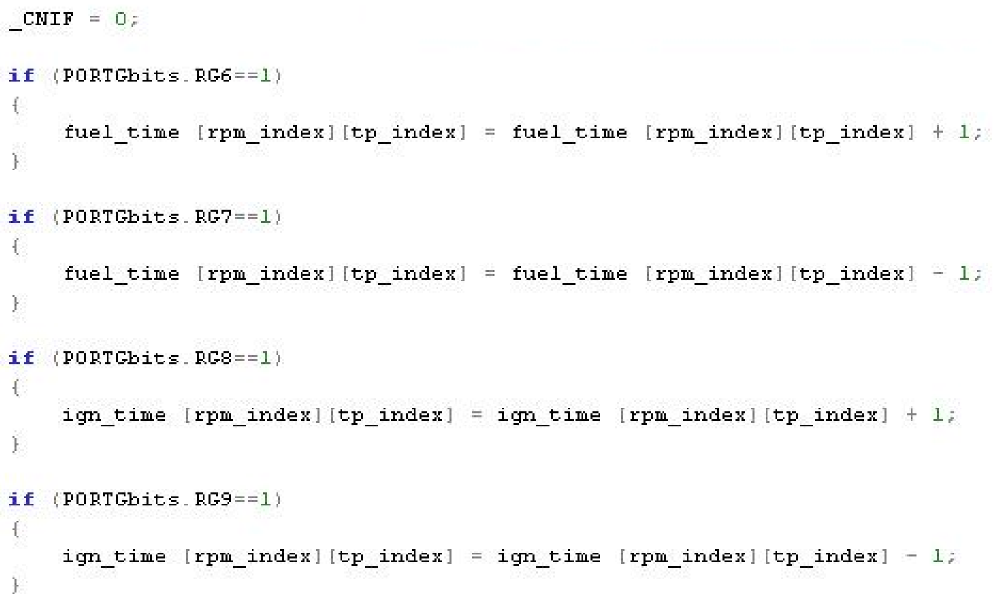
The crankshaft sensor interrupt routine has all the vital information and calculations. Since the engine speed and all related information has been calculated here, the camshaft sensor routine only includes the control of the ignition firing of cylinder number 2 and 3.

The code below illustrates the interrupt service routine for the camshaft sensor. A separate timer is used for the output compare module used for the control of cylinders 2 and 3.



```
_IC2IF = 0;
T3CON = 0x0000;
TMR3 = 0x0000;
OC1CON = 0x0000;

OC1R = ((unsigned int)Ign1_time - 146) * 2;
OC1RS = OC1R + 4000;
OC1CON = 0x000C;

PR3 = 0xFFFF;

T3CON = 0x8010;
```

During the engine tuning phase of this project it was realized that additional coding had to be developed to enable live adjustments to the fuel and ignition timing parameters. By having the access to control these parameters it was possible to see the influence of these adjustments directly on the engine. The code below illustrates the controlling of the fuel and timing parameters when the specific increment or decrement buttons were pressed. These buttons can be seen in the overview in Figure 3.13.



```
_CNIF = 0;

if (PORTGbits.RG6==1)
{
    fuel_time [rpm_index][tp_index] = fuel_time [rpm_index][tp_index] + 1;
}

if (PORTGbits.RG7==1)
{
    fuel_time [rpm_index][tp_index] = fuel_time [rpm_index][tp_index] - 1;
}

if (PORTGbits.RG8==1)
{
    ign_time [rpm_index][tp_index] = ign_time [rpm_index][tp_index] + 1;
}

if (PORTGbits.RG9==1)
{
    ign_time [rpm_index][tp_index] = ign_time [rpm_index][tp_index] - 1;
}
```

With the code optimization discussed earlier in this chapter, the screenshot in Figure 3.15 below shows the execution time of the crankshaft interrupt service routine which is 105 microseconds. The optimization proved to be very beneficial into obtaining low code execution time which is crucial in engine control systems.

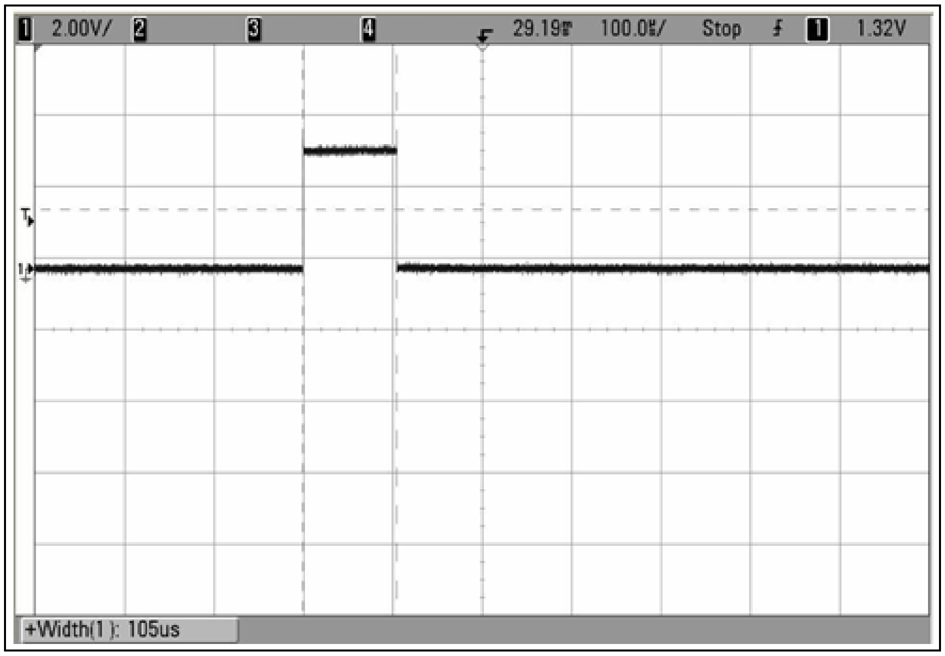


Figure 3.15 Measurement of the crankshaft interrupt routine

The development then moved on to the actual engine itself set up on the NMMU engine dyno. The wires for the magnetic sensors were screened cables so as to prevent any interference with the input signals and give false signal information. A complete wiring harness was made up for the application on the dyno. An LCD display was used to display the important information necessary during engine tuning. The complete system is illustrated in Figure 3.16 below.

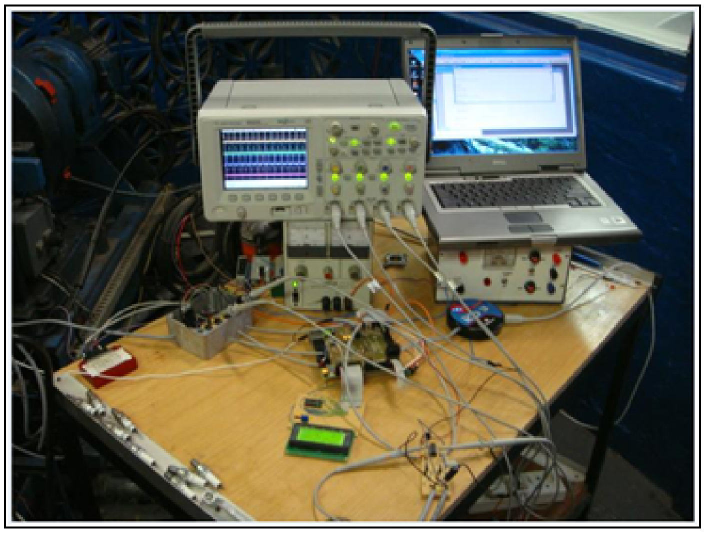


Figure 3.16 Final prototype setup in dyno cell

The formula student project required a lot of further research and development work through the project. Due to the regulations for the engine specific air intake and exhaust systems are required to be engineered. Since there is not much technical information available from the manufacturer a lot of individual work had to be carried out. An engine simulation software was made available for use to the team and this required raw data from the engine for input parameters into the software.

Figure 3.17 and Figure 3.18 illustrate some of the development work that was carried out.

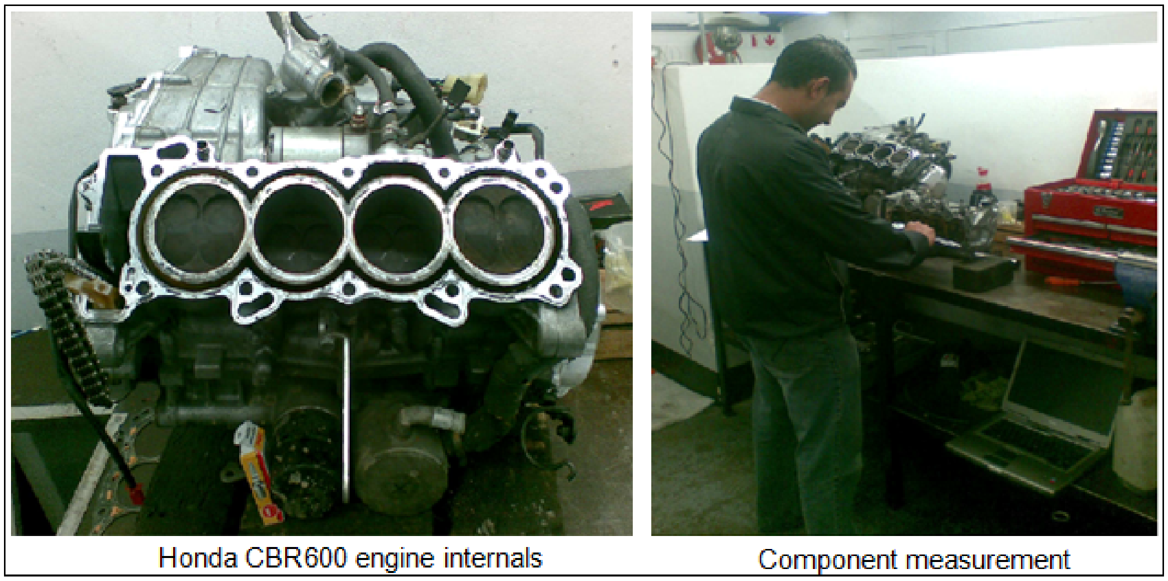
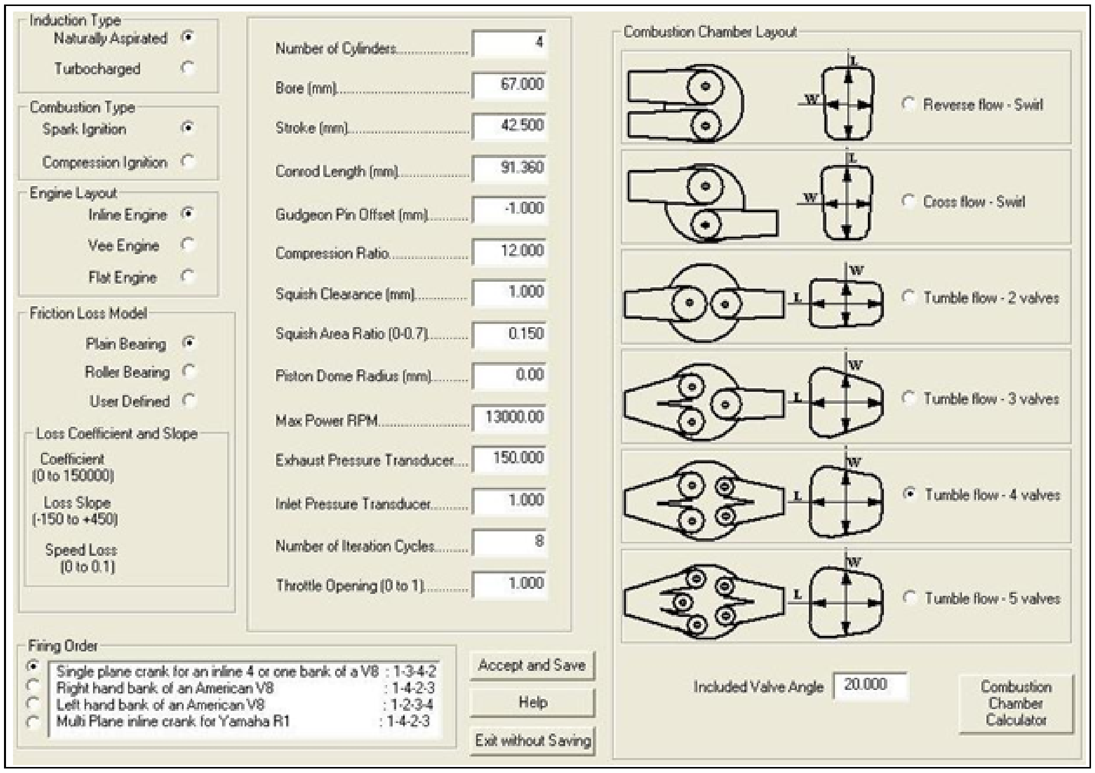


Figure 3.17 Engine data measurement



Induction Type
Naturally Aspirated [✓]
Turbocharged

Combustion Type
Spark Ignition [✓]
Compression Ignition

Engine Layout
Inline Engine [✓]
Vee Engine
Flat Engine

Friction Loss Model
Plain Bearing [✓]
Roller Bearing
User Defined

Loss Coefficient and Slope
Coefficient (0 to 150000)
Loss Slope (-150 to +450)
Speed Loss (0 to 0.1)

Number of Cylinders	4
Bore (mm)	67.000
Stroke (mm)	42.500
Conrod Length (mm)	91.360
Gudgeon Pin Offset (mm)	-1.000
Compression Ratio	12.000
Squish Clearance (mm)	1.000
Squish Area Ratio (0-0.7)	0.150
Piston Dome Radius (mm)	0.00
Max Power RPM	13000.00
Exhaust Pressure Transducer	150.000
Inlet Pressure Transducer	1.000
Number of Iteration Cycles	8
Throttle Opening (0 to 1)	1.000

Combustion Chamber Layout
Reverse flow - Swirl
Cross flow - Swirl
Tumble flow - 2 valves
Tumble flow - 3 valves
Tumble flow - 4 valves [✓]
Tumble flow - 5 valves

Included Valve Angle 20.000

Firing Order
[✓] Single plane crank for an inline 4 or one bank of a V8 : 1-3-4-2
Right hand bank of an American V8 : 1-4-2-3
Left hand bank of an American V8 : 1-2-3-4
Multi Plane inline crank for Yamaha R1 : 1-4-2-3

Figure 3.18 EngMot4T software

Chapter 4 – Engine Tuning

The tuning of an engine involves the setting up, calibration and configuration the ems to control the engine throughout the engine speed and load range for optimal engine performance. The final overall output of the engine is dependent on the application of the engine use, whether for passenger vehicles or motorsport. Initially the specific engine parameters need to be setup in the ems, this being cylinder numbers, input sensor type, injection and ignition type, as well as many ems specific settings and capabilities. The specific sensors used need to be calibrated as the ems needs to understand the operating parameters of the sensor to be able to carry out appropriate tasks and calculations.

The lambda meter essentially gives an indication of the fuel mixture and there is no direct relation to ignition timing. Additional equipment is required for ignition timing monitoring of which there are a few types such as cylinder pressure measurement and knock analysis. Having access to this type of equipment allows greater scope for an engine as well as finer optimization for the requirements, be it for fuel economy or high performance race engines.

During the earlier stages of this project the researcher carried out some research work at the University of Applied Sciences in Wolfsburg, Germany. The institution has an extensive automotive engine test facility some of which is illustrated in
Figure 4.1 and Figure 4.2 and explained later on.

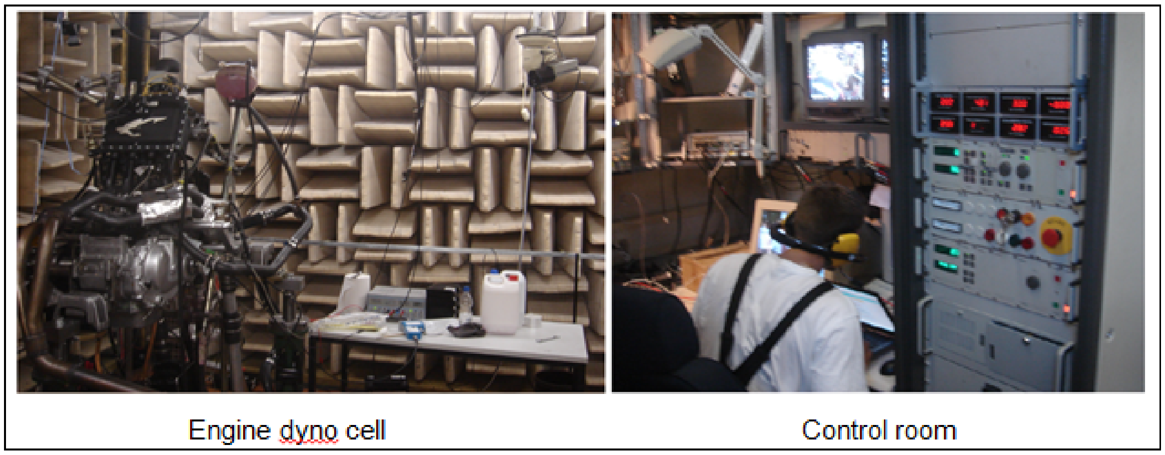


Figure 4.1 UAS Wolfsburg Engine test facility

Figure 4.1 shows the engine dyno cell fitted with sound proofing to dampen engine noise, two cameras with joystick operation provided 360 degree view of the engine and supporting components from the control room TV screens. A microphone fitted inside the dyno cell provided audible feedback into the control room for any case of engine malfunction. All temperatures, pressures and control parameters were available on display via the control system. Safety features such as low oil pressure and excessive temperature control are in place to trigger alarms and switch off the engine to prevent any critical damage to the engine during operation.

Generally, a lambda sensor is fitted in the exhaust system at the point where all the gases collect from each of the cylinders; this would give an overall reading of the fuel mixture. However for precise measurement, individual lambda sensor can be installed on each cylinder to give an exact indication of each cylinder's operation. This is a great assistance in fault finding for any engine component failure as the problem can be pinpointed to the specific cylinder. In the same light as the individual cylinder lambda measurements, the same can be done for cylinder pressure measurement. This entire package will give an engine tuner full scope of the engine characteristics as well as allow extreme limits to be reached and tested.

