

Abstract

This project aims to design and implement a driver information and feedback system for the Student Formula Car Project, which fourth year engineering students at the University of Exeter are taking part of every year. Several important electronic components have been designed and built for this project, including a tachometer measuring engine speed, a speedometer measuring the speed at which the car is travelling at any one time, and a gear selection indicator which notifies the driver about when to change gear. The gauges are all digital, meaning they are either displayed on an LCD screen, or displayed using LED lights. All of those components can be implemented in the Student Formula cars of the future, as they are all feature important gauges and numbers which can help inform the driver about the state of the car. This project also aims to reduce the costs of manufacturing such components, as it can be seen later in this report, several driver information systems are already available in the market ready to purchase, however they are not within the budget of the Student Formula Project. This means that the cost of the components designed and built in this project has been taken into account, and all of the electronics have been chosen accordingly.

Keywords: Tachometer, Speedometer, Gearbox, Arduino, Electronics |

Table of contents

1. Introduction	1
1.1. Presentation of Problem	1
1.1. Aims	2
1.2. Objectives.....	2
2. Literature review.....	3
2.1. Formula 1 Display Module	3
2.2. Road Cars - Instrument Cluster	5
2.3. Display Modules using Arduino Uno	6
3. Methodology and theory.....	7
3.1. Measuring Engine Speed.....	7
3.2. Measuring Engine Speed with Arduino Uno	9
3.3. Measuring Car Speed using Arduino Uno	11
4. Design.....	12
4.1. Design of Tachometer Circuit	12
4.2. Design of Tachometer Circuit using the Arduino Uno Board	13
4.3. Design of Speedometer Circuit Using the Arduino Uno Board.....	15
5. Experimental Results.....	16
5.1. Tachometer Circuit Source Code	16
5.2. Results of Tachometer Circuit.....	18
5.3. Speedometer Circuit Source Code	19
6. Discussion and Conclusions	21
7. Project Management, Sustainability and Health and Safety.....	22
7.1. Project Management.....	22
7.2. Sustainability.....	23

7.3. Health and Safety / Risk Assessment.....	23
8. APPENDIX A - Arduino Uno Source Code for Tachometer Circuit.....	27
9. APPENDIX B - Arduino Uno Source Code for Speedometer Circuit.....	29

|

1. Introduction

1.1. *Presentation of Problem*

The aim of this project is to design and build a driver display module for the Student Formula Car project at the University of Exeter. This is a competition hosted by the Institute of Mechanical Engineers in which engineering students from universities from all around Europe produce a Formula racing car that they would later take to the Silverstone race track to compete with other universities. The university with a car that best fits the rules of the competition is declared the winner. [1]

Engineering students at the University of Exeter have been involved in the Student Formula Car Project since 2012. The cars that were made by the students had basic driver information display modules, which consisted of LED indicator lights showing basic functions such as ignition status. This raised the issue that the information displayed for the driver was not up to standards. For example, the driver did not know at what speed the engine was running at any particular time, or what speed the car was doing, until telemetry information was viewed after testing.

Information from the car can be displayed to the driver using a programmable display module that can be fitted inside the cabin, either on the dashboard behind the steering wheel, or on the steering wheel itself. There are companies that produce ready-made programmable display modules which can be purchased online. However, the cost of the modules is not within the budget of the project as they tend to be very expensive and complex.

This report will go through each of the steps that were followed in order to achieve the objectives of this project, including initial designs and ideas towards the beginning of the report, as well as the final design and conclusions at the end of the report. The electronic circuits of each components will be discussed as well as explained thoroughly.

Although the components designed and built for this project have not been used for the Student Formula Car project of 2015, they have been designed to be used with the 2015 car, meaning that engine data, information and characteristics are all assumed to be used for the 2015 car.

1.1. *Aims*

Programmable display modules are well known in the automotive industry, as most modern road and racing cars tend to house such components in the dashboard or on the steering wheel. However, implementing a display module within the Student Formula Project while remaining within the budget is quite a task, as purchasing them is very costly. Therefore, the aims of this project are the following:

- Design and implementation of a digital tachometer measuring engine speed, a digital speedometer measuring the speed of the car, and a gear selection indicator which allows the driver to the optimum time to change gear.
- Introduce new ways of implementing driver display modules
- Reduce the cost of production as well as the cost of equipment
- Introduce new electronics into the Student Formula Car project, which may be useful in the future of the project
- Optimisation of the Arduino Uno board for use in Motorsport

1.2. *Objectives*

The objectives of this project are the following:

- A tachometer is to be designed and built using LEDs as indicators of engine speed, measured in Revolutions Per Minute (RPM)
- Compare and contrast the different versions of a tachometer circuit, including one which is built using simple circuitry and a breadboard implementing a frequency to voltage converter and bar/dot display drivers connected to LED lights. The other circuit is built using the Arduino Uno Board, which has outputs connected to LED lights as well as an LCD screen.
- Design a speedometer using the Arduino Uno Board, which displays the speed of the car by using a speed sensor to collect speed/voltage data, which is then programmed to be displayed using an LCD screen
- Design and build a gear selection indicator using LED lights

2. Literature review

The driver of a high speed racing car needs a lot of information and feedback from the car in order to perform well. The development of display modules can be seen with Formula 1 over the past 50 years. Formula 1 cars in the 1960s had very basic steering wheels with one function only (to turn the car).[2] Throughout the years, the steering wheel gained a lot of functionalities such as pit speed limiter, how much fuel the car uses and even allows the driver to rehydrate with a drink during the race. The display modules started off with analogue displays with very basic information (such as speedometer, tachometer and oil temperature).

In 2009, McLaren implemented the display module within their steering wheel to save weight. [3] This also meant that the driver could see all the information he needed within the steering wheel, and be able to alter different functions and procedures in the car. The LCD screen in most modern F1 cars can display over 100 pages of information to the driver. [4] This shows that the display module is very important in modern day racing, and not only Formula 1 racing but other competition such as DTM where the display module is also very important for the driver. [5] As the budget is limited for the Student Formula Car project, the display module that is going to be designed as part of this individual project would have to deliver the same useful information to the driver that the more advanced companies can produce, but at a much lower price.

2.1. *Formula 1 Display Module*

The modern day Formula 1 steering wheel is far from the wooden steering wheels that were used in early Formula 1 cars. It consists of a very complex array of buttons and switches which the driver can make use of in different ways. As well as providing the engineers and the crew in the pits information about the car, the electronics in modern Formula 1 cars allow the driver to change different settings on the go, such as the air to fuel mix ratio, the break balance, activation of the Drag Reduction System, and many more functions. [4]

As far as this project is concerned, the budget and the size of the Student Formula Car project is too small to consider implementing on the go settings which are mentioned above. Therefore the main area of concern regarding the Formula 1 steering wheel is the display of the engine speed, the gear selection indicator and the speedometer.

Figure 1 shows the steering wheel which was used in Lewis Hamilton's Mercedes AMG Formula 1 racing car in the year 2013. As it can be seen, the steering wheel is used for many different buttons which perform many of the different settings mentioned before. The average cost of a modern day Formula 1 steering wheel is \$50,000 [6]. Of course this is way off the budget of the Student Formula Car project, however there are some useful features that can be witnessed and learnt from with regards to the Formula 1 steering wheel.



Figure 1 - Lewis Hamilton's F1 steering wheel of 2013. Source: <https://pbs.twimg.com/media/BC07KFUCIAAk8qE.jpg:large>

In the example shown in figure 1, there are 15 LEDs which are placed at the top of the steering wheel, which indicate the engine speed to the driver. They are also an indication to when the driver should change gear. The way in which gear changes are displayed to the driver differ from one Formula 1 car to the other, as they are all made by different companies. This can be done by having the lights turn on linearly one after the other when engine speed increases, and then all lights start flashing when it is time to change gear, or when the rev limiter is reached. The gear selection indicator is shown by the BCD to 7-segment display shown in figure 1 just below the 15 LED lights. This indicates to the driver which gear is engaged at any one time, from either neutral (in which case an N is displayed), reverse (in which case an R is displayed) or the number of the gear.

The speed of the car is also displayed to the driver using the displays to the left or to the right of the gear indicator. This is especially important to the driver, as racing involves entering and exiting corners at correct speeds, therefore it helps the driver slow down or speed up to the correct speed when cornering. It is also important for pit stops, as Formula 1 rules on each track state that the driver should not exceed a certain speed limit when entering the pits, therefore it is important for the driver to know what speed the car is doing when in the pits in order to avoid a penalty. The speed of the car when entering the pits can be limited using a button on the steering wheel [7]. This is very important as it can limit the engine speed, and when the driver engages the correct gear, the speed of the car would not exceed the limit which is programmed.

This project will base a lot of the designs and implementations on the Formula 1 steering wheel, as this would be ideal for the Student Formula Car due to the electronics and the nature of the cars. As the Student Formula Project is a racing competition, the information displayed to the driver needs to be shown clearly, and since Formula 1 have been using roughly the same sort of displays for a long time, this is the method that will be followed in this project.

2.2. Road Cars - Instrument Cluster

Instrument Clusters in road cars all vary depending on the make and type of car. Figure 2 shows a typical instrument cluster fitted to an Audi Q7. This features a needle which is calibrated to display the correct engine speed, and another one displaying the speed of the car, either in KPH or MPH. The way it works is that it changes electrical signals from the engine into movement of the needles. The needle is placed in a square metal hub which a coil of wire wrapped around



Figure 2 - Audi Q7 instrument cluster.

Source:

http://images.thecarconnection.com/lrg/2009-audi-q7-quattro-4-door-3-6l-premium-instrument-cluster_100254846_1.jpg

it, which is then placed in between two magnets consisting of a north pole magnet and a south pole magnet. Current from the signal amplifier is passed through the coil of wire, and depending on the amount of current passing through the wire, the needle either moves to the right or to the left [8]. The gear selection is either not displayed at all due to insignificance or if the car has a manual gearbox, or it can be displayed simply by using one of the trip computer screens which are placed below the instruments, as shown in figure 2.

Other road cars use digital instrumentation to show the same information to the driver. Shown in figure 3 is the instrument cluster found in the Honda S2000. It can be seen that it differs significantly from the instruments shown in figure 2, as the display in the Honda is completely digital. The engine speed is displayed through an array of lights which turn on linearly as the engine speed increases, much like the Formula 1



Figure 3 - Honda S2000 instrument cluster.

Source:

http://www.s2ki.com/s2000/gallery/page__module__images__section__img_ctrl__img__771815__file__med

steering wheel discussed in section 2.1. The speed of the car is displayed using an LCD screen (seen in figure 3, displaying 0).

Even though the instrument cluster shown in figure 2 has been proven to be very useful, as most of the cars today have the same sort of technology fitted to them, due to the nature of this project, a digital display would be more suitable. Therefore, a lot can be learnt from the display shown in figure 3. The digital display in the Honda S2000 is clear and well designed, and as it is a sports car the instrument cluster has been inspired by race cars. The designs that are later discussed in this project were highly influenced by the Honda's instrument cluster. |

2.3. Display Modules using Arduino Uno

There are several sources on the web which show the Arduino Uno being used as an RPM gauge, or tachometer. Source [9] shows the Arduino Uno being used to show the RPM of a small electric motor. It is done by using an LED which emits Infrared light, and an Infrared phototransistor which detects the infrared light. The motor used has 2 propellers, meaning when the motor is spinning, 2 cuts of the infrared beam count as one revolution. The Arduino Uno is programmed so that it displays the number of revolutions per minute on an LCD screen.

This is quite an accurate way of measuring RPM of a small motor, but it is not sufficient enough to measure the RPM of a car engine. This project involves data coming straight from the ECU, meaning there is no need to detect the actual signal coming from the engine, as it can easily be obtained.

Another source [10] shows that the Arduino Uno has been used to measure the speed of a bicycle, which is then displayed on an LCD screen. The speed of the bicycle needs to be measured by using a magnetic switch connected to one of the bicycle's wheels, then a code is written for the Arduino Uno, including an input for the radius of the wheel. The circuit then calculates the speed in MPH and sends the information to the LCD screen to be displayed to the rider.

This is useful for the speedometer that can be designed for this project. However, instead of using a magnetic switch/reed switch, the signal for the speed of the car can be obtained by using a speed sensor. This can be bought and installed easily. It gives a signal to the circuit, which similarly to [10], can be inputted into the Arduino Uno to be programmed and then displayed on an LCD screen.