A typical example, if one fuel injector had to be malfunctioning, there would be minor noticeable difference with one lambda sensor as the sensor would be measuring the overall gases. However with individual cylinder lambda measurement this would be immediately noticed.

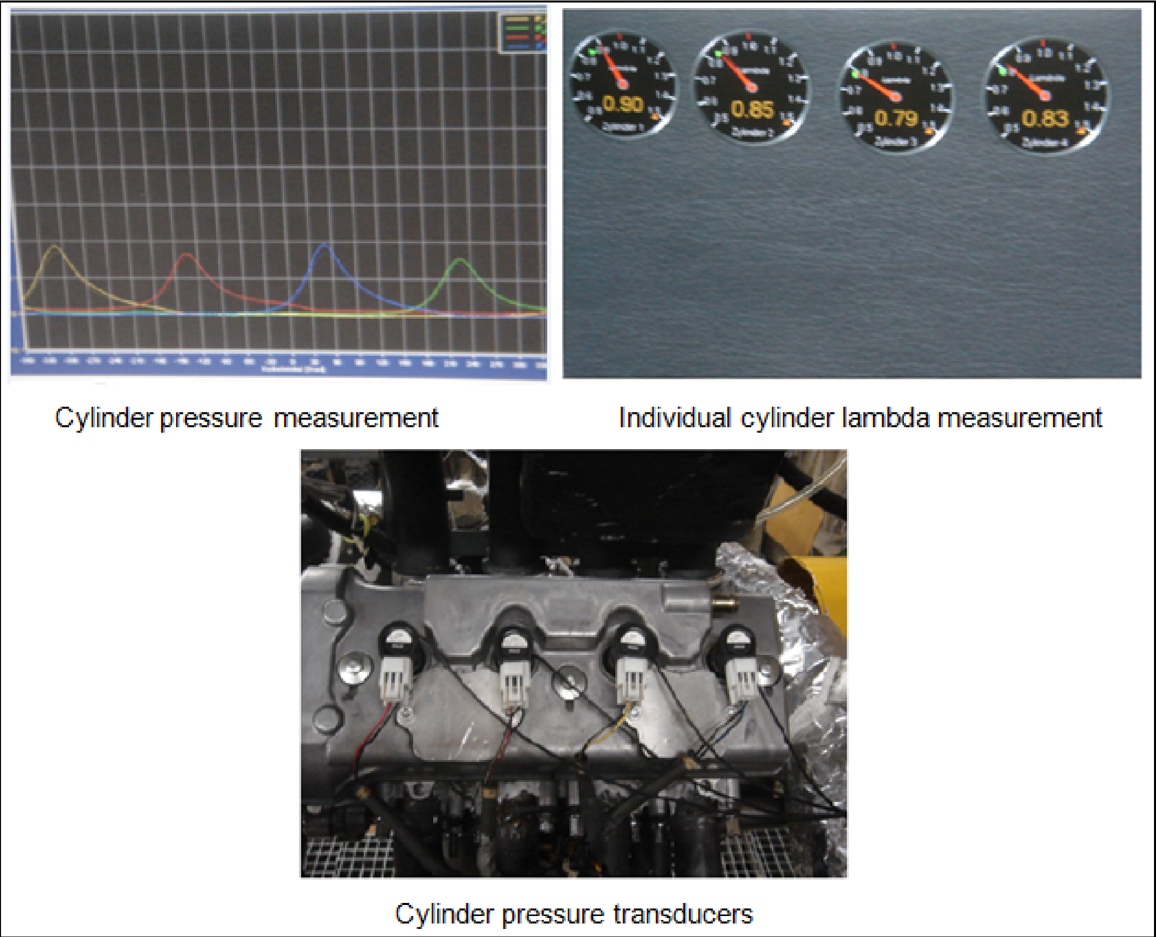

0.90
0.85
0.79
0.83
Cylinder pressure measurement
Individual cylinder lambda measurement
Cylinder pressure transducers

Figure 4.2 UAS Wolfsburg engine tuning equipment

Figure 4.2 above show the full measurement system at the UAS Wolfsburg. The cylinder pressure measurement graphs will indicate any leaking cylinders, as well as engine knock conditions. The lambda readings also show each cylinder value and it can be noticed in Figure 4.3 that there is an unbalanced cylinder, cylinder 3.
In this case the adjustments will then be made in the ems software to reduce the amount of fuel for cylinder 3 only.

The cylinder pressure transducers are engine specific as they replace the spark plug and have integrated spark plug and pressure transducer.

4.1 Fuel injection calibration

The first task on engine tuning is getting the engine started. A ‘start-up’ map needs to be created for the fuel injection and ignition timing parameters. These start-up parameters are dependent on the configuration of the engine, fuel type and injector size, just to name a few. The combustion process is a mixture of air and fuel, and a spark to ignite the combined mixture. The chemically correct term for complete combustion with no extra fuel or air left over is referred to as stoichiometric. The numeric value for this condition is given as a ratio of air-to-fuel which is 14.68: 1, referred to as air-fuel ratio. This is also equivalently expressed as a lambda
value of 1. Mixtures having excess air – leaner – are lambda >> 0.9, and in turn mixtures having an air deficiency – richer – are lambda << 0.9 [9, 15]. Figure 4.3 gives an overview of air-fuel ratio characteristics.

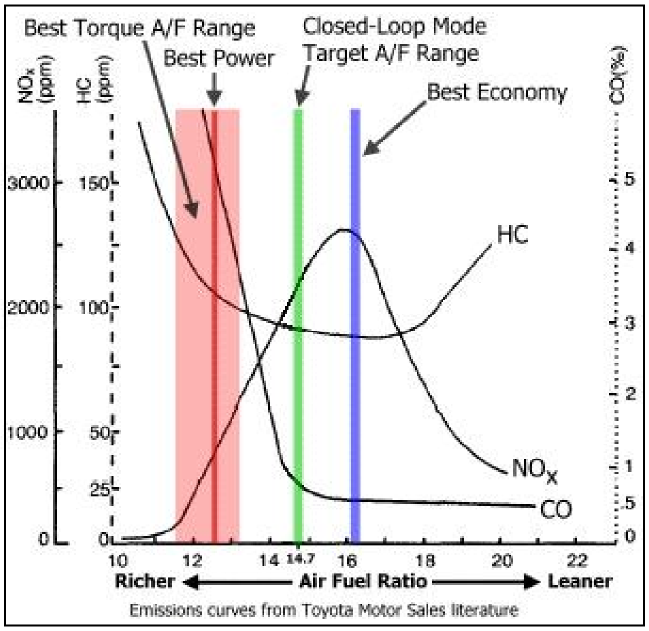


Figure 4.3 Air-fuel ratio characteristics [17]

To be able to determine how much fuel to inject into the engine, it relates to the amount of air present at the given time as well as the desired air-fuel-ratio for the load condition.

From a theoretical point of view it is a very tedious exercise to calculate the specific values to be entered for the fuel injection. Some insight into these calculations involves aspects such the Ideal Gas Law, measured values and calculations from input sensors and engine parameters such as volumetric efficiency, engine capacity, injector size, fuel pressure, just to name a few.

The Ideal Gas Law (representing the air) is given as

$$\mathbf{PV = nRT} \qquad \text{Eq. 4}$$

Where: **P** = pressure

V = volume

n = number of moles (which is related to the mass of the gas)

R = the ideal gas constant

T = the absolute temperature

The volumetric efficiency (VE) is a percentage of the pressure inside the cylinder versus the pressure in the manifold. Furthermore, to then calculate the amount of fuel required involves computing the mass of the fuel, injector flow rate, fuel pressure, and input sensor readings.

For the theoretical part, there are always unknowns that are encountered which then leads to a time consuming process for the start-up map. Experienced engine tuners are able to create a fuel injection start-up map from known parameters of the engine capacity as well as the fuel injector flow rates.

In this case an initial set of numbers are entered into the injection map and appropriate ignition timing and the engine is then given a start signal. Based on the responses from the engine the tuner can then adjust the parameters.

Generally, on engine starting, if there is any black smoke coming out of the exhaust then there is too much fuel being injected. In the alternate case, if there is too little fuel then there would either be no response from the engine or a backfire through the exhaust. Once the engine is started it becomes easier to judge what the engine requires. After starting the engine a quick feedback tool is the exhaust lambda readings as well the colouration of the spark plugs.

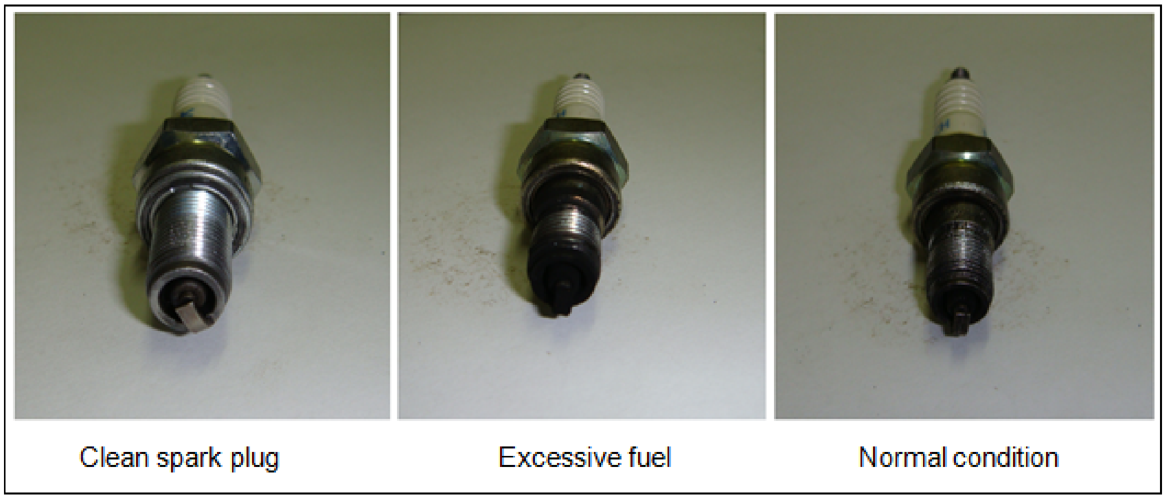


Figure 4.4 Spark plug analysis

Figure 4.4 gives example of a quick indication of what is taking place in the combustion chamber; these were photographed during the tuning process of this project. The spark plug on the left shows a clean spark plug, before any combustion has taken place. The middle one can be seen as very black and thus having excessive fuel. The one on the right shows ideal combustion taking place.

Several aftermarket ems manufacturers recommend an idle pulse width of not lower than 1.7 milliseconds and operated up to 80% duty cycle to prevent overheating of the injectors where they remain more in an on state than off state. The main task is to get the engine started and idling, the map can be further developed from this.
There is no harm done to the engine during these conditions as there is no load on the engine. The attention to detail on the lambda readings need to be carried out when there is load applied onto the engine.

Slightly leaner mixtures result in better fuel economy and slightly richer mixtures result in better torque. The peak flame speed during combustion is found at lambda ≈ 0.9 and adding excess fuel or air will slow down the combustion process. Leaning out mixtures increases the flame speed and adding fuel slows the flame travel. It is better to start out which slightly richer mixtures and then lean then out to the desired mixture. The actual optimal air-fuel mixture requirements of an engine vary as a function of temperature, rpm and load [6].

4.2 Ignition timing calibration

Although ignition timing values are specific to numerous factors, there are some general guidelines. Most piston engines, regardless of compression ratio idle well with at least 10 degrees of advance at idle; the ignition timing also plays a big role in idle quality.

At slow engine speeds, typically up to 3000rpm, 8 to 20 degrees of timing works well with gradually up to 40 degrees at higher engine speeds. “An engine with too much timing will detonate, regardless of how much fuel is thrown at it. An engine with too little timing will perform poorly and overheat the exhaust in short order”.

Ignition timing can be advanced up to a degree before the condition of knock occurs. Figure 4.5 below illustrates cylinder pressure graphs under normal condition and under knock. Earlier in this chapter, Figure 4.2 illustrates the typical cylinder pressure measurement software in application.

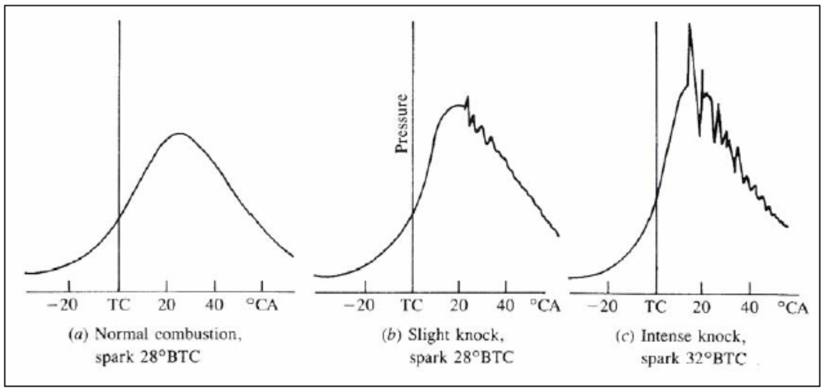


Figure 4.5 Cylinder pressure measurement illustrating knock [5]

During the ignition timing calibration of this project, high exhaust temperatures were experienced due to slow ignition timing as discussed earlier. This is illustrated in Figure 4.6 below.

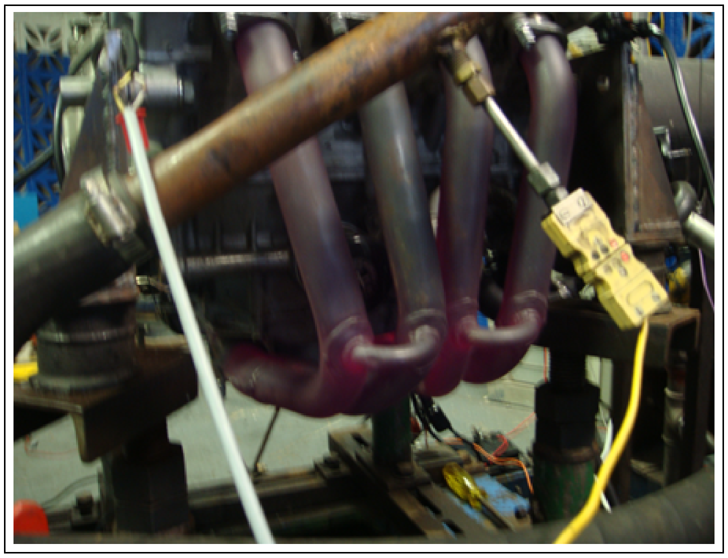


Figure 4.6 Exhaust temperatures during tuning

To rectify these high exhaust temperatures, additional ignition timing and fuel were added at the specific engine speed and load, the exhaust temperatures had later returned to the normal operating range.

Chapter 5 – Testing, Results and Conclusion

Once all the software and hardware were written and developed respectively, each was specifically tested for its functionality as one fault in either can affect the post calculations and processes.

The initial hardware testing involved generating appropriate signals as per the real environment from a signal generator which were then fed into the corresponding hardware inputs. An oscilloscope was later used on the outputs of the hardware to validate functionality as per the designed process. Photographs and screen captures were documented during each of the tests which will be discussed in this chapter.

The testing of the all the software functionality explained in the previous chapter was initially carried out with the internal simulation function in the MPLAB software. Each register was monitored under the different input conditions. The theoretical calculations of the register values for the different conditions were then validated.

The entire system was then setup on the real engine and calibrated as per use on the Honda CBR600 engine. The appropriate start-up map and main map tuning for load conditions up to 3000rpm were carried out, as per communication with project promoter Professor Theo van Niekerk and supported by Professor Udo Becker (University of Applied Sciences Wolfsburg, Germany).

5.1 Hardware testing

The first hardware test involved validating the functionality of the input sensor signal conditioning circuitry with appropriate signals via a function generator. The raw crankshaft sensor signal would follow a sinusoidal waveform and the required output waveform is a square wave for the microcontroller input. The appropriate voltages were also validated.

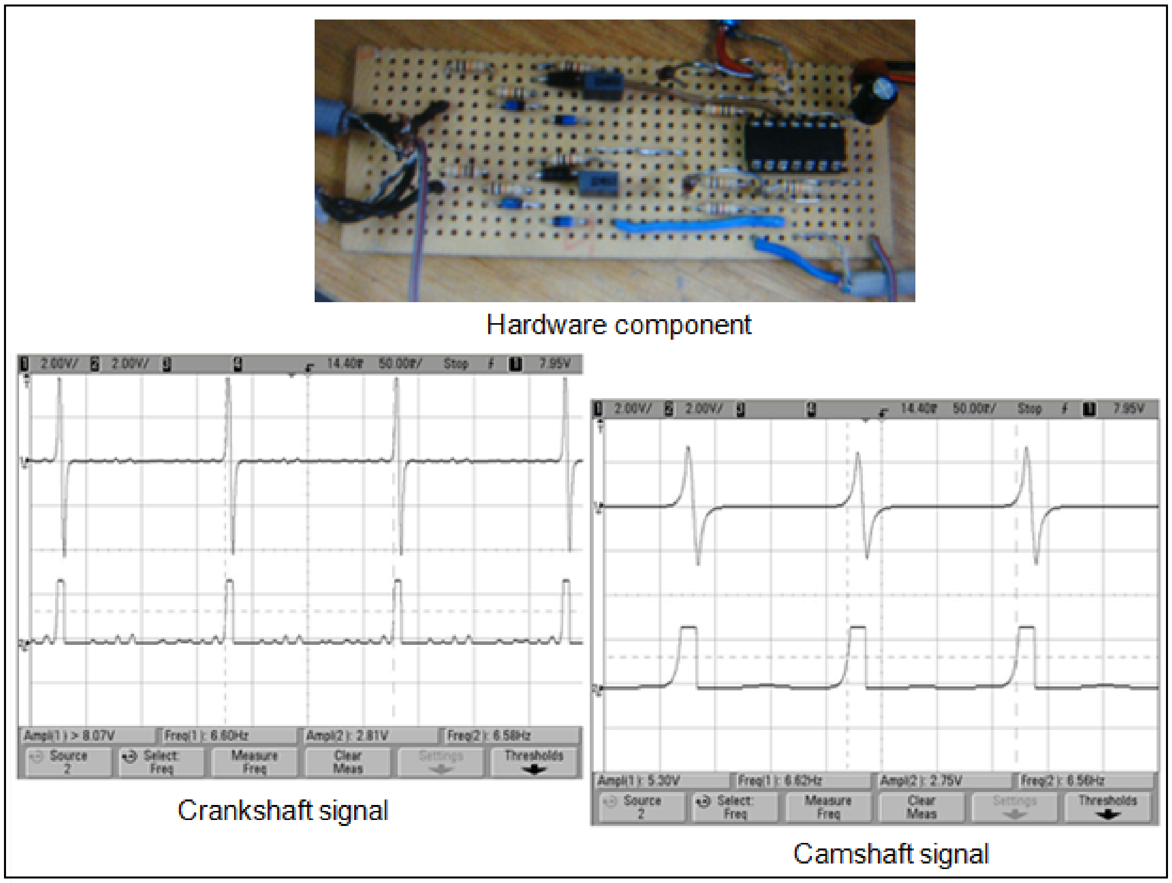


Figure 5.1 Input signal waveform analysis

In Figure 5.1, the crankshaft signal on the left side shows the sensor sinusoidal waveform on signal 1 (top) and the filtered square wave on signal 2 (bottom). It can be noted that with the same time div setting on the oscilloscope for both waveforms, there are more pulses on the crankshaft signal than the camshaft signal.