3. Methodology and theory

The main focus of this project is implementing the various components of the display module using the Arduino Uno board, which is pictured in Figure 4. The Arduino Uno is a microcontroller board with analogue inputs, as well as digital input and output pins, which can be powered by a USB cable connected to a computer. The connection between the Arduino Uno and the computer can also allow the module to be programmed to perform all kinds of different operations. However once it is programmed, it does not need to be connected to a computer to operate, meaning the board can be placed in the car to be powered by the battery on board. It operates at a relatively low voltage of 5V [11].

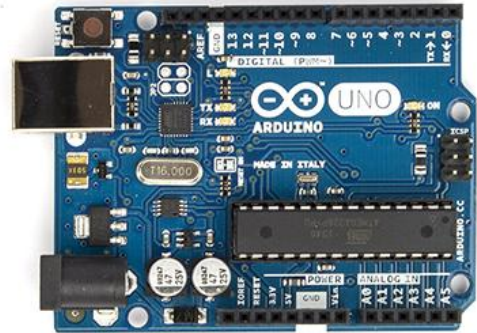


Figure 4 - Arduino Uno board. Source: <http://www.arduino.cc/en/main/arduino> BoardUno

However, there are also other methods of making the same components for the display module without using the Arduino Uno. These methods will be discussed, compared and contrasted with the Arduino Uno method.

3.1. *Measuring Engine Speed*

In order to design an instrument which correctly displays engine speed, information about the engine is required. This includes maximum RPM, and when the driver is expected to change gear. For this project, the instruments are going to be designed to be used for the Student Formula Car of 2015 at the University of Exeter. The information provided about the engine is the following:

- The engine has a maximum RPM of 12000
- The driver is expected to change gears between 8000 and 10000RPM

This suggests that the tachometer to be designed for this project must follow the stated information above. This means that it must indicate engine speed to the driver at all times, while giving signals to when the driver should change gear, as well as reaching the rev limiter at 12000RPM.

Engine speed in vehicles is typically is measured by a tachometer. As mentioned in section

1.2. the digital tachometer can be built using a simple circuit which uses a frequency to voltage converter and bar/dot LED display driver chips to show engine speed. Data from the Engine Control Unit (ECU), which is given as square wave pulses with a typical voltage of 5V, must be converted into an analogue signal which is then processed by the circuit to light up the LEDs in turn. The higher the analogue signal, the higher RPM and so therefore the more LEDs light up. The threshold for each LED to light up can be calibrated.

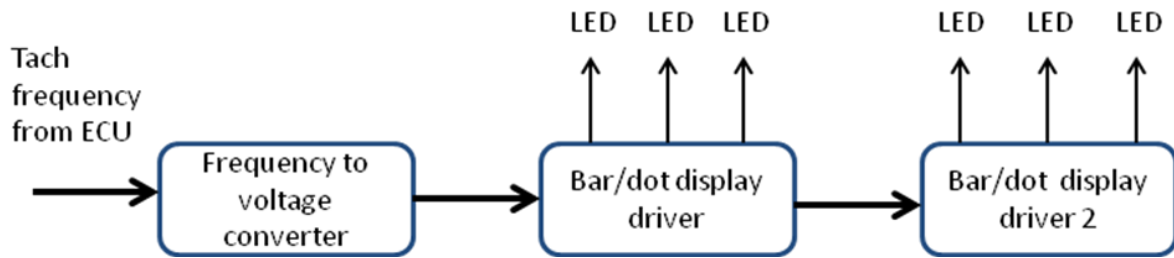


Figure 5 - Block diagram showing digital tachometer (breadboard circuit)

Figure 5 shows the block diagram for the tachometer circuit. The frequency from the ECU is to be converted to an analogue voltage signal using the frequency to voltage converter. The converter chosen for this circuit is the LM2917N chip, which can be used for many applications such as the tachometer to be designed in this project, speedometers and other electronics in vehicles. The datasheet can be found in [12].

Once the frequency from the ECU is converted to an analogue signal, it is transmitted to the bar/dot display driver, which in this case is the LM3914 chip. The LM3914 chip is an integrated circuit which can detect analogue voltage levels to drive 10 LED lights in a linear manner. This can be adjusted, or calibrated using the built in adjustable reference through resistors, and a 10-step voltage divider, meaning that in order for the system to indicate the engine speed with the LEDs, it has to be calibrated so that one LED represents a certain number of revolutions. The advantage of using LM3914 chips is that several chips can be connected in tandem, allowing more than just 10 LEDs to be used for the display. The guidelines used for the LM3914 can be found in the datasheet [13].

The block diagram in figure 5 can be used to design a tachometer circuit using the components stated above. As a control for this project, the circuit has been built and tested so that it can be compared to the circuits designed and discussed later on in this report.

3.2. *Measuring Engine Speed with Arduino Uno*

As mentioned in section 2.3. the Arduino Uno can be used as a tachometer in order to measure the RPM of a moving motor. For this project, the Arduino Uno is being used as an alternative method of implementing a tachometer. The method discussed in section 3.1. uses bar/dot display drivers in order to display the RPM. The Arduino Uno is used instead of the bar/dot display drivers as shown in figure 6 below:

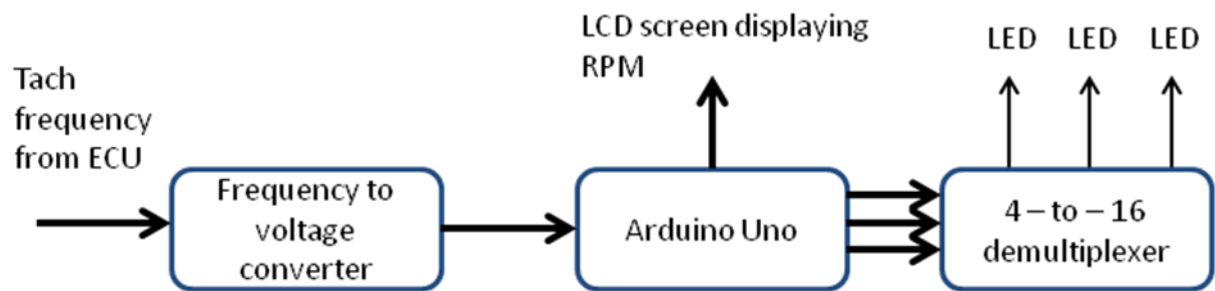


Figure 6 - Block diagram showing digital tachometer using the Arduino Uno

Figure 6 shows the block diagram for the digital tachometer circuit to be designed using the Arduino Uno board. The frequency to voltage converter is still used as the signal from the ECU must be converted to an analogue voltage, following the same principles used in section 3.1. However, it is used as an input into the Arduino Uno board. The outputs from the Arduino Uno are then connected to a 4-to-16 demultiplexer chip [14]. As the name suggests, this chip has 4 inputs and 16 outputs. The reason for using the 16 output demultiplexer is because the Arduino Uno only has 13 outputs, meaning that more LEDs can be placed within the circuit if the demultiplexer is used.

The Arduino Uno can be programmed using the the Arduino software which can be downloaded from the Arduino website [15]. This was used to write all the programming necessary for the circuits designed in this project. Mentioned in section 2.3. is a tachometer circuit using the Arduino Uno, measuring RPM using a propeller connected to a motor. However, for this circuit it is not necessary to measure the RPM, as the signal comes directly from the ECU, meaning that the output signal from the frequency to voltage converter is correlates linearly with the RPM.

Assuming that an analogue voltage is used as in input into the Arduino Uno board, it must be programmed so that it displays the engine speed using the LCD screen in RPM. The LEDs in this circuit are used as gear changing indicators, meaning they are designed so that they

indicate to the driver when to change gear, and also a rough indication of engine speed. In section 3.1. it was mentioned that the maximum RPM is 12000RPM, and the driver is expected to change gear between 8000 and 10000RPM. Therefore, the following designs were taken into consideration when building this circuit:

- The LCD displays the RPM at all times.
- 3 different colours of LEDs are used. Yellow LEDs indicate low RPM, green LEDs indicate mid range RPM and red LEDs indicate high RPM.
- Between 0 to 4000RPM, all the LEDs are off. This indicates to the driver that the engine is not revving at a high engine speed.
- Between 4000 to 6000RPM, the yellow LEDs turn on.
- Between 6000 to 8000RPM, the green LEDs turn on as well.
- Between 8000 to 10000RPM, the red LEDs turn on, indicating to the driver that this is the optimum time to change gear.
- Between 10000 to 12000RPM, all the lights begin to flash, which indicate to the driver that the engine is revving at its highest, reaching the rev limiter.

The Arduino Uno has analogue inputs which provide 10 bits of resolution, meaning that they have 1024 different values between 0 and 5V [11]. This means that in order to program the LEDs, the threshold for each RPM value must be calculated in volts, assuming that a 5V analog signal is being inputted into the Arduino Uno board:

$$(5V / 12000RPM) * 0 = 0RPM \text{ at } 0V$$

$$(5V / 12000RPM) * 4000 = 4000RPM \text{ at } 1.65V$$

$$(5V / 12000RPM) * 6000 = 6000RPM \text{ at } 2.5V$$

$$(5V / 12000RPM) * 8000 = 8000RPM \text{ at } 3.34V$$

$$(5V / 12000RPM) * 10000 = 10000RPM \text{ at } 4.17V$$

$$(5V / 12000RPM) * 12000 = 12000RPM \text{ at } 5V$$

The voltages stated above the thresholds for each of the RPM values shown, which will be used in the programming of the Arduino Uno. Displaying the RPM also requires the value of the RPM to be calculated. This is done using the following calculations:

The analogue reading of the Arduino Uno gives a value between 0 and 1024, corresponding to 0 to 5V respectively. Therefore to find the real voltage, the analogue reading must be divided by 1024, then multiplied by 5.

i.e. if the analogue reading of the Arduino Uno is given as 512, the real voltage is:

$(512/1024) * 5 = 2.5V$. This means that the reading corresponds to 2.5V, which is the analogue voltage being provided to the ArduinoUno. To find what RPM that corresponds to, the value is multiplied by 2400. This value is obtained by dividing the total RPM (12000) by the voltage (5V). This means that 2.5V corresponds to 6000RPM, as stated on page 10. The Arduino Uno can then be programmed to display the RPM.

3.3. *Measuring Car Speed using Arduino Uno*

The speedometer designed for this project uses the same principle used in the tachometer circuit mentioned in section 3.2. However instead of obtaining the information from the car's ECU, the data about the speed of the car must be obtained using a speed sensor. This is a device which can be attached so that it senses the rotation of one of the car's wheels, it then produces a pulse signal (usually square wave), which can used to interpret the speed of the car [16]. For this project, the speedometer circuit has been designed, but it has not been built.



Figure 7 - Block diagram showing digital speedometer using Arduino Uno

As the voltage from the speed sensor is similar to the ECU signal from the tachometer (i.e. it is a square wave pulse frequency with a constant voltage), the frequency to voltage converter can be used again to provide an analogue voltage for the Arduino Uno to process. The Arduino Uno can then be programmed and used to display the speed of the car on an LCD screen. Information was provided about the expected speed of the car in the Student Formula Project, which is around 80KpH.