The reason for this is that the camshaft travels at half the speed of the crankshaft, for every 720 degrees of crankshaft revolution, the camshaft will go through 360 degrees of rotation. This is why a full combustion cycle is 720 degrees because the full operations of the crankshaft and camshaft have been both completed.

The form of the signal and voltage levels of the square wave is suitable enough for the microcontroller to be able to register a valid input signal. The first simulations via a signal generator proved to be all functional and acceptable. However during the real environment testing on the engine in the dyno cell, precautionary measure were taken to prevent noise interference on the input signals and screen cables were used for the signal wires from the sensor to the interfacing hardware.

The microcontroller output driver hardware was initially developed on a separate development board and the first simulation and testing via signal generator proved to be all functional as per design. The problems encountered in the dyno cell testing were that of noise interference due to the high current mosfet driver for the fuel injectors. Isolating of this hardware into a metal enclosure as in Figure 5.2 below (right) proved to rectify all electrical noise previously encountered.

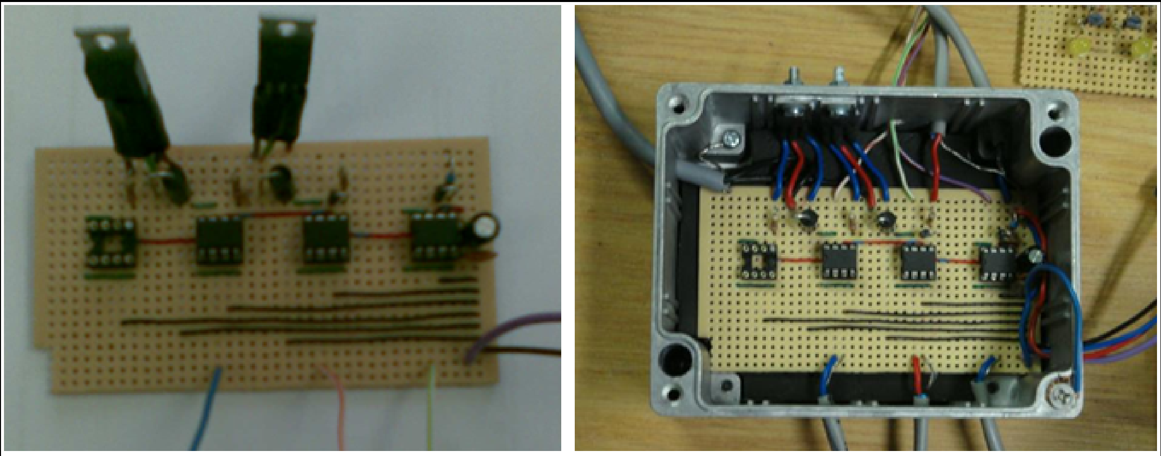

Output driver hardware, stage 1 Output driver hardware, stage 2

Figure 5.2 Output driver hardware

In Figure 5.3 below the oscilloscope screen shot illustrates the type of interference on the output channels. Signal 1 (top) is the crankshaft input signal waveform, signal 2 (middle) is the fuel injector output signal waveform, and signal 3 (bottom) is the output ignition signal for cylinders 2 and 3. The interference experienced was the random "double triggers" of the ignition output signal, signal 3.

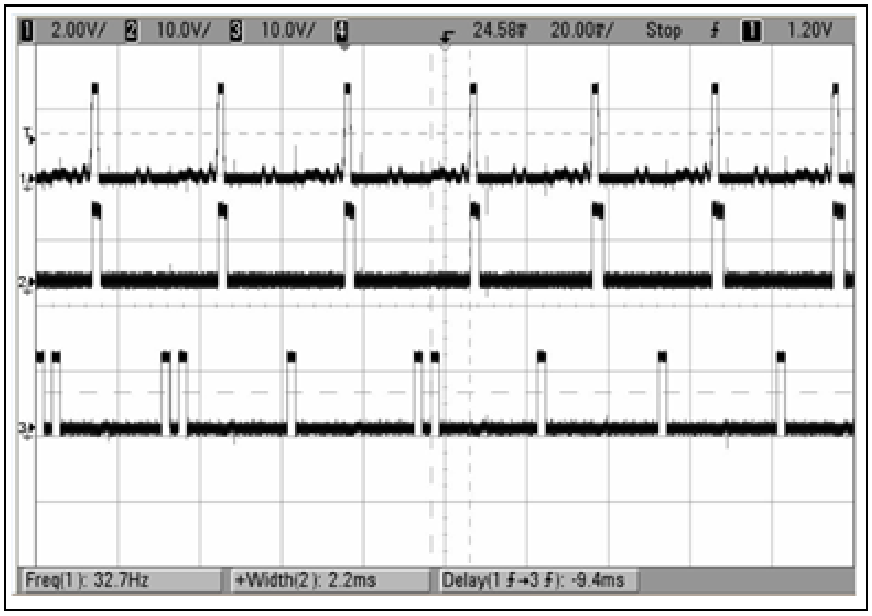
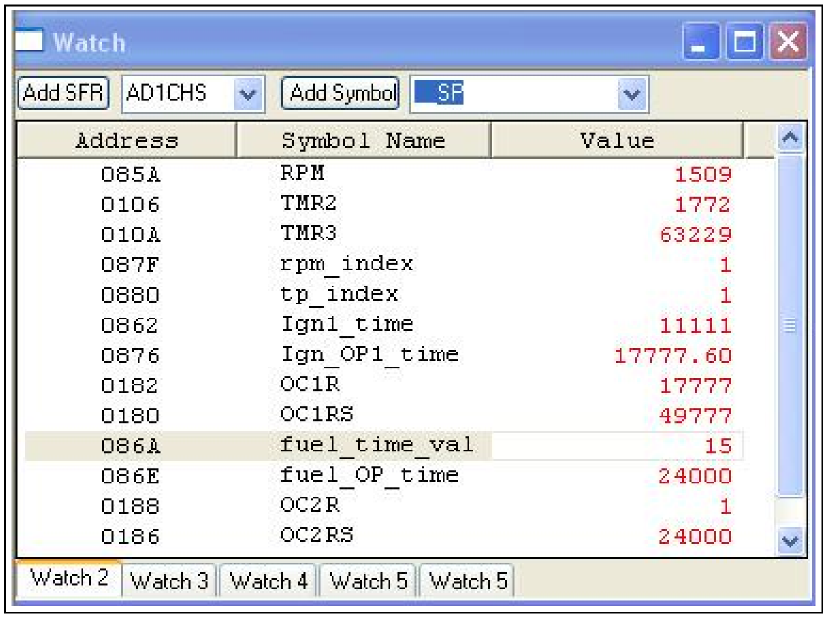


Figure 5.3 Ignition signal interference

The full testing of the output driver hardware was carried out once the software was developed as this provided the appropriate timing of the input and output signals as well as the voltage levels to and from the microcontroller.

5.2 Software testing

The validation of the software was done via the MPLAB SIM debugger and Watch windows. Software breakpoints were inserted in various sections of the code to confirm functionality. A clock stimulus was generated and assigned to the input pin to represent the appropriate crankshaft signal for engine speed.



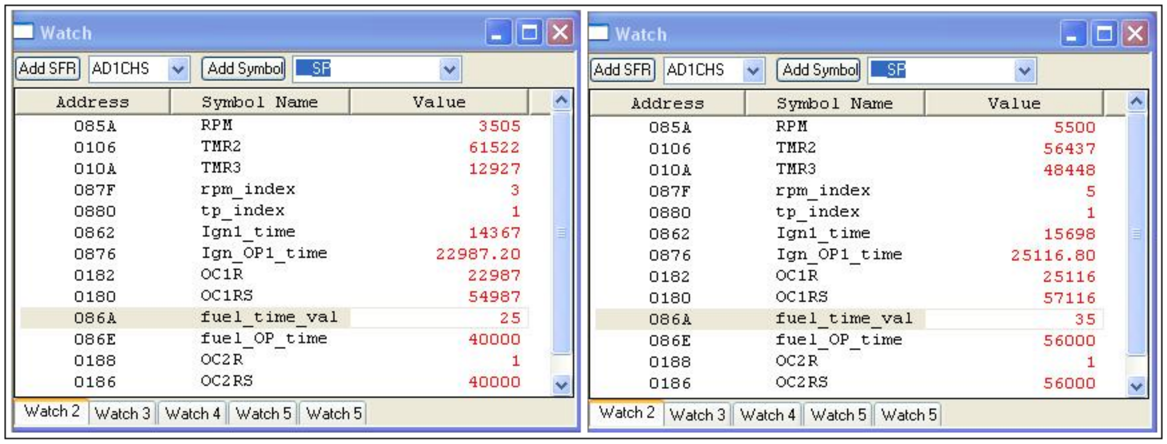
Address	Symbol Name	Value
085A	RPM	1509
0106	TMR2	1772
010A	TMR3	63229
087F	rpm_index	1
0880	tp_index	1
0862	Ign1_time	11111
0876	Ign_OP1_time	17777.60
0182	OC1R	17777
0180	OC1RS	49777
086A	fuel_time_val	15
086E	fuel_OP_time	24000
0188	OC2R	1
0186	OC2RS	24000

Figure 5.4 MPLAB software watch window

Initially all main registers – timers, and rpm were closely monitored for correct values according to the calculated values. From the oscillator speed defined and timer register configurations, it was possible to calculate each the timer incremental time as well as overflow time for each of the timers used. From these values it was possible to estimate what the register values for each will be for the desired conditions.

Once the main timer registers and the rpm register were verified of their correct operation and values, specific conditions were set in software for engine speed and load to confirm the indexing of the map arrays for the fuel injection and ignition timing.

In Figure 5.5 below, the left side watch window shows a generated rpm of 3505 with the corresponding rpm_index of 3, and in the right side of Figure 5.5 the rpm of 5005 shows an rpm_index of 5. This confirms the functionality of the map indexing.



```
Address   Symbol Name      Value
085A      RPM               3505
0106      TMR2             61522
010A      TMR3             12927
087F      rpm_index            3
0880      tp_index             1
0862      Ign1_time        14367
0876      Ign_OP1_time  22987.20
0182      OC1R             22987
0180      OC1RS            54987
086A      fuel_time_val       25
086E      fuel_OP_time     40000
0188      OC2R                 1
0186      OC2RS            40000
```

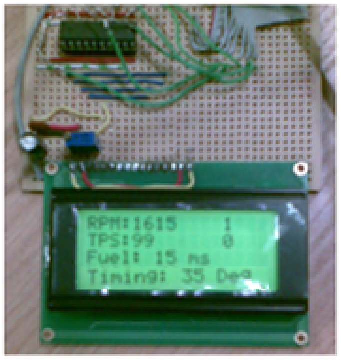
```
Address   Symbol Name      Value
085A      RPM               5500
0106      TMR2             56437
010A      TMR3             48448
087F      rpm_index            5
0880      tp_index             1
0862      Ign1_time        15698
0876      Ign_OP1_time  25116.80
0182      OC1R             25116
0180      OC1RS            57116
086A      fuel_time_val       35
086E      fuel_OP_time     56000
0188      OC2R                 1
0186      OC2RS            56000
```

Figure 5.5 MPLAB software simulation

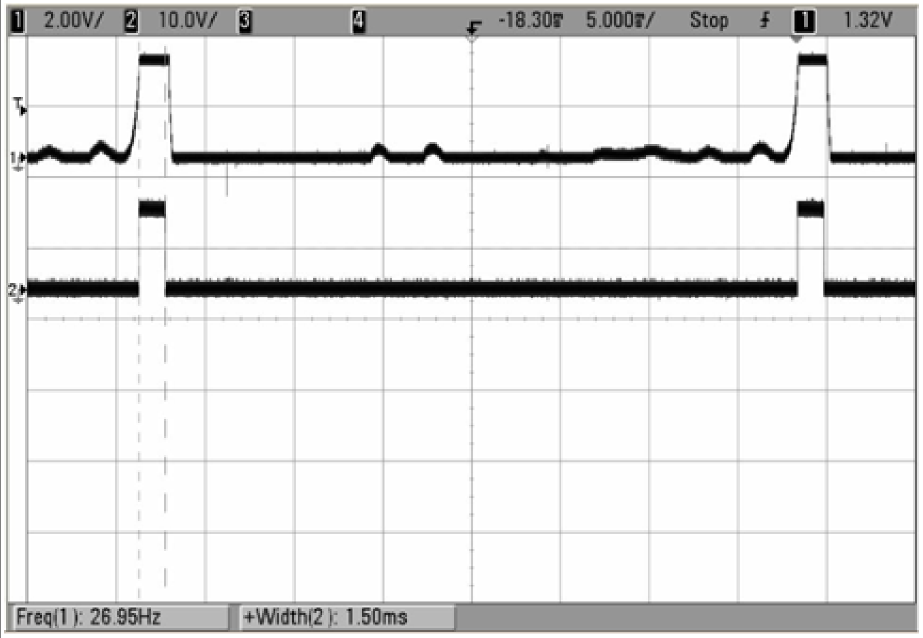
Based on the rpm and tp_index, specific fuel injection and ignition timing values from the map array of each were then verified. The output compare module register values can be noticed to change in value based on different index conditions. Concluding these simulation tests, the software functionality proved to working.

5.3 Engine testing

5.3.1 Test Condition 1: Engine at idle speed



Engine tuning data on LCD



Oscilloscope screen shot

Figure 5.6 Test condition 1.0

The LCD displays engine rpm as 1615 and its corresponding array index as 1, the TPS has a raw value of 99 and as per the calibration is the code in Appendix D corresponds to an index of 0.

The specific load site in the map array for the load condition of 1;0 is given as fuel injection of 15 milliseconds. Signal 1 (top) is given as the crankshaft signal and signal 2 (bottom) as the fuel Injector signal.

The rpm of 1615 as a frequency with reference to Table 3.2 would relate to 26.91 Hz which is validated on the oscilloscope screen shot. As per the trigger patters explained in chapter 3, the fuel injection takes place on each signal received from the crankshaft sensor, twice per crankshaft revolution.

From the given load map, 15 milliseconds of injection time (signal 2) is validated on every crank input pulse on the oscilloscope. The markers also indicate an overlay of the injection signal taking place on the crank input signal.

Figure 5.7 validates the appropriate fuel injection on a different quantity of
2.3 milliseconds.

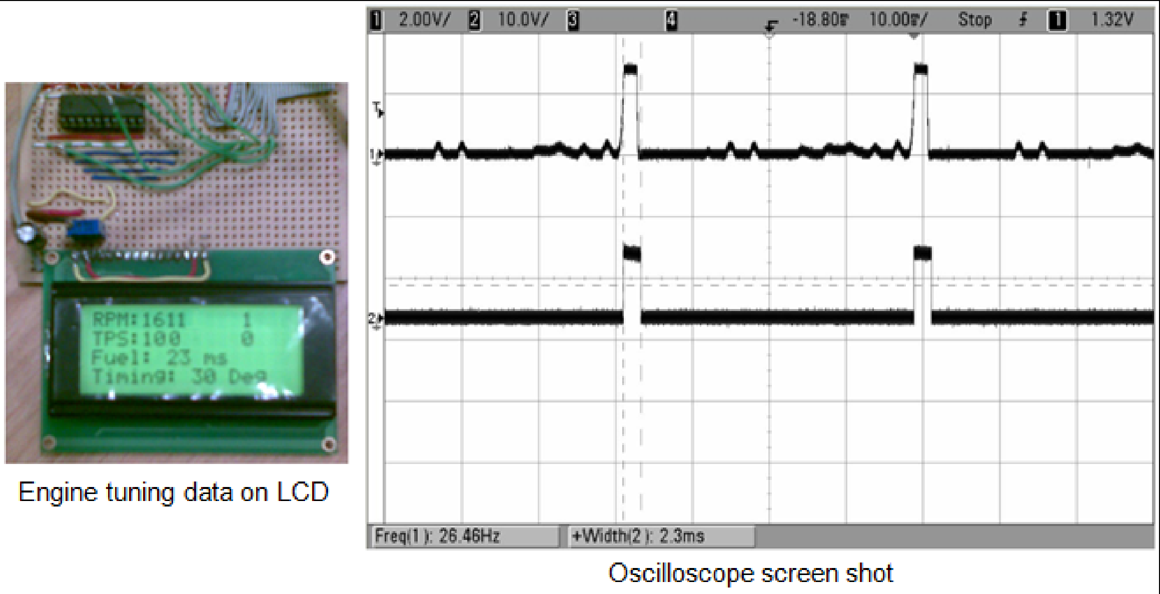


Figure 5.7 Test condition 1.1

Under the conditions as Test 1.0, the map array for the ignition timing is 35 degrees. Ignition timing is essentially the degrees (time span) before TDC that the ignition coils are energized to ignite the air-fuel mixture in the combustion chamber. As defined in chapter 3, the crankshaft and camshaft triggers take place at 60 degrees BTDC thus allowing the microcontroller adequate time to carry out the necessary calculations and trigger the outputs at the required time.