4. Design

4.1. Design of Tachometer Circuit

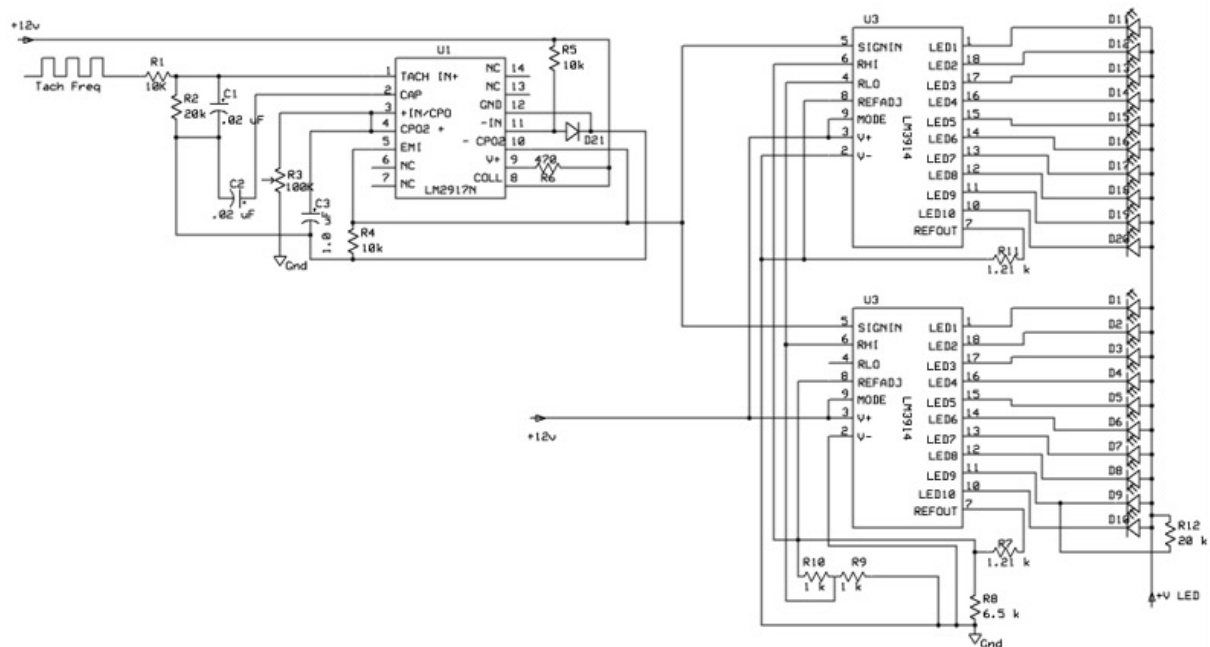


Figure 8- Circuit showing digital tachometer designed with frequency to voltage converter, and dot/bar display drivers. Source: http://www.rollmeover.com/bronco_fab/odds_n_ends/digi_tach_schematic_low.jpg

Figure 8 shows the circuit that can be designed using the frequency to voltage converter along with the bar/dot display drivers connected in tandem. The frequency signal of the engine speed is used as an input into the frequency to voltage converter. The value of the resistors and the capacitors follows the typical applications section in [12]. Once the frequency has been converted into an analogue voltage, the output from the frequency to voltage converter is connected to the input signal of the bar/dot display drivers. The value of the resistors again follows the typical applications section for the drivers, found in [13].

The circuit shown in figure 8 was built on a Breadboard, which is shown in figure 9.

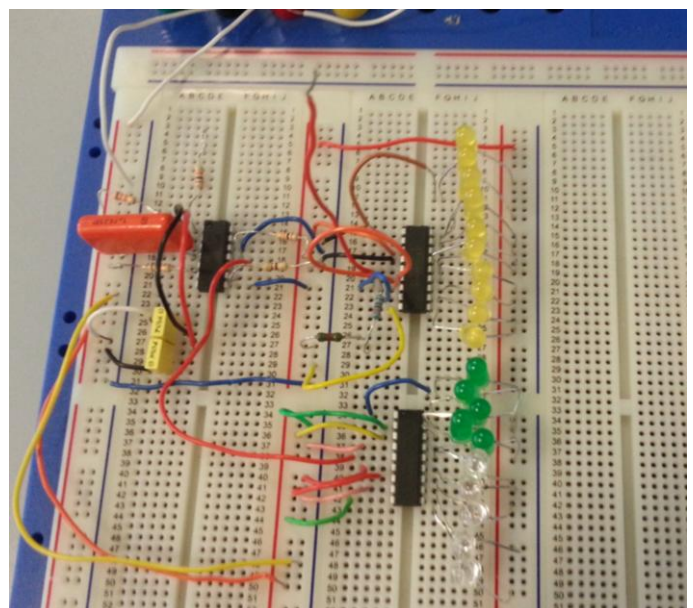


Figure 9 - Tachometer circuit built on a Breadboard.

The chip shown on the left hand side of the picture is the LM2917N frequency to voltage converter, while the two chips connected to the LEDs are the LM3917 chips, which are cascaded.

As there was no access to the Formula Car for this project, it was not possible to test the circuit on the actual car. Therefore all the tests had to be simulations in the lab. The input signal to the frequency to voltage converter was sourced by a signal generator, giving a square wave signal with an amplitude of 5V. The amplitude of the signal is not an important factor in these circumstances, as the frequency to voltage converter does not measure the amplitude of the square wave signal, but instead its frequency. Meaning the higher the frequency of the square wave produced by the signal generator, the higher the voltage at the output of the chip. The output of the frequency to voltage converter is then inputted into the two cascaded bar/dot display drivers, which are both connected to 10 LED lights each. As this is a tachometer circuit, the first 10 lights used were yellow indicating low RPM, the next 5 were green indicating to when the driver should ideally change gear, and the last 5 were red which indicate that the rev limiter is being reached.

This circuit has been built in order to be compared and contrasted with the Arduino Uno equivalent, which will determine which method is the more practical for use in the Student Formula Car.

4.2. *Design of Tachometer Circuit using the Arduino Uno Board*

By using the information provided in section 3.2. as well as the block diagram shown in figure 6, a circuit for the tachometer can be designed using the Arduino Uno.

Instead of using a signal generator to produce a square wave signal, the analogue voltage for the Arduino Uno will be provided by a power supply. This was done so it would be easier to programme the Arduino Uno without having to use a frequency to voltage converter as used in the tachometer circuit in section 4.1. The power supply is provided by the Agilent Dual Output DC power supply shown in figure 10, which can supply an analogue voltage between 0 and 5V.

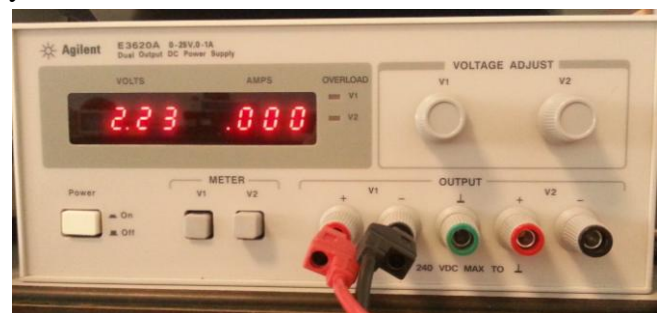


Figure 10 - Agilent power supply

The tachometer circuit that was designed using the Arduino Uno is shown below in figure 11:

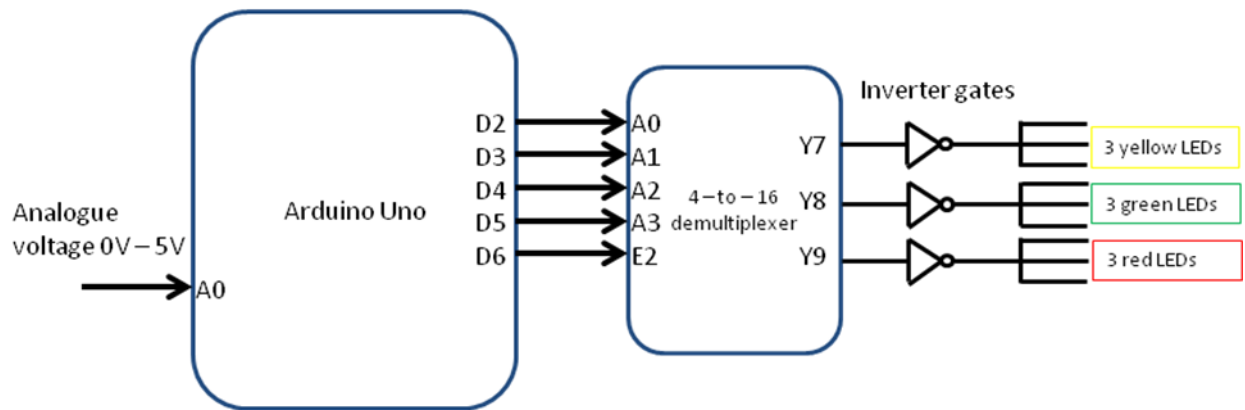


Figure 11: Tachometer circuit that was designed using the Arduino Uno, 4-to-16 demultiplexer, inverter gates and multicoloured LEDs.

The circuitry shown in figure 11 shows the scheme that was used to connect all the components to the Arduino Uno. In order to simulate a real reading, the power supply provides a voltage between 0 to 5V, which will be used to determine the engine speed of the car. Port A0 on the Arduino Uno is used as the input for the analogue input. The outputs that were used were the digital outputs from the Arduino Uno, from D2 to D6. These outputs determine the input into the 4-to-16 demultiplexer, which in turn will determine which LEDs are switched on and which ones are switched off.

INPUTS						OUTPUTS															
E1	E2	A3	A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = High Voltage Level, L = Low Voltage Level, X = Don't Care

Figure 12: Truth table for the 4-to-16 line demultiplexer. Source: [14]

Figure 12 shows the truth table for the 4-to-16 line demultiplexer. It can be seen that no matter what the state of the lines at the input are, the outputs are all high except for one. This presented an issue as for this circuit, as it is required that the opposite of this situation is performed, i.e. it is required for 1 output to be on while the rest are off. The simple solution to this problem was to use an inverter gate. This was done by using an inverter chip, given by the code CD74HC104E.

The output from the inverter gates is then inputted into 3 LEDs, which are connected in parallel with a 220Ω resistor connected to each one.

The circuit is then programmed to perform the tasks mentioned in section 3.2. i.e. display the RPM as well as indicate the gear indicator by using LED lights. The code written can be seen in section 8 (Appendix A) of this report, and will be explained in section 5.

4.3. Design of Speedometer Circuit Using the Arduino Uno Board

As mentioned in section 3.3. the data regarding the speed of the car can be obtained using a speed sensor, which produces a square wave much like the engine speed data obtained from the ECU. This means that the use of a frequency to voltage converter is necessary for this circuit as well. However, for ease of design, this project will assume that the frequency to voltage converter produces a voltage between 0 and 5V, therefore a power supply can be used as a substitute for simulation purposes.

This will serve as the input to the Arduino Uno, and the output would simply be to an LCD screen, as there is no need to display LEDs like shown in section 4.2. for the tachometer circuit. This circuit also does not require the use of any IC chips, as the connection is straight forward between the Arduino Uno and the LCD screen. This is shown in figure 13 below on page 16.

The power supply can provide a voltage to one of the analogue input pins on the Arduino Uno, which can then be programmed in order to display the speed at which the car is travelling at. The code for the speedometer circuit can be found in section 9 (Appendix B). The code will also be explained in section 5.3. |

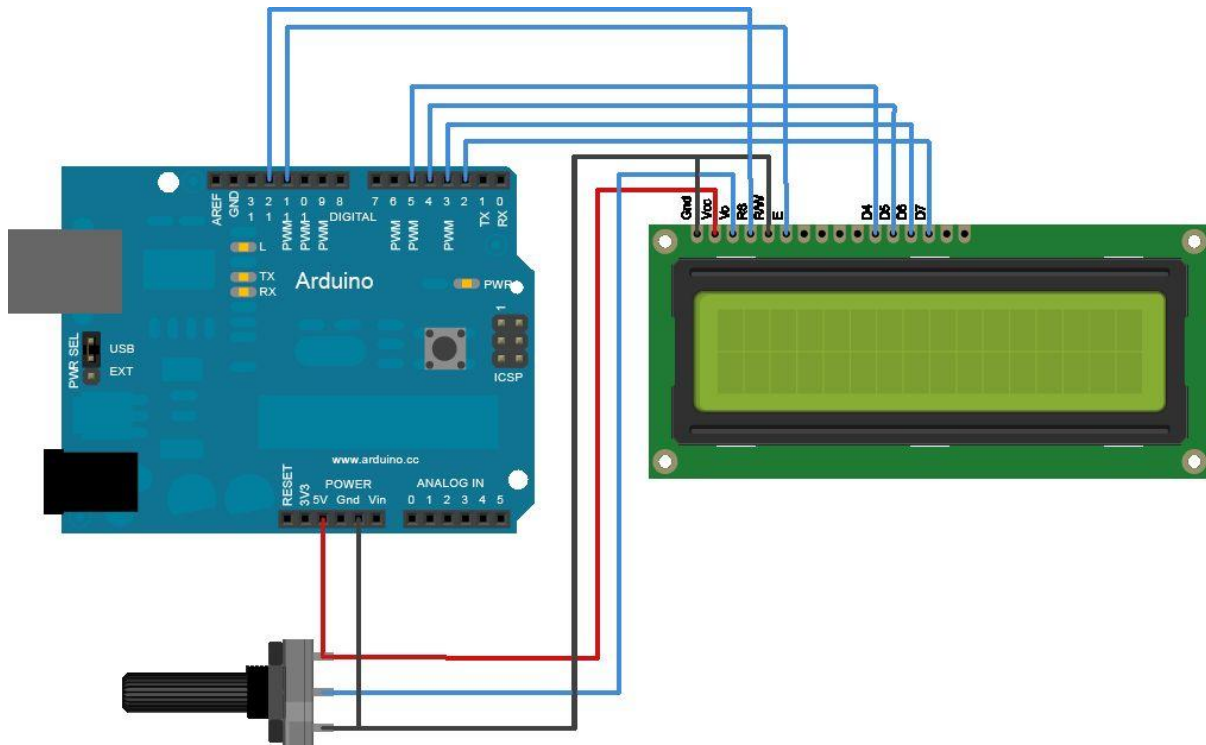


Figure 13: Arduino Uno board connections to an LCD screen. Source: <http://cdn.instructables.com/FFL/625T/H5IRUO7I/FFL625TH5IRUO7I.LARGE.jpg>