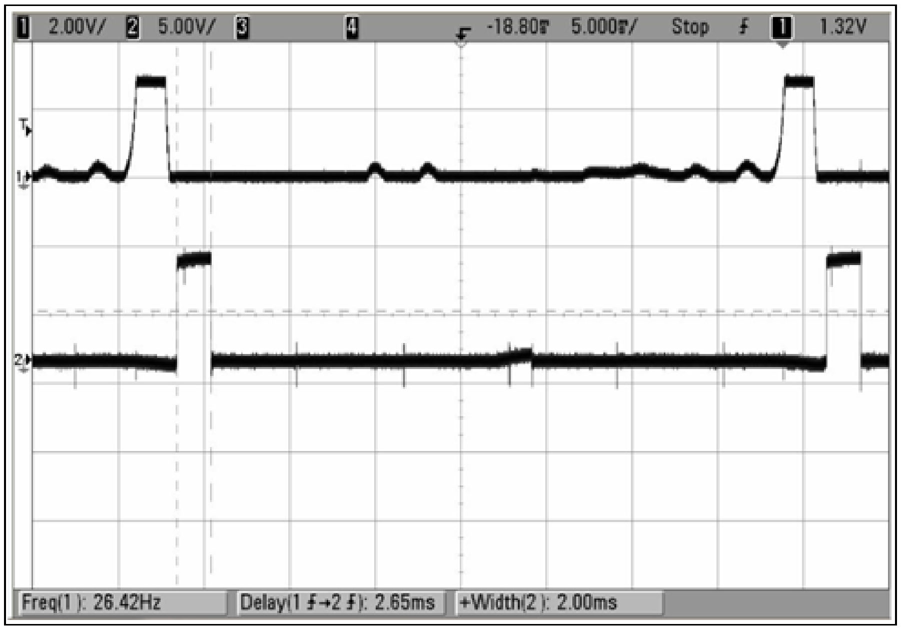


Figure 5.8 Test condition 1.2

In Figure 5.8 above, signal 1 (top) is the crankshaft sensor and signal 2 is the Ignition output signal for cylinder 1 and 4. From equation 3 for ignition timing given in Chapter 3,

$$t = \frac{\theta_{spk}}{360} * \frac{60\sec s}{rpm}$$

Substituting the screen capture conditions (factoring in the 60 degrees trigger)

$$t = \frac{(60-35)}{360} * \frac{60}{1585}$$

t = 2.63 milliseconds

The above calculation verifies the software functionality that at an engine speed of 1585 rpm and ignition map timing of 35 degrees, the ignition output was triggered at 2.63 milliseconds from trigger as per calculation. The ignition timing was then measured on the engine with the ignition timing light.

A further explanation to Figure 5.8 is the pulse with of the ignition signal, given as 2 milliseconds in the screen shot. This is what is referred to as the ignition coil dwell time. It is essentially the time taken for the ignition coil to fully charge and allow an effective spark discharge off the spark plug. This value is varied from engine to engine as well as from coil manufacturer. In addition higher voltage, longer spark duration (about 2ms) has been found to extend the engine operating conditions over which satisfactory ignition is achieved [4].

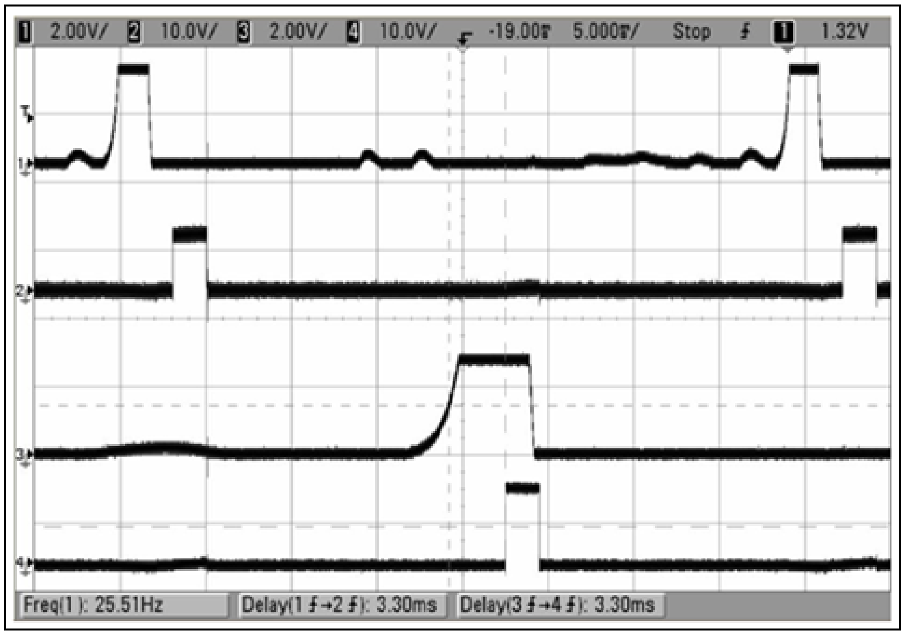


Figure 5.9 Test condition 1.3

Figure 5.9 above illustrates signal 1 as the crankshaft sensor, signal 2 as Ignition 1 (cylinder 1 and 4), signal 3 as camshaft sensor and signal 4 as Ignition 2 (cylinder 2 and 3). Reference is made to test condition 1.1 with map ignition timing of 30 degrees.

Figure 5.9 verifies ignition 1 (cylinder 1 and 4) in synchronous with ignition 2 (cylinder 2 and 3), both taking place 3.3 milliseconds after their respective input. The waveforms also validate the phase difference between crankshaft (1) and camshaft (3) signals due to the camshaft rotating at half the speed of the crankshaft.

5.3.2 Test Condition 2: Engine load test, 3000rpm

As described in chapter 2 and 4, the engine dyno places an adjustable load onto the engine to be able to evaluate the performance of the engine. Various parameters are closely monitored and adjustments are made to the ems to improve on the efficiency of the engine. The Honda CBR600 engine used in this project is from a motorcycle and thus has the gearbox built in as part of the engine casing. As seen in Figure 3.1 the output shaft of the gearbox is connected to the dyno and as a result the shaft speed recorded by the speed sensor Figure 2.11 is a function of the ratios in the gearbox. To calculate back for engine speed, the speed sensor values need to be multiplied by the appropriate ratios as per Table 3.1. The engine load test was carried out in 3rd gear and so the shaft speed is multiplied by 1.611 (3rd gear ratio) and then by 2.111 (Primary drive ratio) to obtain the engine speed during the testing sequence.

Figure 5.10 is the data acquired from the load dyno test carried out with the prototype engine management system developed in this project. The respective columns are given as,

Speed (rpm) – dyno input shaft rotational speed

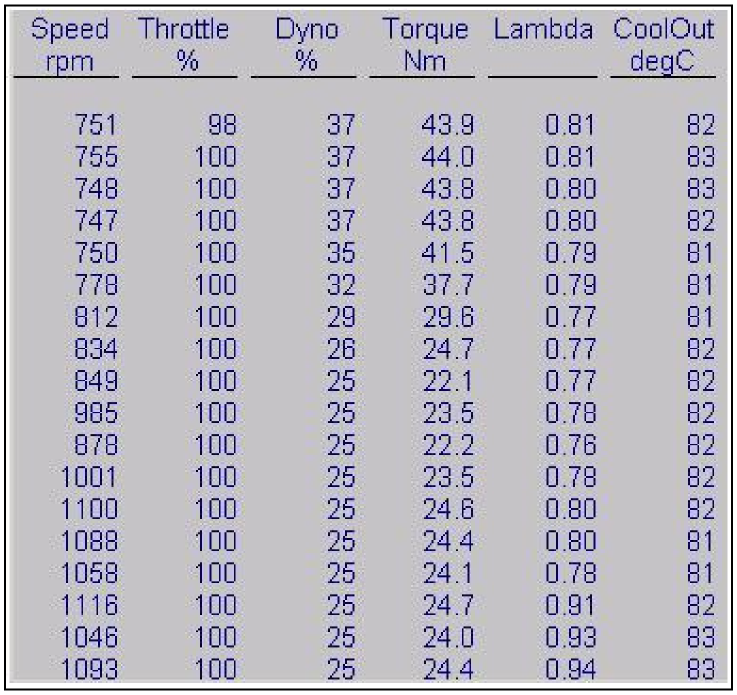
Throttle (%) – the percentage amount of opening of the throttle valve controlling the air flow into the engine

Dyno (%) – the amount of load placed onto the engine by the dyno

Torque (Nm) – this is the calculated turning force exerted by the engine under load condition

Lambda – the air-fuel measurement in the exhaust

CoolOut (degC) – the water temperature of the engine cooling system



Speed rpm	Throttle %	Dyno %	Torque Nm	Lambda	CoolOut degC
751	98	37	43.9	0.81	82
755	100	37	44.0	0.81	83
748	100	37	43.8	0.80	83
747	100	37	43.8	0.80	82
750	100	35	41.5	0.79	81
778	100	32	37.7	0.79	81
812	100	29	29.6	0.77	81
834	100	26	24.7	0.77	82
849	100	25	22.1	0.77	82
985	100	25	23.5	0.78	82
878	100	25	22.2	0.76	82
1001	100	25	23.5	0.78	82
1100	100	25	24.6	0.80	82
1088	100	25	24.4	0.80	81
1058	100	25	24.1	0.78	81
1116	100	25	24.7	0.91	82
1046	100	25	24.0	0.93	83
1093	100	25	24.4	0.94	83

Figure 5.10 Engine Dyno test data

As can be seen from the column of throttle %, full load is achieved at 100% throttle. Since the engine under test has a very small capacity, 600cc, only 25% of the dyno load is required at this engine speed. To reach a target engine speed for specific load testing, as per the outline of this project – 3000rpm, the sequence is as follows. The throttle % is increased to allow engine speed to increase, and then the dyno % load is then increased to load the engine and reach the desired engine speed. This process is continued until the throttle % has reached 100%, full load condition. The dyno % load is then finely adjusted to the required engine speed.

At full load, 100% throttle, is where the parameters for the load condition can be adjusted to get the desired output of the engine - performance, fuel economy or emission control. All these output conditions are primarily dependent of the lambda values as described in chapter 4.

Multiple load tests were carried out for the purpose of this research, with adjustments on both fuel injection and ignition timing and various recordings of lambda values were obtained thus resulting in a variance engine torque responses. Due to malfunctioning of some of the sensors on the university measurement system the corrected power figures were unobtainable but instead, the direct torque readings from the load cell representing the engine force will be referenced to as an engine output measurement.

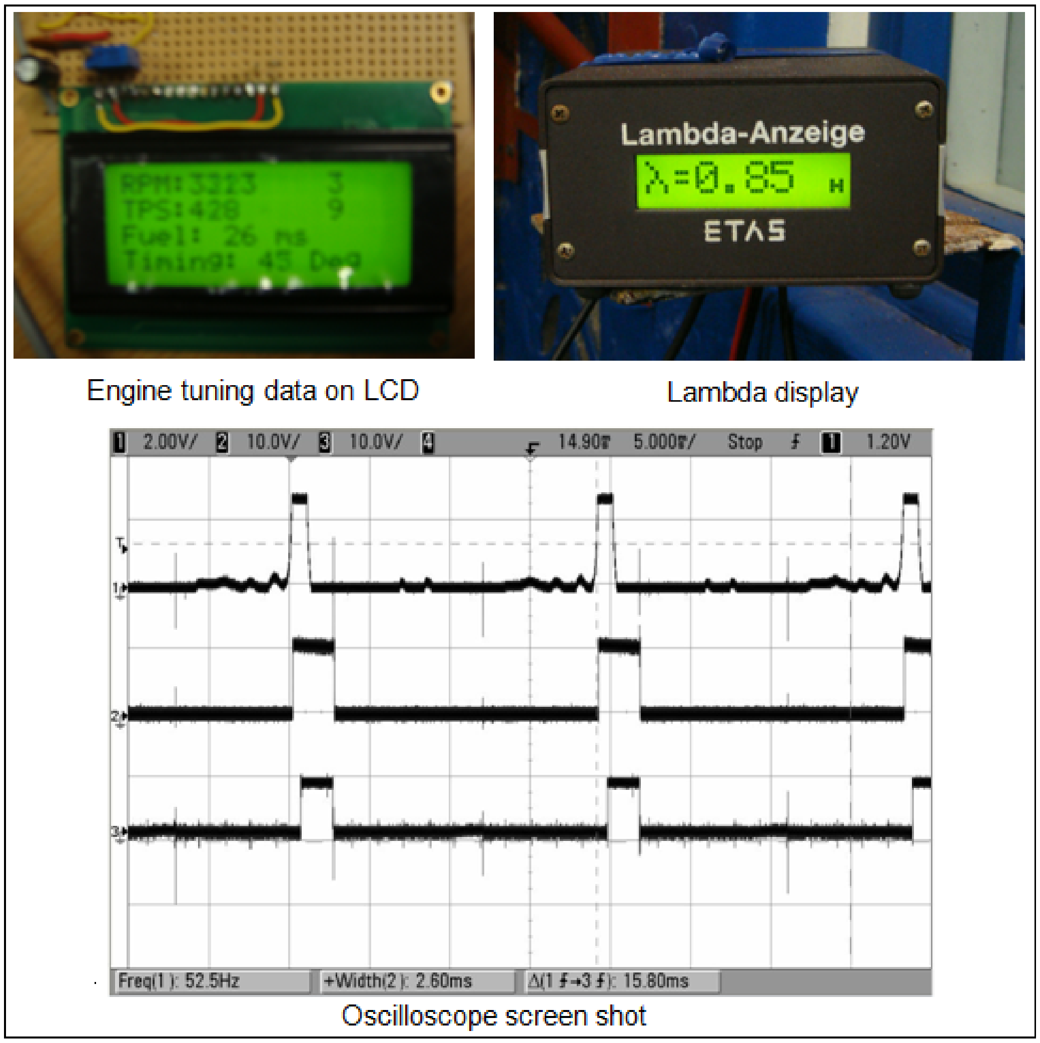


Figure 5.11 Full load test condition

5.3.3 Full load test 1

Shaft speed rpm	Throttle %	Dyno load %	Torque Nm	Lambda	Coolant Out degC	Engine speed rpm
751	98	37	43.9	0.81	82	2554
755	100	37	44.0	0.81	84	2568
748	100	37	43.8	0.80	83	2544
747	100	37	43.8	0.80	82	2540
750	100	35	41.5	0.79	81	2551
778	100	32	37.7	0.79	81	2646
812	100	29	29.6	0.77	81	2761
834	100	26	24.7	0.77	82	2836
849	100	25	22.1	0.77	82	2887
985	100	25	23.5	0.78	82	3350
878	100	25	22.2	0.76	82	2986
1001	100	25	23.5	0.78	82	3404
1100	100	25	24.6	0.80	82	3741
1088	100	25	24.4	0.80	81	3700
1058	100	25	24.1	0.78	81	3598
1116	100	25	24.7	0.91	82	3795
1046	100	25	24.0	0.93	83	3557
1093	100	25	24.4	0.94	83	3717
1154	100	25	25.1	0.86	83	3925
1149	100	25	25.1	0.87	83	3908
1092	100	25	24.3	0.9	83	3714
1146	100	25	25.0	0.93	83	3897
1127	100	25	24.7	0.91	84	3833
1123	100	26	25.2	0.89	84	3819

Table 5.1 Engine load test 1

In Table 5.1 above, the calculated engine speed is given on the extreme right side column. As described in chapter 4, during the tuning process it is easier to start off with rich air-fuel mixtures and then lean out towards the desired conditions. This strategy can be seen in the column of lambda display. Specific air-fuel ratios and additional ignition timing actually promote combustion and thus increase engine speed; this can be explained for the changes in engine speed based on the same throttle opening and dyno load.

5.3.4 Full load test 2

Shaft speed rpm	Throttle %	Dyno load %	Torque Nm	Lambda	Coolant Out degC	Engine speed rpm
723	97	39	47.3	0.81	80	2459
724	100	39	47.3	0.81	81	2462
696	100	39	46.6	0.81	81	2367
734	100	39	47.6	0.81	81	2496
788	100	39	49.3	0.79	80	2680
846	100	39	50.7	0.93	80	2877
831	100	39	50.4	1.08	79	2826
832	100	39	50.4	0.92	79	2829
766	100	39	48.7	0.92	79	2605
814	100	39	50.0	0.93	80	2768
815	100	39	50.1	0.92	80	2772
817	100	39	50.1	0.92	81	2778
819	100	39	50.1	0.92	82	2785
824	100	38	48.9	0.92	82	2802
843	100	35	44.3	0.92	81	2867
868	100	33	38.7	0.91	80	2952
871	100	33	38.2	0.90	80	2962
872	100	33	38.2	0.89	80	2966

Table 5.2 Engine load test 2

In the second load test carried out illustrated in Table 5.2, further adjustments were made to the fuel injection quantities and ignition timing. The results revealed that with leaner air-fuel ratio levels given by the lambda readings recorded, there was an increase in the torque output of the engine.

5.3.5 Full load test 3

Shaft speed rpm	Throttle %	Dyno load %	Torque Nm	Lambda	Coolant Out degC	Engine speed rpm
768	95	39	62.0	1.10	83	2612
769	100	39	62.0	0.96	83	2615
762	100	39	61.7	0.87	83	2591
784	100	39	62.2	1.00	83	2666
780	100	39	62.0	1.02	83	2653
776	100	39	62.0	0.92	84	2639
782	100	39	62.0	1.05	84	2659
784	100	38	61.8	1.04	83	2666
1053	91	0	25.1	0.97	84	3581

Table 5.3 Engine load test 3

The third test load test revealed the highest torque output from the engine. The highest value of 62Nm was achieved at lambda values ≈ 1 as shown in Table 5.3. These results are specific for this engine design construction as well as the fuel rating used, 95 octane unleaded fuel. With higher octane rating fuel, additional ignition timing can be used which will result in improved torque output of the engine.

5.2 Conclusion

This master’s thesis was carried out for a programmable engine management system for fuel injection and ignition timing control. The proposal outlined the use of a water temperature sensor, as researched. During the first testing of the prototype it was found that with the correct amount of fuel and ignition timing it was possible to start the engine under cold start, and hot start. It can be deduced that the water temperature sensor is generally used for extreme temperature conditions where additional fuel compensation is required in cold temperatures and less fuel required for hot conditions.

Chapter 2 provided an overview of the components for engine control. The functions and operation of each component was extensively discussed. Chapter 3 focused on the experimental setup of the work carried out. The hardware and software concept was also discussed and explained. Chapter 4 provided a short insight into engine tuning. Each engine tuner has an individual method of engine tuning but essentially the same targets are achievable.

Chapter 5 has outlined the tests carried out and the results obtained reveal that a programmable engine management system hardware, and software design was developed and proved functional to control fuel injection and ignition timing parameters on a Honda CBR 600 engine.

By having access to control these parameters of the engine, future developments can include additional controls such as traction control system, advanced gear shifting, launch control.

References

[1] SAE International, (*2008), Formula SAE Rules,* SAE International USA

[2] Richard van Basshusan, et al, (2006), *Modern Engine Technology from A to Z*, SAE International

[3] Ronald K. Jurgen, (1999), *Automotive Electronics Handbook*, Mc Graw-Hill Inc.

[4] Greg Banish, (2007), *Engine Management Advance Tuning*, Brooklands Books Ltd

[5] John B. Heywood, (1988), *Internal Combustion Engine Fundamentals*, Mc Graw-Hill Book Company

[6] Lucio Di Jasio, (2008), *Programming 32-bit microcontrollers in C*, Newnes Publication

[7] John B Peatman, (1997), *Design with PIC microcontrollers*, Prentice-Hall Inc

[8] Tom Denton, (2004), *Automobile Electrical and Electronic Systems*, Butterworth Heinemann

[9] Jeff Hartman, (2003), *How to Tune and Modify Engine Management Systems*, MBI Publishing Company

[10] Mario Farrugia, et al., (2005), *On the Use of a Honda 600cc 4-Cylinder Engine for Formula SAE Competition*, SAE paper 2005-01-0025

[11] Santi Udomkesmalee, et al., *Evolution and Design of the 2003 Cornell University Engine Control Module for an FSAE Racecar*, SAE paper 2004-01-0425

[12] Pedro Kulzer, (2005), *Bosch MS3.1 Function Sheet*, Robert Bosch GmbH

[13] Robert Bosch GmbH, (2008), *MS 3 Sport User Manual*, Robert Bosch GmbH

[14] Johan Eriksson, et al., (2007), *Distributed Engine Management System for Formula SAE*, SAE paper 2007-01-1602

[15] Electromotive Inc., (2003), *TEC3 Manual Version 1.7*, Electromotive, Inc.

[16] www.instronics.com/sensoronix_variable_reluctance_speed_sensor.html, (2010), *Variable Reluctance Speed Sensor,* Instronics

[17] Kevin Sullivan, 2008, *Toyota automotive technical training series*, Toyota Motor Sales, USA

[18] Larry Carley, 2008, *AA1 Car Auto Diagnosis*, www.aa1car.com/library

[19] Microchip, (2007), *MPLAB C30 C Complier user's guide*, Microchip Technology Inc.

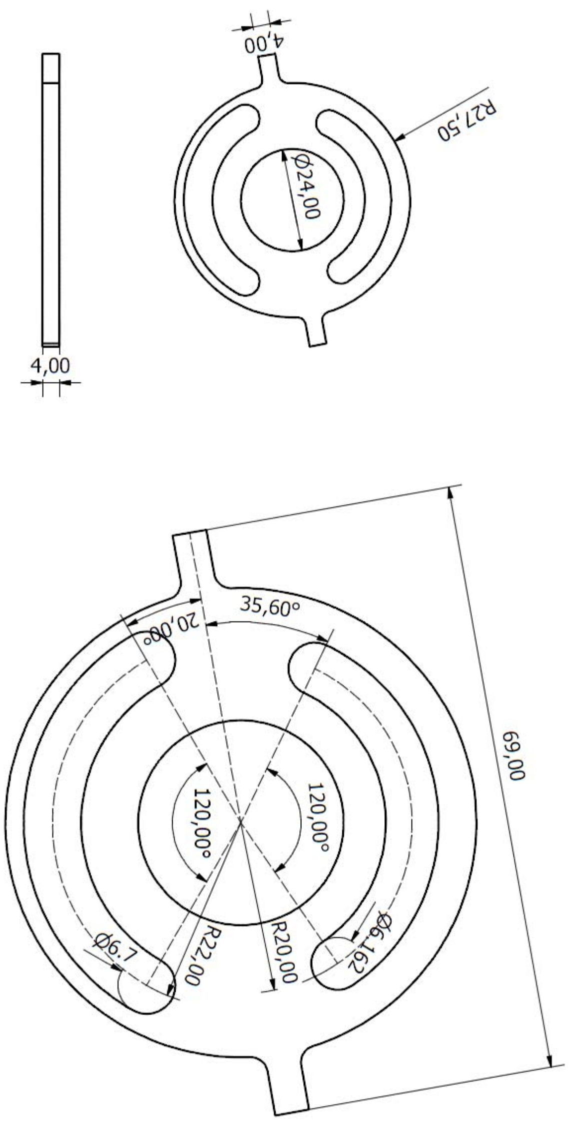
[20] Microchip (2003), *PICmicro CCP and ECCP Tips n Tricks*, Microchip Technology Inc.

[21] Microchip (2006), *PIC24FJ128GA Family Data Sheet*, Microchip Technology Inc.

[22] Microchip (2005), *Explorer 16 Development Board User's Guide*, Microchip Technology Inc.

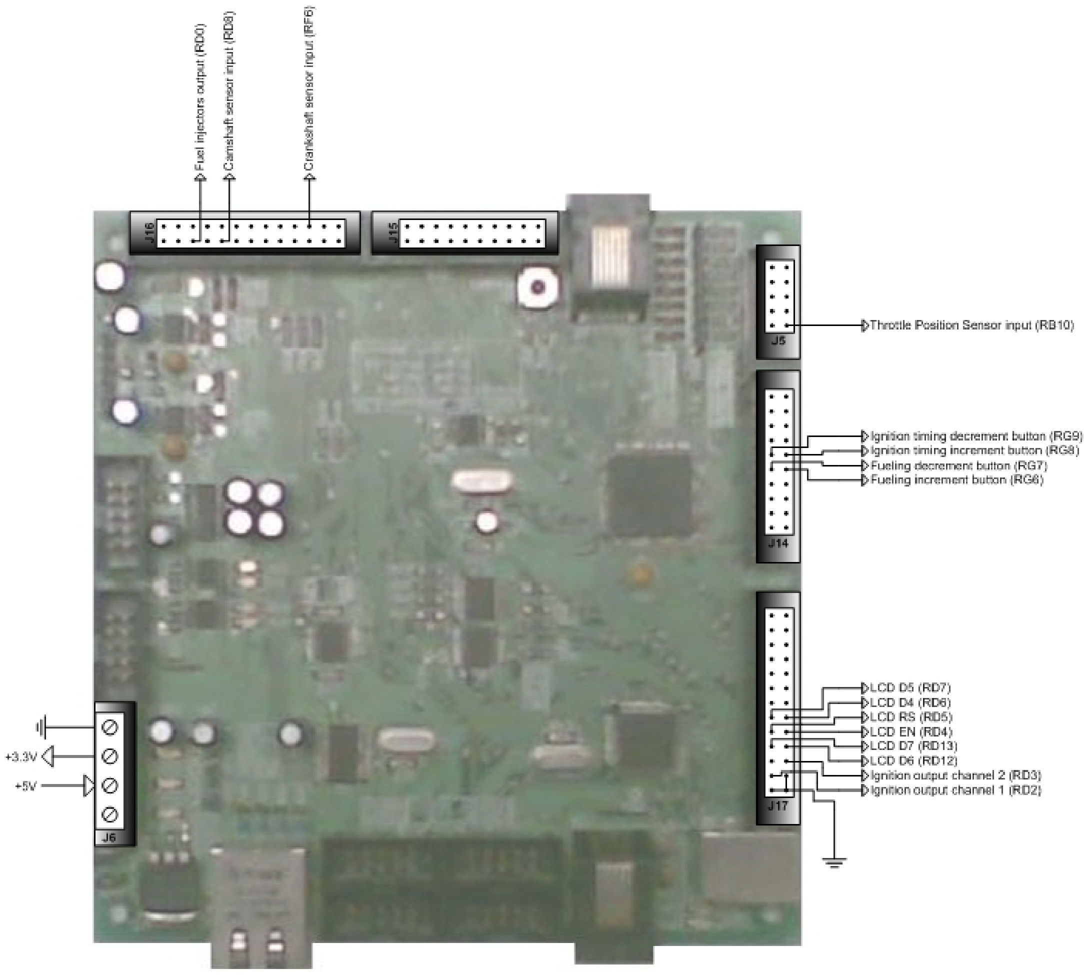
[23] Frank Adlam, Riaan Ehlers, et al, (2008), *Multi IO PIC24F board schematic*, Nelson Mandela Metropolitan University

Appendix A - Camshaft sensor trigger wheel drawing

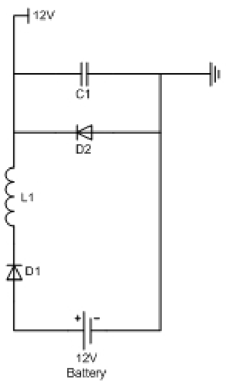


Appendix B – Hardware Schematics

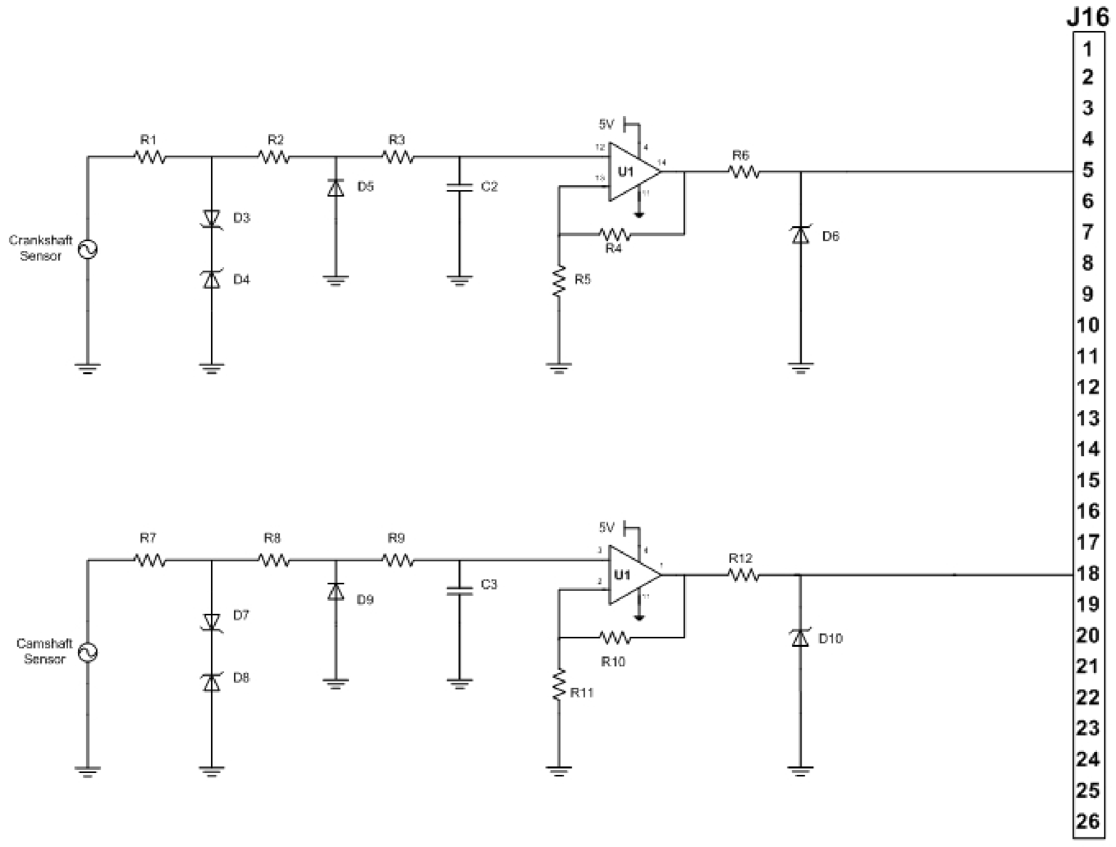
PIC24FJ128GA010 Multi IO Board

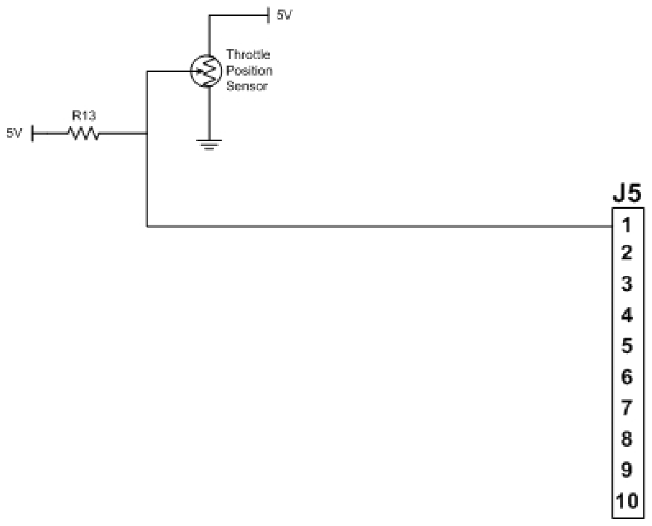

Fuel injectors output (RD0)
Camshaft sensor input (RD8)
Crankshaft sensor input (RF6)
J16
J15
J5
Throttle Position Sensor input (RB10)
Ignition timing decrement button (RG9)
Ignition timing increment button (RG8)
Fueling decrement button (RG7)
Fueling increment button (RG6)
J14
LCD D5 (RD7)
LCD D4 (RD6)
LCD RS (RD5)
LCD EN (RD4)
LCD D7 (RD13)
LCD D6 (RD12)
Ignition output channel 2 (RD3)
Ignition output channel 1 (RD2)
J17
+3.3V
+5V
J6