5. Experimental Results

5.1. Tachometer Circuit Source Code

The Arduino Uno must be programmed in order to perform certain tasks. In this project, a program was written for each of the components designed. The source code for the tachometer circuit can be found in Appendix A.

Lines 1 and 2 in the code identify the constants which are used. The constant integer 'voltagePin' is used to identify the analogue input pin on the Arduino Uno, meaning whenever 'voltagePin' is used in the code, it is referring to pin A0 on the Arduino Uno. The constant 'count' is used as a form of delay in the code. Line 4 declares a necessary function (setup) for the Arduino to be used, and line 5 allows a connection between the Arduino Uno and the computer to be established, running at 9600 bits per second. The (loop) function in line 8 is also necessary for the Arduino Uno to operate, as it states when the loop of the program starts [17].

As mentioned in section 3.2. the real voltage can be calculated by using a calculation, as the

value given by the Arduino Uno is between 0 and 1024. Therefore, the real voltage is calculated using the equation shown in line 10 of the code, using the 'float' function, as this allows the number to be displayed to be a floating number, since the voltage can be anywhere between 0 to 5V, so it is not an integer.

The RPM is then calculated using the equation shown on line 11, which is also mentioned in section 3.2. Lines 12 and 13 simply print the value of the RPM to the screen on the computer in real time.

Lines 15 and 16 identify the output pins on the Arduino Uno, which are pins 2 to 6 that are connected to the demultiplexer. Pin 6 outputs a signal to the E2 pin on the demultiplexer. As it can be seen in figure 12 on page 14, pins E1 and E2 must both be set to LOW in order for the demultiplexer to output a LOW signal at one of the outputs (which will be HIGH due to the inverter gates used). Therefore, lines 19 and 20 in the code output a HIGH signal to E2 when the voltage at the analogue input pin is less than 1.65V, which is the threshold for 4000RPM, as mentioned in section 3.2.

Lines 23 to 28 follow the scheme shown in figure 12 for the demultiplexer in order to turn on the yellow LED lights when the voltage is between 1.65V and 2.52V. This is when the engine speed is expected to be between 4000RPM and 6000RPM .

Lines 31 through to 43 turn the green LED lights, which is when the engine speed is between 6000RPM and 8000RPM.

Lines 47 to 65 turn on the red LED lights, where the engine is working between 8000RPM and 10000RPM. The red lights indicate to the driver that this is the optimum time to change gear.

Lines 69 through to 91 allow all the LED lights to start flashing. This is to indicate the maximum RPM of the engine is being reached, which is between 10000RPM and 12000RPM, or between 4.17V and 5V.

The source code was written as soon as all the necessary connections between the Arduino Uno and the computer were established. This was shown in figure 11.

5.2. Results of Tachometer Circuit

After the circuit was built complete, the Arduino Uno was connected to a computer using a USB cable, which provides the necessary voltage for it to operate as well as provide the connection between the compiler program and the board, so that the code that is written can be uploaded onto the board. The program was compiled and loaded onto the board, and the results are the following:

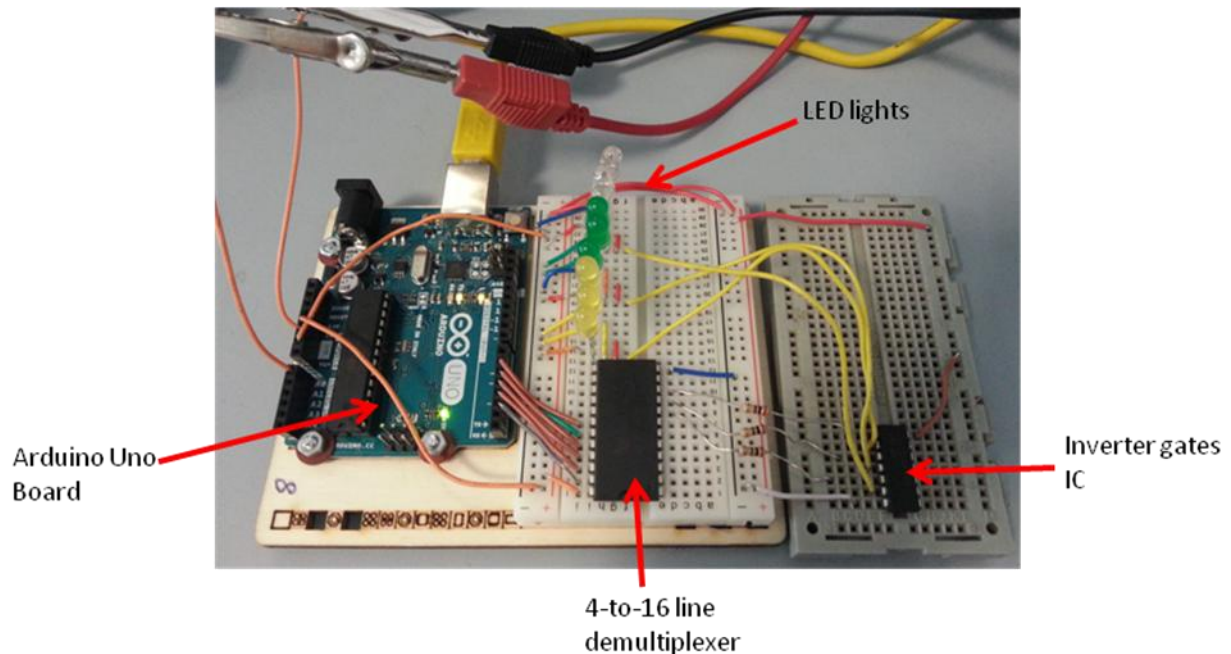


Figure 14: The final tachometer circuit built using the Arduino Uno, demultiplexer, inverter gates and LEDs

The circuit that was built is shown in figure 14. This circuit displays the RPM to the driver as well as show the driver when to change gear using the LED lights.