12V supply

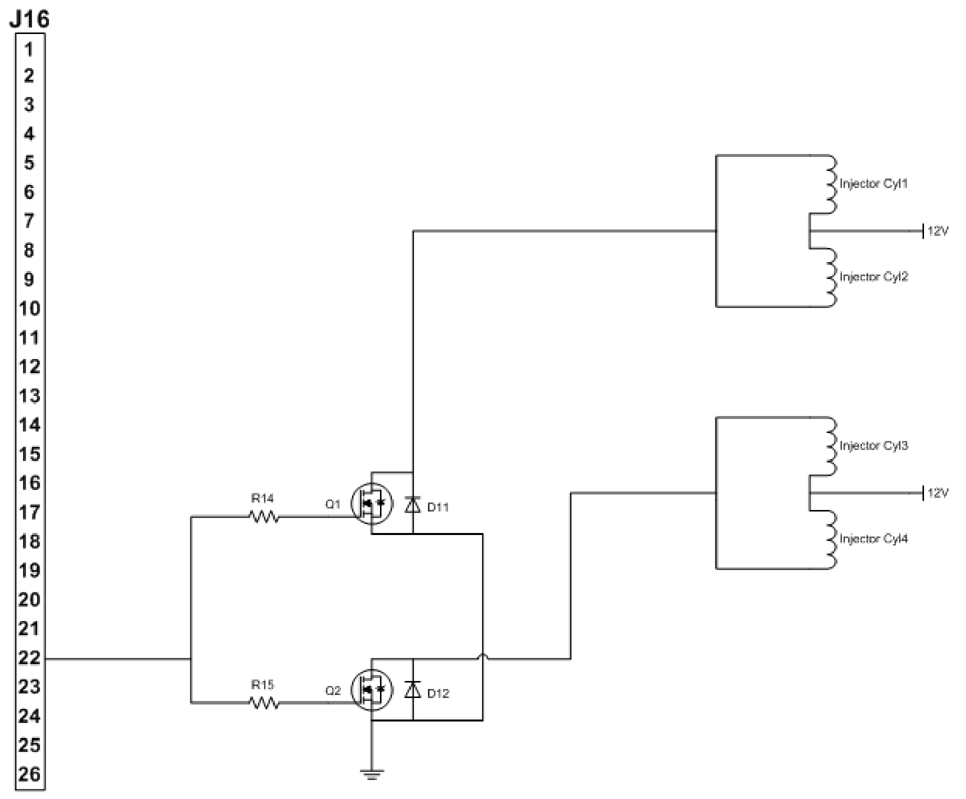


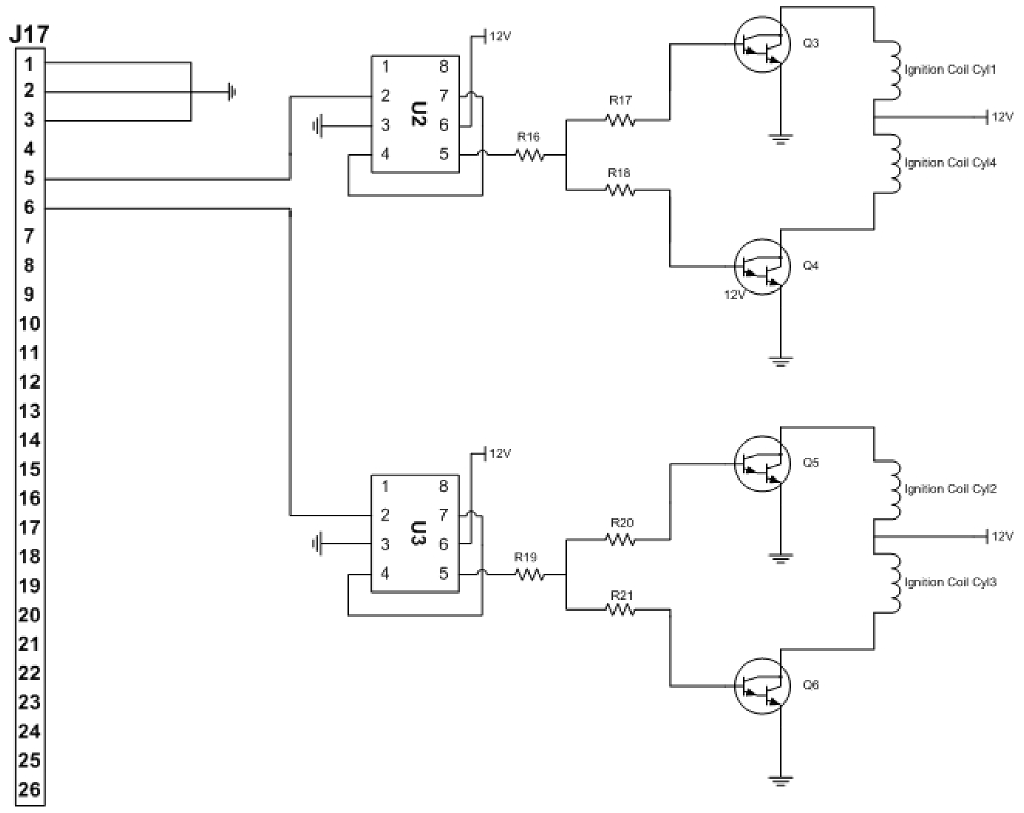
Input signal conditioning



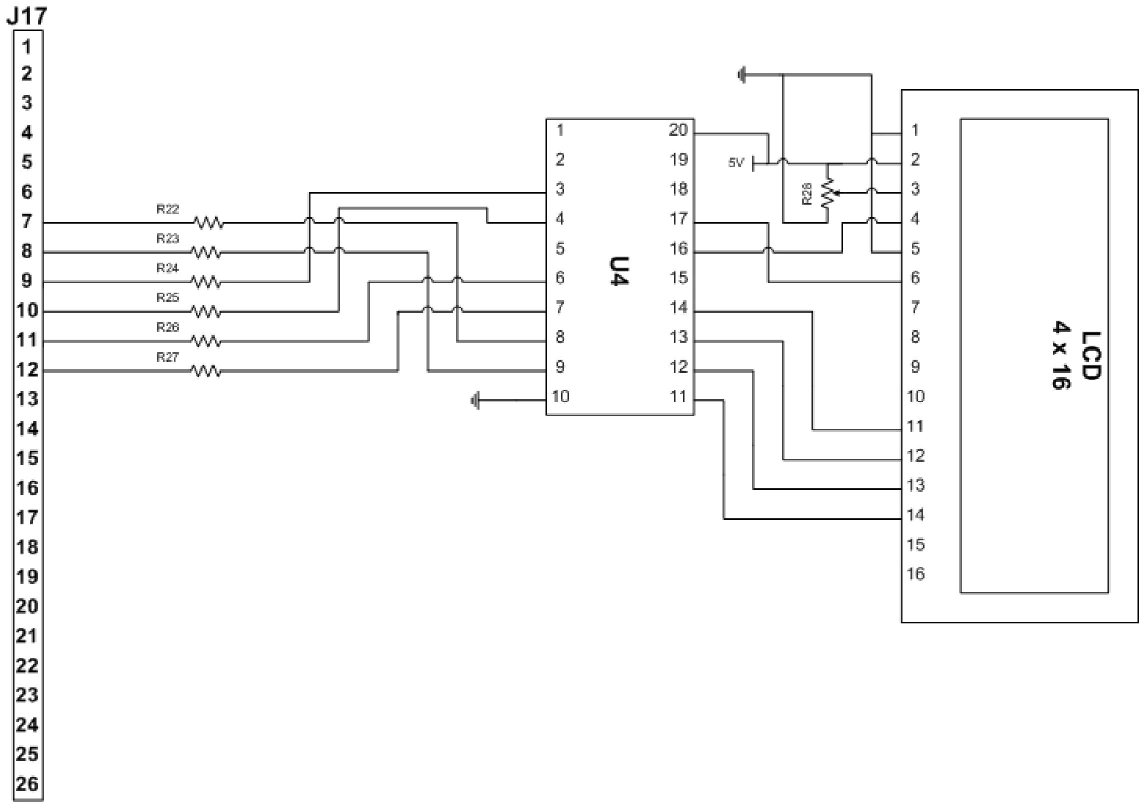


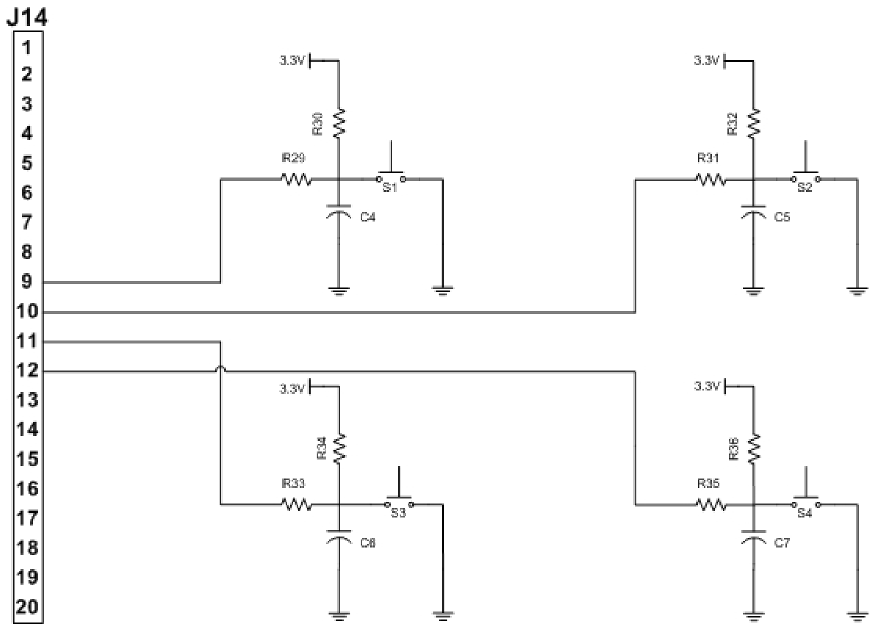
Output signals



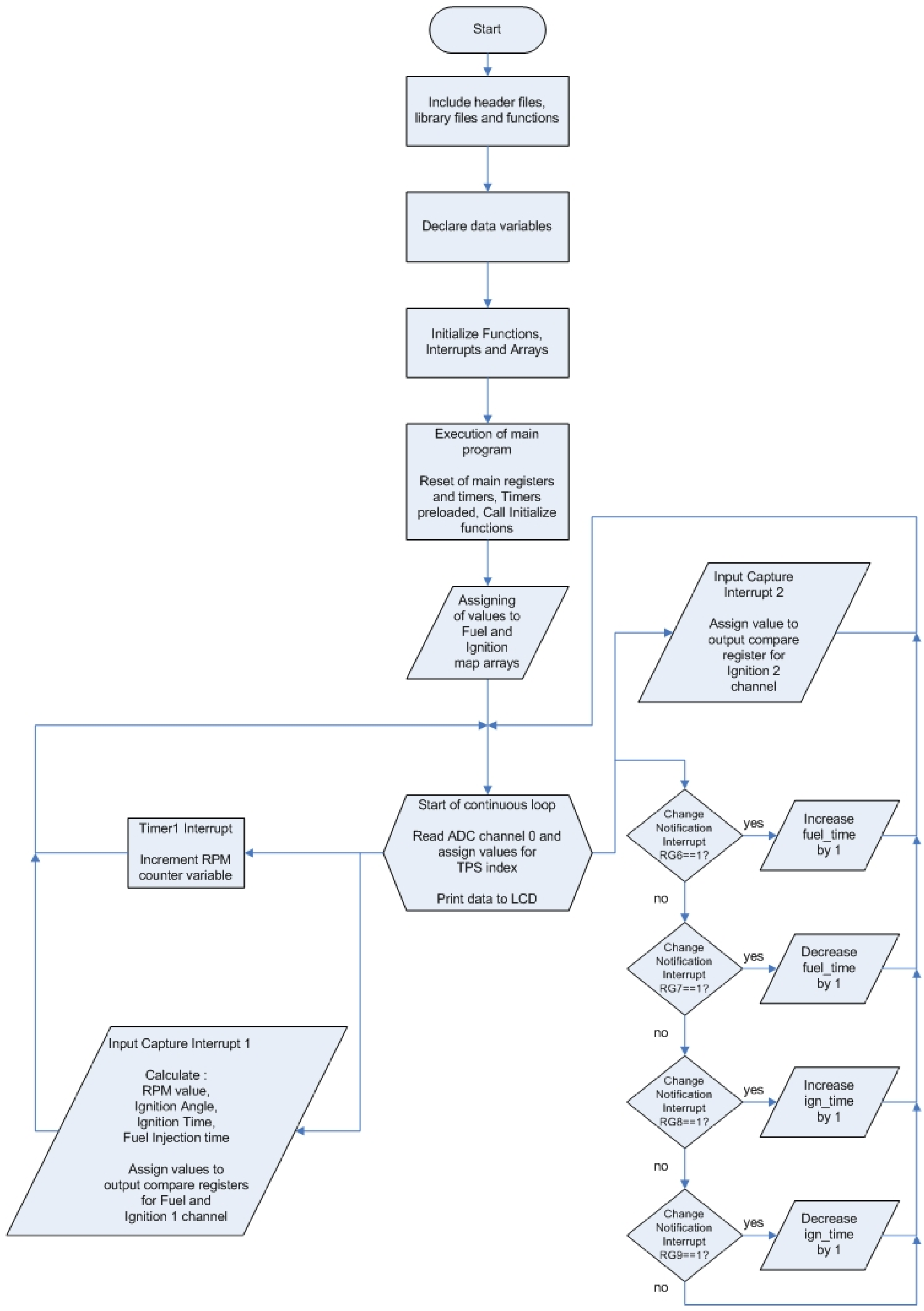


User display and controls





Appendix C - Software flowchart


Start
Include header files, library files and functions
Declare data variables
Initialize Functions, Interrupts and Arrays
Execution of main program
Reset of main registers and timers, Timers preloaded, Call Initialize functions
Assigning of values to Fuel and Ignition map arrays
Input Capture Interrupt 2
Assign value to output compare register for Ignition 2 channel
Start of continuous loop
Read ADC channel 0 and assign values for TPS index
Print data to LCD
Timer1 Interrupt
Increment RPM counter variable
Input Capture Interrupt 1
Calculate : RPM value, Ignition Angle, Ignition Time, Fuel Injection time
Assign values to output compare registers for Fuel and Ignition 1 channel
Change Notification Interrupt RG6==1?
yes
Increase fuel_time by 1
no
Change Notification Interrupt RG7==1?
yes
Decrease fuel_time by 1
no
Change Notification Interrupt RG8==1?
yes
Increase ign_time by 1
no
Change Notification Interrupt RG9==1?
yes
Decrease ign_time by 1
no

Appendix D - Software code

```
#include <p24FJ128GA010.h>
#include "stdio.h"
#include "stdlib.h"

#define OSC_SPEED OSC_32MHz         // 8MHz crystal x 4 (PLL) = 32MHz
#include "delay_ms.c"
#include "delay_us.c"

//Set Configuration bits
_CONFIG1(JTAGEN_OFF & GCP_OFF & GWRP_OFF & FWDTEN_OFF)        // JTAG off, watchdog timer off
_CONFIG2 (FNOSC_PRIPLL & POSCMOD_XT)

#define True  1
#define False 0

#include "lcd.c"                    //Ver 2.0
#include "itoa.c"
#include "ADC1.h"                   //ver 1.20

int overflow_countRPM, RPM, a, i;
unsigned int start_timeRPM, end_timeRPM;
long period, one_rev_time, Ign1_time, fuel_OP_time, Ign_angle;
float RPS, Ign_OP1_time;
char rpm_index, tp_index;
char St[10]="";

void Init_PIC(void);                // Initialize the PIC
void Init_Ext_Int(void);            // Initialize external interupt
void Init_Timer1(void);             // Initialize Timer1 function
void Init_Timer3(void);             // Initialize Timer3 function
void Init_IP_Capture1(void);        // Initialize Input Capture Module 1 function
void Init_OP_Compare1(void);        // Initialize Output Compare Module 1 function
void Init_ADC(void);
void Init_CN(void);



//          [rpm][tps]
//          [row][col]                      Col0 Col1 Col2 Col3...10
char ign_time[13][10];              // Row0
                                    // Row1
                                    // Row2
                                    // Row3...13

//          [rpm][tps]
//          [row][col]                      Col0 Col1 Col2 Col3...10
char fuel_time[13][10];             // Row0
                                    // Row1
                                    // Row2
                                    // Row3...13
//        [rpm][tps]                Array for fuel map
    fuel_time[12][9] = 27;          // 12 - 12999rpm fuel load site
    fuel_time[12][8] = 26;
    fuel_time[12][7] = 21;
    fuel_time[12][6] = 22;
    fuel_time[12][5] = 20;
    fuel_time[12][4] = 20;
    fuel_time[12][3] = 20;
    fuel_time[12][2] = 19;
    fuel_time[12][1] = 20;
    fuel_time[12][0] = 16;

    fuel_time[11][9] = 27;          // 11 - 11999rpm fuel load site
    fuel_time[11][8] = 26;
    fuel_time[11][7] = 21;
    fuel_time[11][6] = 22;
    fuel_time[11][5] = 20;
    fuel_time[11][4] = 20;
    fuel_time[11][3] = 19;
    fuel_time[11][2] = 19;
    fuel_time[11][1] = 20;
    fuel_time[11][0] = 16;

    fuel_time[10][9] = 27;          // 10 -10999rpm fuel load site
    fuel_time[10][8] = 26;
    fuel_time[10][7] = 22;
    fuel_time[10][6] = 21;
    fuel_time[10][5] = 20;
    fuel_time[10][4] = 20;
    fuel_time[10][3] = 20;
    fuel_time[10][2] = 19;
    fuel_time[10][1] = 20;
    fuel_time[10][0] = 16;

    fuel_time[9][9] = 27;           // 9 - 9999rpm fuel load site
    fuel_time[9][8] = 26;
    fuel_time[9][7] = 21;
    fuel_time[9][6] = 22;
    fuel_time[9][5] = 20;
    fuel_time[9][4] = 20;
    fuel_time[9][3] = 20;
    fuel_time[9][2] = 19;
    fuel_time[9][1] = 20;
    fuel_time[9][0] = 16;

    fuel_time[8][9] = 27;           // 8 - 8999rpm fuel load site
    fuel_time[8][8] = 26;
    fuel_time[8][7] = 21;
    fuel_time[8][6] = 22;
    fuel_time[8][5] = 20;
    fuel_time[8][4] = 20;
    fuel_time[8][3] = 20;
    fuel_time[8][2] = 19;
    fuel_time[8][1] = 20;
    fuel_time[8][0] = 16;
```

```
fuel_time[7][9] = 59;           // 7 - 7999rpm fuel load site
fuel_time[7][8] = 59;
fuel_time[7][7] = 59;
fuel_time[7][6] = 59;
fuel_time[7][5] = 59;
fuel_time[7][4] = 58;
fuel_time[7][3] = 58;
fuel_time[7][2] = 58;
fuel_time[7][1] = 58;
fuel_time[7][0] = 58;

fuel_time[6][9] = 56;           // 6 - 6999rpm fuel load site
fuel_time[6][8] = 56;
fuel_time[6][7] = 56;
fuel_time[6][6] = 56;
fuel_time[6][5] = 56;
fuel_time[6][4] = 29;
fuel_time[6][3] = 56;
fuel_time[6][2] = 56;
fuel_time[6][1] = 56;
fuel_time[6][0] = 56;

fuel_time[5][9] = 54;           // 5 - 5999rpm fuel load site
fuel_time[5][8] = 54;
fuel_time[5][7] = 54;
fuel_time[5][6] = 54;
fuel_time[5][5] = 54;
fuel_time[5][4] = 54;
fuel_time[5][3] = 54;
fuel_time[5][2] = 54;
fuel_time[5][1] = 54;
fuel_time[5][0] = 54;

fuel_time[4][9] = 27;           // 4 - 4999rpm fuel load site
fuel_time[4][8] = 27;
fuel_time[4][7] = 27;
fuel_time[4][6] = 27;
fuel_time[4][5] = 27;
fuel_time[4][4] = 26;
fuel_time[4][3] = 26;
fuel_time[4][2] = 26;
fuel_time[4][1] = 26;
fuel_time[4][0] = 26;

fuel_time[3][9] = 26;           // 3 - 3999rpm fuel load site
fuel_time[3][8] = 26;
fuel_time[3][7] = 26;
fuel_time[3][6] = 26;
fuel_time[3][5] = 26;
fuel_time[3][4] = 25;
fuel_time[3][3] = 25;
fuel_time[3][2] = 25;
fuel_time[3][1] = 25;
fuel_time[3][0] = 25;
```

```
    fuel_time[2][9] = 26;          // 2 - 2999rpm fuel load site
    fuel_time[2][8] = 26;
    fuel_time[2][7] = 26;
    fuel_time[2][6] = 26;
    fuel_time[2][5] = 26;
    fuel_time[2][4] = 29;
    fuel_time[2][3] = 29;
    fuel_time[2][2] = 29;
    fuel_time[2][1] = 28;
    fuel_time[2][0] = 28;

    fuel_time[1][9] = 32;          // 1 - 1999rpm fuel load site
    fuel_time[1][8] = 32;
    fuel_time[1][7] = 31;
    fuel_time[1][6] = 31;
    fuel_time[1][5] = 30;
    fuel_time[1][4] = 23;
    fuel_time[1][3] = 23;
    fuel_time[1][2] = 23;
    fuel_time[1][1] = 23;
    fuel_time[1][0] = 23;

    fuel_time[0][9] = 43;          // 0 - 999rpm fuel load site
    fuel_time[0][8] = 43;
    fuel_time[0][7] = 43;
    fuel_time[0][6] = 43;
    fuel_time[0][5] = 43;
    fuel_time[0][4] = 43;
    fuel_time[0][3] = 43;
    fuel_time[0][2] = 40;
    fuel_time[0][1] = 40;
    fuel_time[0][0] = 44;

//        [rpm][tps]               Array for ingnition map
    ign_time[12][9] = 50;          // 12000-12999rpm ingition load site
    ign_time[12][8] = 50;
    ign_time[12][7] = 50;
    ign_time[12][6] = 50;
    ign_time[12][5] = 50;
    ign_time[12][4] = 50;
    ign_time[12][3] = 50;
    ign_time[12][2] = 50;
    ign_time[12][1] = 50;
    ign_time[12][0] = 50;

    ign_time[11][9] = 50;          // 11000-11999rpm ingition load site
    ign_time[11][8] = 50;
    ign_time[11][7] = 50;
    ign_time[11][6] = 50;
    ign_time[11][5] = 50;
    ign_time[11][4] = 50;
    ign_time[11][3] = 50;
    ign_time[11][2] = 50;
    ign_time[11][1] = 50;
    ign_time[11][0] = 50;
```

```
ign_time[10][9] = 50;           // 10000-10999rpm ingition load site
ign_time[10][8] = 50;
ign_time[10][7] = 50;
ign_time[10][6] = 50;
ign_time[10][5] = 50;
ign_time[10][4] = 50;
ign_time[10][3] = 50;
ign_time[10][2] = 50;
ign_time[10][1] = 50;
ign_time[10][0] = 50;

ign_time[9][9] = 50;            // 9000-9999rpm ingition load site
ign_time[9][8] = 50;
ign_time[9][7] = 50;
ign_time[9][6] = 50;
ign_time[9][5] = 50;
ign_time[9][4] = 50;
ign_time[9][3] = 50;
ign_time[9][2] = 50;
ign_time[9][1] = 50;
ign_time[9][0] = 50;

ign_time[8][9] = 50;            // 8000-8999rpm ingition load site
ign_time[8][8] = 50;
ign_time[8][7] = 50;
ign_time[8][6] = 50;
ign_time[8][5] = 50;
ign_time[8][4] = 50;
ign_time[8][3] = 50;
ign_time[8][2] = 50;
ign_time[8][1] = 50;
ign_time[8][0] = 50;

ign_time[7][9] = 45;            // 7000-7999rpm ingition load site
ign_time[7][8] = 45;
ign_time[7][7] = 45;
ign_time[7][6] = 45;
ign_time[7][5] = 45;
ign_time[7][4] = 45;
ign_time[7][3] = 45;
ign_time[7][2] = 45;
ign_time[7][1] = 45;
ign_time[7][0] = 45;

ign_time[6][9] = 45;            // 6000-6999rpm ingition load site
ign_time[6][8] = 45;
ign_time[6][7] = 45;
ign_time[6][6] = 45;
ign_time[6][5] = 45;
ign_time[6][4] = 45;
ign_time[6][3] = 45;
ign_time[6][2] = 45;
ign_time[6][1] = 45;
ign_time[6][0] = 45;
```

```
ign_time[5][9] = 45;            // 5000-5999rpm ingition load site
ign_time[5][8] = 45;
ign_time[5][7] = 45;
ign_time[5][6] = 45;
ign_time[5][5] = 45;
ign_time[5][4] = 45;
ign_time[5][3] = 45;
ign_time[5][2] = 45;
ign_time[5][1] = 45;
ign_time[5][0] = 45;

ign_time[4][9] = 45;            // 4000-4999rpm ingition load site
ign_time[4][8] = 45;
ign_time[4][7] = 45;
ign_time[4][6] = 45;
ign_time[4][5] = 45;
ign_time[4][4] = 40;
ign_time[4][3] = 40;
ign_time[4][2] = 40;
ign_time[4][1] = 40;
ign_time[4][0] = 40;

ign_time[3][9] = 47;            // 3000-3999rpm ingition load site
ign_time[3][8] = 47;
ign_time[3][7] = 47;
ign_time[3][6] = 47;
ign_time[3][5] = 47;
ign_time[3][4] = 40;
ign_time[3][3] = 40;
ign_time[3][2] = 40;
ign_time[3][1] = 40;
ign_time[3][0] = 40;

ign_time[2][9] = 47;            // 2000-2999rpm ingition load site
ign_time[2][8] = 47;
ign_time[2][7] = 47;
ign_time[2][6] = 47;
ign_time[2][5] = 47;
ign_time[2][4] = 40;
ign_time[2][3] = 40;
ign_time[2][2] = 40;
ign_time[2][1] = 40;
ign_time[2][0] = 40;

ign_time[1][9] = 35;            // 1000-1999rpm ingition load site
ign_time[1][8] = 35;
ign_time[1][7] = 35;
ign_time[1][6] = 35;
ign_time[1][5] = 35;
ign_time[1][4] = 35;
ign_time[1][3] = 35;
ign_time[1][2] = 35;
ign_time[1][1] = 35;
ign_time[1][0] = 35;
```

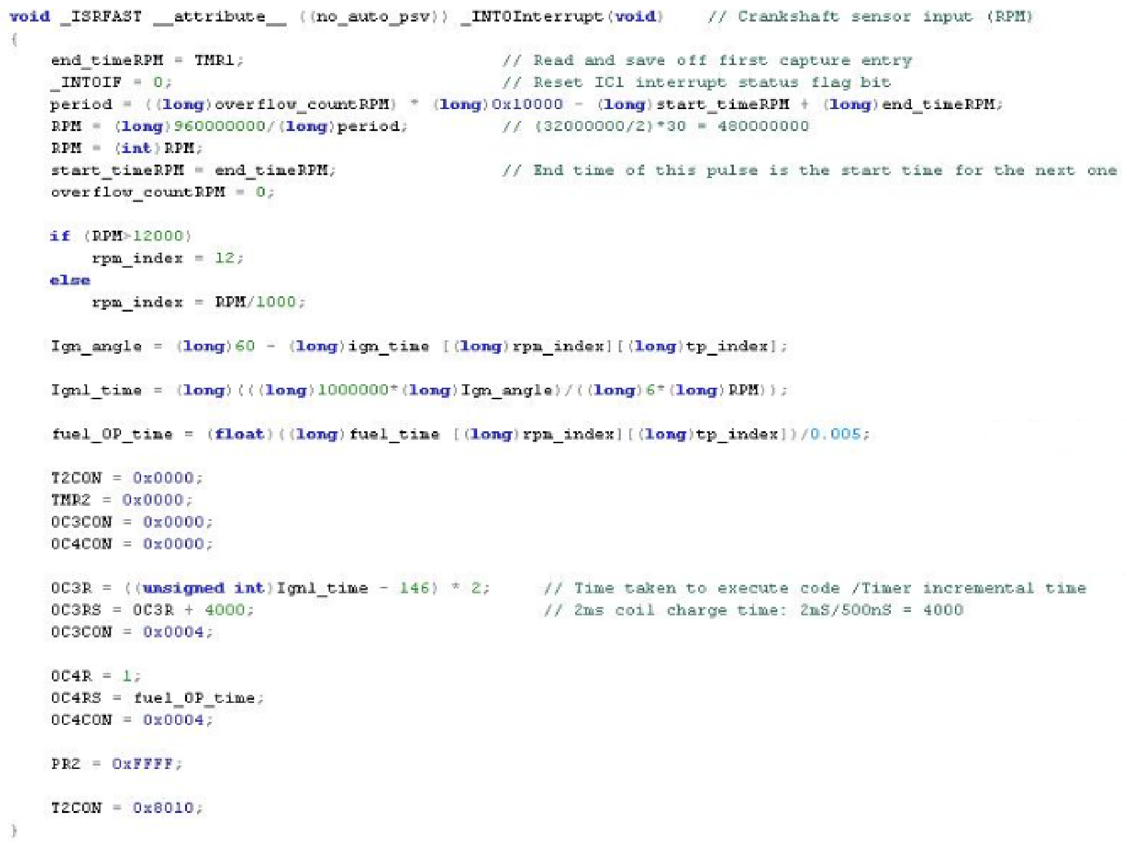
```
    ign_time[0][9] = 30;                    // 0 - 999rpm ingition load site
    ign_time[0][8] = 30;
    ign_time[0][7] = 30;
    ign_time[0][6] = 30;
    ign_time[0][5] = 30;
    ign_time[0][4] = 30;
    ign_time[0][3] = 30;
    ign_time[0][2] = 30;
    ign_time[0][1] = 30;
    ign_time[0][0] = 30;



 while(1)
 {
     i=readADC1(ADC_CH0);         //Read analog value on pin 1 on J5 of MuitiIO board

     if (i <=127)                          // TPS calibration for tp_index
         tp_index = 0;
     else if (i <=162)
         tp_index = 1;
     else if (i <=197)
         tp_index = 2;
     else if (i <=232)
         tp_index = 3;
     else if (i <=267)
         tp_index = 4;
     else if (i <=302)
         tp_index = 5;
     else if (i <=337)
         tp_index = 6;
     else if (i <=372)
         tp_index = 7;
     else if (i <=407)
         tp_index = 8;
     else if (i <=438)
         tp_index = 9;

    lcd_gotoxy(0,0);              // Printing RPM value to LCD
    lcd_outtext("RPM:");

    itoa(RPM,St);
    lcd_gotoxy(4,0);
    lcd_outtext(St);

    if (RPM<1000)
    {
        lcd_gotoxy(7,0);
        lcd_outtext("  ");
    }
    if (RPM<10000)
    {
        lcd_gotoxy(8,0);
        lcd_outtext("  ");
    }
```

```
a = (int)rpm_index;          // Printing RPM index to LCD
itoa(a,St);
lcd_gotoxy(12,0);
lcd_outtext(St);
if (rpm_index<10)
{
    lcd_gotoxy(13,0);
    lcd_outtext(" ");
}

lcd_gotoxy(0,1);             // Printing TPS value to LCD
lcd_outtext("TPS:");

itoa(i,St);
lcd_gotoxy(4,1);
lcd_outtext(St);
if (i <10)
{
    lcd_gotoxy(5,1);
    lcd_outtext(" ");
}
else if (i <100)
{
    lcd_gotoxy(6,1);
    lcd_outtext(" ");
}

a = (int)tp_index;           // Printing TP index to LCD
itoa(a,St);
lcd_gotoxy(12,1);
lcd_outtext(St);
if (tp_index <10)
{
    lcd_gotoxy(13,1);
    lcd_outtext(" ");
}

lcd_gotoxy(-4,2);
lcd_outtext("Fuel:");


a = (int)(fuel_time [(int)rpm_index][(int)tp_index]);   // Printing fueling value to LCD
itoa(a,St);
lcd_gotoxy(2,2);
lcd_outtext(St);
if (fuel_time [(int)rpm_index][(int)tp_index] <10)
{
    lcd_gotoxy(3,2);
    lcd_outtext(" ");
}

lcd_gotoxy(5,2);
lcd_outtext("ms");

lcd_gotoxy(-4,3);
lcd_outtext("Timing:");
```

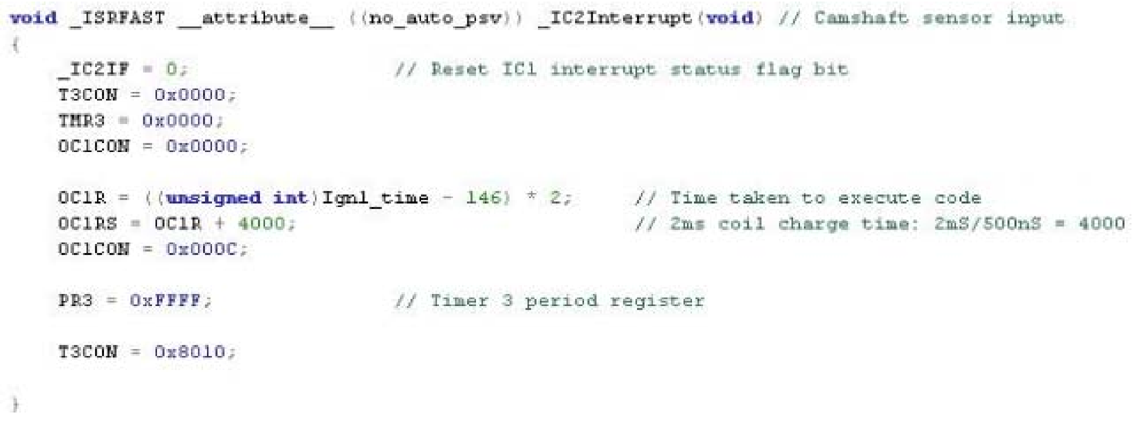
```
        a = ((int)ign_time [(int)rpm_index][(int)tp_index]);  // Printing ingition timing value to LCD
        itoa(a,St);
        lcd_gotoxy(4,3);
        lcd_outtext(St);
        if (ign_time [(int)rpm_index][(int)tp_index] <10)
        {
            lcd_gotoxy(5,3);
            lcd_outtext(" ");
        }

        lcd_gotoxy(7,3);
        lcd_outtext("Deg");
    }

    return(0);
}

void Init_PIC(void)                // Initialize the PIC
{
    _TRISA1 = 1;                   // RA1 as input
    _TRISB0 = 1;                   // RB0 as input - Analogs
    _TRISB1 = 1;                   // RB1 as input
    _TRISB2 = 1;                   // RB2 as input
    _TRISB3 = 1;                   // RB3 as input
    _TRISB4 = 1;                   // RB4 as input
    _TRISB5 = 1;                   // RB5 as input
    _TRISB6 = 1;                   // RB6 as input
    _TRISB7 = 1;                   // RB7 as input
    _TRISB8 = 1;                   // RB8 as input
    _TRISB9 = 1;                   // RB9 as input
    _TRISB10 = 1;                  // RB10 as input - TPS sensor
    _TRISB11 = 0;                  // RB11 as output
    _TRISD8 = 1;                   // RD8 (IC1) as input
    _TRISD9 = 1;                   // RD9 (IC2) as input - Camshaft sensor
    _TRISD0 = 0;                   // RD0 (OC1) as output - Ignition coil output 2
    _TRISD1 = 0;                   // RD1 (OC2) as output
    _TRISD2 = 0;                   // RD2 (OC3) as output - Ignition coil output 1
    _TRISD3 = 0;                   // RD3 (OC4) as output - Fuel injectors output
    _TRISF6 = 1;                   // RF6 as input - Crankshaft sensor
    _TRISG6 = 1;                   // RG6 as input
    _TRISG7 = 1;                   // RG7 as input
    _TRISG8 = 1;                   // RG8 as input
    _TRISG9 = 1;                   // RG9 as input

}

void Init_Ext_Int(void)
{
    _INT0IE = 1;                  // External Interrupt 0 Enable bit
                                  // 1 = Interrupt request enabled
                                  // 0 = Interrupt request not enabled

    _INT0EP = 0;                  // External Interrupt 0 Edge Detect Polarity Select bit
                                  // 1 = Interrupt on negative edge
                                  // 0 = Interrupt on positive edge

    _INT0IP = 1;                  // External Interrupt Priority level 1 (level 4 Default)

    _INT0IF = 0;                  // External Interrupt 0 Flag Status bit
                                  // 1 = Interrupt request has occurred
                                  // 0 = Interrupt request has not occurred
}
```

```
void Init_Timer1(void)
{
    _T1IP = 1;                    // Timer1 Interrupt Priority level 1 (level 4 Default)
    TMR1 = 0x00;                  // Clear Timer1
    T1CON = 0x8000;               // Configure Timer1 Control Register
                                  // Start 16-bit Timer1
                                  // Continue module operation in Idle mode
                                  // Gated time accumulation disabled
                                  // Prescale Select bits 1:1
                                  // Two 16-bit timers
                                  // Internal clock (FOSC/2)
                                  // Timer1 will increment every: (1/32000000)*2*1 = 62.5nS
                                  // Timer1 will overflow  every: 62.5nS*65536 = 4.096mS
    _T1IF = 0;                    // Reset Timer1 interrupt status flag bit
    _T1IE = 1;                    // Enable Timer1 Interrupt
}

void Init_Timer3(void)
{
    _T3IP = 4;                    // Timer3 Interrupt Priority level 1 (level 4 Default)
    TMR3 = 0;                     // Clear Timer3
    T3CON = 0x0010;               // Configure Timer3 Control Register
                                  // Start 16-bit Timer3
                                  // Continue module operation in Idle mode
                                  // Gated time accumulation disabled
                                  // Prescale Select bits 1:1
                                  // Two 16-bit timers
                                  // Internal clock (FOSC/2)
                                  // Timer3 will increment every: (1/32000000)*2*1 = 62.5nS
                                  // Timer3 will overflow  every: 62.5nS*65536 = 4.096mS
    _T3IF = 0;                    // Reset Timer3 interrupt status flag bit
    _T3IE = 0;                    // Enable Timer3 Interrupt
}

void Init_ADC(void)               // Initialize the ADC
{
    AD1PCFG = 0;                  // All input pins are analog
    AD1CON1  = 0x00E0;            // Auto convert after end of sampling
    AD1CSSL  = 0x0000;            // No scanning required
    AD1CON3  = 0x1F02;            // Max sample time = 31Tad, Tad = 2 x Tcy = 125ns >75ns
    AD1CON2  = 0x0000;            // Use MUXA, AVss and AVdd are used as Vref+/-
    AD1CON1bits.ADON = 1;         // Turn on the ADC
}

void Init_IP_Capture1(void)
{
    _IC2IP = 4;                   // Input Capture Channel 2 Interrupt Priority level 4 (Default)
    _IC2IF = 0;                   // Reset IC2 interrupt status flag bit
    _IC2IE = 1;                   // Enable Input Capture Channel 2 Interrupt
    IC2CON = 0x0000;              // Turn off Input Capture 2 Module
    IC2CON = 0x0083;              // Configure Input Capture Control 2 Register
                                  // Input capture will continue to operate in CPU Idle mode
                                  // TMR2 contents are captured on capture event
                                  // Interrupt on every capture event
                                  // Capture mode, every rising edge
    IC2BUF = 0;                   // Reset Input Capture 2 Buffer Register
}
```

```
void Init_OP_Compare1(void)
{
    _T2IP = 4;                  // Timer2 Interrupt Priority level 4 (Default)
    TMR2 = 0;                   // Clear Timer2

    OC1CON = 0x0000;            // Turn off Output Compare Modules 1,2,3
    OC3CON = 0x0000;
    OC4CON = 0x0000;
    OC1CON = 0x0004;            // Configure Output Compare Registers 1,2,3
                                // Output capture 1 will continue to operate in CPU Idle mode
                                // Timer3 is the clock source for output Compare 1
                                // Initialize OC1 pin low, generate single output pulse on OC1 pin
    OC3CON = 0x000C;
    OC4CON = 0x0004;
    OC1R = 0x0000;              // Reset Compare Registers 1,2,3
    OC1RS = 0x0000;             // Reset Secondary Compare Registers 1,2,3
    OC3R = 0x0000;
    OC3RS = 0x0000;
    OC4R = 0x0000;
    OC4RS = 0x0000;
    _OC1IE = 0;                 // Diable Output Compare interrupts 1,2,3
    _OC3IE = 0;
    _OC4IE = 0;

    T2CON = 0x0010;             // Configure Timer2 Control Register
                                // Start 32-bit Timer2
                                // Continue module operation in Idle mode
                                // Gated time accumulation disabled
                                // Prescale Select bits 1:256
                                // Internal clock (FOSC/2)
                                // Timer2 will increment every: (1/32000000)*2*1 = 62.5nS
                                // Timer2 will increment every: (1/32000000)*2*8 = 500nS
                                // Timer2 will overflow  every: 62.5nS*65536 = 4.096mS
                                // Timer2 will overflow  every: 500nS*65536 = 32.768mS
                                // Timer3 will overflow  every: 62.5nS*4294967296 = 268.235 sec
    _T2IF = 0;                  // Reset Timer3 interrupt status flag bit
    _T2IE = 0;                  // Disable Timer3 Interrupt
}


void Init_CN(void)
{
    _CN8IE = 1;                 // Enable change notification interrupt
    _CN9IE = 1;                 // Enable change notification interrupt
    _CN10IE = 1;                // Enable change notification interrupt
    _CN11IE = 1;                // Enable change notification interrupt

    _CN8PUE = 0;                // Disable internal pull-ups
    _CN9PUE = 0;                // Disable internal pull-ups
    _CN10PUE = 0;               // Disable internal pull-ups
    _CN11PUE = 0;               // Disable internal pull-ups

    _CNIP = 4;                  // Change notification priority 4
    _CNIF = 0;                  // Clear the change notification interrupt bit
    _CNIE = 1;                  // Change notification interrupt enable
}

void _ISRFAST __attribute__ ((no_auto_psv)) _T1Interrupt(void)  // RPM tick timer interrupt
{
    _T1IF = 0;                  // Reset Timer2 interrupt status flag
    ++overflow_countRPM;        // Increment whenever an overflow occurs
}
```



```c
void _ISRFAST __attribute__ ((no_auto_psv)) _INT0Interrupt(void)    // Crankshaft sensor input (RPM)
{
    end_timeRPM = TMR1;                         // Read and save off first capture entry
    _INT0IF = 0;                                // Reset IC1 interrupt status flag bit
    period = ((long)overflow_countRPM) * (long)0x10000 - (long)start_timeRPM + (long)end_timeRPM;
    RPM = (long)960000000/(long)period;         // (32000000/2)*30 = 480000000
    RPM = (int)RPM;
    start_timeRPM = end_timeRPM;                // End time of this pulse is the start time for the next one
    overflow_countRPM = 0;

    if (RPM>12000)
        rpm_index = 12;
    else
        rpm_index = RPM/1000;

    Ign_angle = (long)60 - (long)ign_time [(long)rpm_index][(long)tp_index];

    Ign1_time = (long)(((long)1000000*(long)Ign_angle)/((long)6*(long)RPM));

    fuel_OP_time = (float)((long)fuel_time [(long)rpm_index][(long)tp_index])/0.005;

    T2CON = 0x0000;
    TMR2 = 0x0000;
    OC3CON = 0x0000;
    OC4CON = 0x0000;

    OC3R = ((unsigned int)Ign1_time - 146) * 2;     // Time taken to execute code /Timer incremental time
    OC3RS = OC3R + 4000;                            // 2ms coil charge time: 2mS/500nS = 4000
    OC3CON = 0x0004;

    OC4R = 1;
    OC4RS = fuel_OP_time;
    OC4CON = 0x0004;

    PR2 = 0xFFFF;

    T2CON = 0x8010;
}
```



```c
void _ISRFAST __attribute__ ((no_auto_psv)) _IC2Interrupt(void) // Camshaft sensor input
{
    _IC2IF = 0;                 // Reset IC1 interrupt status flag bit
    T3CON = 0x0000;
    TMR3 = 0x0000;
    OC1CON = 0x0000;

    OC1R = ((unsigned int)Ign1_time - 146) * 2;     // Time taken to execute code
    OC1RS = OC1R + 4000;                            // 2ms coil charge time: 2mS/500nS = 4000
    OC1CON = 0x000C;

    PR3 = 0xFFFF;               // Timer 3 period register

    T3CON = 0x8010;

}
```

```
void _ISRFAST __attribute__ ((no_auto_psv)) _CNInterrupt(void)
{
    _CNIF = 0;                  // clear the change notification interrupt bit

    if (PORTGbits.RG6==1)
    {
        fuel_time [rpm_index][tp_index] = fuel_time [rpm_index][tp_index] + 1;
    }

    if (PORTGbits.RG7==1)
    {
        fuel_time [rpm_index][tp_index] = fuel_time [rpm_index][tp_index] - 1;
    }

    if (PORTGbits.RG8==1)
    {
        ign_time [rpm_index][tp_index] = ign_time [rpm_index][tp_index] + 1;
    }

    if (PORTGbits.RG9==1)
    {
        ign_time [rpm_index][tp_index] = ign_time [rpm_index][tp_index] - 1;
    }

}
```

Appendix E – Ignition Timing Register analysis

TIMING	10	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
RPM																											
1000	1.667	2.500	2.667	2.833	3.000	3.167	3.333	3.500	3.667	3.833	4.000	4.167	4.333	4.500	4.667	4.833	5.000	5.167	5.333	5.500	5.667	5.833	6.000	6.167	6.333	6.500	6.667
1250	1.333	2.000	2.133	2.267	2.400	2.533	2.667	2.800	2.933	3.067	3.200	3.333	3.467	3.600	3.733	3.867	4.000	4.133	4.267	4.400	4.533	4.667	4.800	4.933	5.067	5.200	5.333
1500	1.111	1.667	1.778	1.889	2.000	2.111	2.222	2.333	2.444	2.556	2.667	2.778	2.889	3.000	3.111	3.222	3.333	3.444	3.556	3.667	3.778	3.889	4.000	4.111	4.222	4.333	4.444
2000	0.833	1.250	1.333	1.417	1.500	1.583	1.667	1.750	1.833	1.917	2.000	2.083	2.167	2.250	2.333	2.417	2.500	2.583	2.667	2.750	2.833	2.917	3.000	3.083	3.167	3.250	3.333
2500	0.667	1.000	1.067	1.133	1.200	1.267	1.333	1.400	1.467	1.533	1.600	1.667	1.733	1.800	1.867	1.933	2.000	2.067	2.133	2.200	2.267	2.333	2.400	2.467	2.533	2.600	2.667
2750	0.606	0.909	0.970	1.030	1.091	1.152	1.212	1.273	1.333	1.394	1.455	1.515	1.576	1.636	1.697	1.758	1.818	1.879	1.939	2.000	2.061	2.121	2.182	2.242	2.303	2.364	2.424
3000	0.556	0.833	0.889	0.944	1.000	1.056	1.111	1.167	1.222	1.278	1.333	1.389	1.444	1.500	1.556	1.611	1.667	1.722	1.778	1.833	1.889	1.944	2.000	2.056	2.111	2.167	2.222
3250	0.513	0.769	0.821	0.872	0.923	0.974	1.026	1.077	1.128	1.179	1.231	1.282	1.333	1.385	1.436	1.487	1.538	1.590	1.641	1.692	1.744	1.795	1.846	1.897	1.949	2.000	2.051
3500	0.476	0.714	0.762	0.810	0.857	0.905	0.952	1.000	1.048	1.095	1.143	1.190	1.238	1.286	1.333	1.381	1.429	1.476	1.524	1.571	1.619	1.667	1.714	1.762	1.810	1.857	1.905
3750	0.444	0.667	0.711	0.756	0.800	0.844	0.889	0.933	0.978	1.022	1.067	1.111	1.156	1.200	1.244	1.289	1.333	1.378	1.422	1.467	1.511	1.556	1.600	1.644	1.689	1.733	1.778
4000	0.417	0.625	0.667	0.708	0.750	0.792	0.833	0.875	0.917	0.958	1.000	1.042	1.083	1.125	1.167	1.208	1.250	1.292	1.333	1.375	1.417	1.458	1.500	1.542	1.583	1.625	1.667
4250	0.392	0.588	0.627	0.667	0.706	0.745	0.784	0.824	0.863	0.902	0.941	0.980	1.020	1.059	1.098	1.137	1.176	1.216	1.255	1.294	1.333	1.373	1.412	1.451	1.490	1.529	1.569
4500	0.370	0.556	0.593	0.630	0.667	0.704	0.741	0.778	0.815	0.852	0.889	0.926	0.963	1.000	1.037	1.074	1.111	1.148	1.185	1.222	1.259	1.296	1.333	1.370	1.407	1.444	1.481
4750	0.351	0.526	0.561	0.596	0.632	0.667	0.702	0.737	0.772	0.807	0.842	0.877	0.912	0.947	0.982	1.018	1.053	1.088	1.123	1.158	1.193	1.228	1.263	1.298	1.333	1.368	1.404
5000	0.333	0.500	0.533	0.567	0.600	0.633	0.667	0.700	0.733	0.767	0.800	0.833	0.867	0.900	0.933	0.967	1.000	1.033	1.067	1.100	1.133	1.167	1.200	1.233	1.267	1.300	1.333
5250	0.317	0.476	0.508	0.540	0.571	0.603	0.635	0.667	0.698	0.730	0.762	0.794	0.825	0.857	0.889	0.921	0.952	0.984	1.016	1.048	1.079	1.111	1.143	1.175	1.206	1.238	1.270
5500	0.303	0.455	0.485	0.515	0.545	0.576	0.606	0.636	0.667	0.697	0.727	0.758	0.788	0.818	0.848	0.879	0.909	0.939	0.970	1.000	1.030	1.061	1.091	1.121	1.152	1.182	1.212
5750	0.290	0.435	0.464	0.493	0.522	0.551	0.580	0.609	0.638	0.667	0.696	0.725	0.754	0.783	0.812	0.841	0.870	0.899	0.928	0.957	0.986	1.014	1.043	1.072	1.101	1.130	1.159
6000	0.278	0.417	0.444	0.472	0.500	0.528	0.556	0.583	0.611	0.639	0.667	0.694	0.722	0.750	0.778	0.806	0.833	0.861	0.889	0.917	0.944	0.972	1.000	1.028	1.056	1.083	1.111
6250	0.267	0.400	0.427	0.453	0.480	0.507	0.533	0.560	0.587	0.613	0.640	0.667	0.693	0.720	0.747	0.773	0.800	0.827	0.853	0.880	0.907	0.933	0.960	0.987	1.013	1.040	1.067
6500	0.256	0.385	0.410	0.436	0.462	0.487	0.513	0.538	0.564	0.590	0.615	0.641	0.667	0.692	0.718	0.744	0.769	0.795	0.821	0.846	0.872	0.897	0.923	0.949	0.974	1.000	1.026
6750	0.247	0.370	0.395	0.420	0.444	0.469	0.494	0.519	0.543	0.568	0.593	0.617	0.642	0.667	0.691	0.716	0.741	0.765	0.790	0.815	0.840	0.864	0.889	0.914	0.938	0.963	0.988
7000	0.238	0.357	0.381	0.405	0.429	0.452	0.476	0.500	0.524	0.548	0.571	0.595	0.619	0.643	0.667	0.690	0.714	0.738	0.762	0.786	0.810	0.833	0.857	0.881	0.905	0.929	0.952
7250	0.230	0.345	0.368	0.391	0.414	0.437	0.460	0.483	0.506	0.529	0.552	0.575	0.598	0.621	0.644	0.667	0.690	0.713	0.736	0.759	0.782	0.805	0.828	0.851	0.874	0.897	0.920
7500	0.222	0.333	0.356	0.378	0.400	0.422	0.444	0.467	0.489	0.511	0.533	0.556	0.578	0.600	0.622	0.644	0.667	0.689	0.711	0.733	0.756	0.778	0.800	0.822	0.844	0.867	0.889
7750	0.215	0.323	0.344	0.366	0.387	0.409	0.430	0.452	0.473	0.495	0.516	0.538	0.559	0.581	0.602	0.624	0.645	0.667	0.688	0.710	0.731	0.753	0.774	0.796	0.817	0.839	0.860
8000	0.208	0.313	0.333	0.354	0.375	0.396	0.417	0.438	0.458	0.479	0.500	0.521	0.542	0.563	0.583	0.604	0.625	0.646	0.667	0.688	0.708	0.729	0.750	0.771	0.792	0.813	0.833
8250	0.202	0.303	0.323	0.343	0.364	0.384	0.404	0.424	0.444	0.465	0.485	0.505	0.525	0.545	0.566	0.586	0.606	0.626	0.646	0.667	0.687	0.707	0.727	0.747	0.768	0.788	0.808
8500	0.196	0.294	0.314	0.333	0.353	0.373	0.392	0.412	0.431	0.451	0.471	0.490	0.510	0.529	0.549	0.569	0.588	0.608	0.627	0.647	0.667	0.686	0.706	0.725	0.745	0.765	0.784
8750	0.190	0.286	0.305	0.324	0.343	0.362	0.381	0.400	0.419	0.438	0.457	0.476	0.495	0.514	0.533	0.552	0.571	0.590	0.610	0.629	0.648	0.667	0.686	0.705	0.724	0.743	0.762
9000	0.185	0.278	0.296	0.315	0.333	0.352	0.370	0.389	0.407	0.426	0.444	0.463	0.481	0.500	0.519	0.537	0.556	0.574	0.593	0.611	0.630	0.648	0.667	0.685	0.704	0.722	0.741
9250	0.180	0.270	0.288	0.306	0.324	0.342	0.360	0.378	0.396	0.414	0.432	0.450	0.468	0.486	0.505	0.523	0.541	0.559	0.577	0.595	0.613	0.631	0.649	0.667	0.685	0.703	0.721
9500	0.175	0.263	0.281	0.298	0.316	0.333	0.351	0.368	0.386	0.404	0.421	0.439	0.456	0.474	0.491	0.509	0.526	0.544	0.561	0.579	0.596	0.614	0.632	0.649	0.667	0.684	0.702
9750	0.171	0.256	0.274	0.291	0.308	0.325	0.342	0.359	0.376	0.393	0.410	0.427	0.444	0.462	0.479	0.496	0.513	0.530	0.547	0.564	0.581	0.598	0.615	0.632	0.650	0.667	0.684
10000	0.167	0.250	0.267	0.283	0.300	0.317	0.333	0.350	0.367	0.383	0.400	0.417	0.433	0.450	0.467	0.483	0.500	0.517	0.533	0.550	0.567	0.583	0.600	0.617	0.633	0.650	0.667
10250	0.163	0.244	0.260	0.276	0.293	0.309	0.325	0.341	0.358	0.374	0.390	0.407	0.423	0.439	0.455	0.472	0.488	0.504	0.520	0.537	0.553	0.569	0.585	0.602	0.618	0.634	0.650
10500	0.159	0.238	0.254	0.270	0.286	0.302	0.317	0.333	0.349	0.365	0.381	0.397	0.413	0.429	0.444	0.460	0.476	0.492	0.508	0.524	0.540	0.556	0.571	0.587	0.603	0.619	0.635
10750	0.155	0.233	0.248	0.264	0.279	0.295	0.310	0.326	0.341	0.357	0.372	0.388	0.403	0.419	0.434	0.450	0.465	0.481	0.496	0.512	0.527	0.543	0.558	0.574	0.589	0.605	0.620
11000	0.152	0.227	0.242	0.258	0.273	0.288	0.303	0.318	0.333	0.348	0.364	0.379	0.394	0.409	0.424	0.439	0.455	0.470	0.485	0.500	0.515	0.530	0.545	0.561	0.576	0.591	0.606
11250	0.148	0.222	0.237	0.252	0.267	0.281	0.296	0.311	0.326	0.341	0.356	0.370	0.385	0.400	0.415	0.430	0.444	0.459	0.474	0.489	0.504	0.519	0.533	0.548	0.563	0.578	0.593
11500	0.145	0.217	0.232	0.246	0.261	0.275	0.290	0.304	0.319	0.333	0.348	0.362	0.377	0.391	0.406	0.420	0.435	0.449	0.464	0.478	0.493	0.507	0.522	0.536	0.551	0.565	0.580
11750	0.142	0.213	0.227	0.241	0.255	0.270	0.284	0.298	0.312	0.326	0.340	0.355	0.369	0.383	0.397	0.411	0.426	0.440	0.454	0.468	0.482	0.496	0.511	0.525	0.539	0.553	0.567
12000	0.139	0.208	0.222	0.236	0.250	0.264	0.278	0.292	0.306	0.319	0.333	0.347	0.361	0.375	0.389	0.403	0.417	0.431	0.444	0.458	0.472	0.486	0.500	0.514	0.528	0.542	0.556
12250	0.136	0.204	0.218	0.231	0.245	0.259	0.272	0.286	0.299	0.313	0.327	0.340	0.354	0.367	0.381	0.395	0.408	0.422	0.435	0.449	0.463	0.476	0.490	0.503	0.517	0.531	0.544
12500	0.133	0.200	0.213	0.227	0.240	0.253	0.267	0.280	0.293	0.307	0.320	0.333	0.347	0.360	0.373	0.387	0.400	0.413	0.427	0.440	0.453	0.467	0.480	0.493	0.507	0.520	0.533