The RPM were displayed on the computer rather than an LCD screen due to the connections made on the Arduino Uno. The LCD screen could not be connected to the board as the digital pins were already used for the demultiplexer chip. Figure 13 shows the necessary connections which must be made between the Arduino Uno and the LCD. This includes digital pins 2, 3, 4 and 5 which were all used to power the LEDs through the demultiplexer. This can be changed so that the outputs for the demultiplexer do not include those pins, but instead use the other digital pins which are not used for the LCD, i.e. digital pins 6, 7, 8, 9 and 10. Connecting the demultiplexer using those pins allows the LCD to be connected, meaning the circuit would be able to show the engine speed on the LCD rather than on the PC screen.

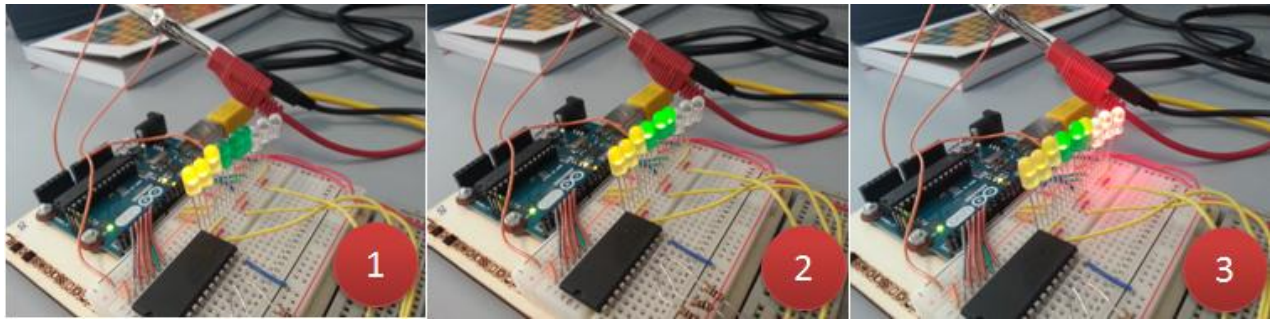


Figure 15: The tachometer when supplied with analogue voltage between 0 and 5V.

The three photos shown in figure 15 show the circuit working as the input voltage is changed. When the voltage is below the threshold of 4000RPM, all the lights remain off.

As soon as the voltage exceeds that voltage, the yellow LEDs turn on, as the RPM is between 4000RPM and 6000RPM. which is shown in picture 1.

The circuit allows the yellow lights to stay on, as well as turn on the green lights at the same time when the RPM is between 6000RPM and 8000RPM, shown in picture 2.

All the LEDs turn on when the RPM is between 8000RPM and 10000RPM, which is shown in picture 3.

When the voltage threshold exceeds the voltage for engine speeds above 10000RPM, all the lights start flashing simultaneously.

The RPM is also displayed in the serial monitor in real time, meaning that it changes immediately as soon as the input voltage to the board is changed.

Even though the program written for this application is looping, meaning the lights turn on and off constantly, the program is looping at a very high frequency, so the lights always seem to be on for the human eye.

5.3. *Speedometer Circuit Source Code*

The source code for the speedometer circuit can be found in Appendix B. Even though the circuit was designed and the code was written, the speedometer circuit was not actually built for this project, therefore it is purely theoretical. It was assumed that a separate Arduino Uno board was used for the speedometer circuit, however as it will be discussed in section 6 of

this report, it is possible for all the components to be built using one Arduino Uno board, which is economical as well as practical.

The source code shown in Appendix B is quite short as it does not require a lot of coding to display the speed on the LCD screen.

Line 1 shows the code importing the LiquidCrystal library, which is required for the LCD screen to be used with the Arduino Uno. Line 2 then identifies the pins that will be used to communicate between the Arduino Uno and the LCD.

Line 4 identifies the analogue input voltage pin, which in this case is identical to the tachometer circuit as a voltage supply is being used to power the circuit.

Lines 6 and 7 setup the board and the LCD screen by identifying the size of the LCD screen, i.e. 16 columns and 2 rows .

Lines 10 through to 22 is the loop that shows the speed of the car. As before, the voltage is assumed to be between 0 and 5V, therefore in order to obtain the real voltage, the voltage from the analogue voltage pin is divided by 1024 and then multiplied by 5. After obtaining the real voltage, the speed of the car can be calculated. Information about the usual speeds that the Student Formula Cars was provided for this project, and that is that they do not exceed 100Km/h. Therefore, the maximum speed will be assumed to be 100Km/h. Therefore in order to obtain the speed, the real voltage was multiplied by 20 (100/5). It is then displayed on the LCD screen as well as the serial monitor.

6. Discussion and Conclusions

This project has shown that it is possible to produce a digital display module for the Student Formula Car project using the Arduino Uno. Even though it had been used for similar purposes before, as mentioned in the literature review, it has been proven that it could be used in motorsport as well.

The whole purpose of this project was to design and build circuits for the Student Formula Car project rather than buying display modules from external companies, as this would reduce the costs of the project significantly as well as stay within the budget of the project. All of the components used in this project add up to a price that is significantly less than buying a readymade display module. It is also highly programmable, meaning that it does not only serve one purpose, and it can be changed any time.

Section 5 showed that two different circuits were designed using the Arduino Uno but only the tachometer circuit was built. Even though this was due to using pins incorrectly (as the LCD screen could not be used alongside the demultiplexer), this can be changed easily to accommodate both the tachometer circuit and the speedometer circuit using one Arduino Uno board. The RPM as well as the speed of the car can both be displayed on the same LCD screen, while at the same time the gear selection indicator connected through the demultiplexer can also be built on the same circuit. With a few tweaks to the source code and the connections made between the components, this is definitely possible and highly programmable.

A tachometer circuit was designed and built without using the Arduino Uno board, as a control for the project. This showed that without using the Arduino Uno board, the tachometer circuit could get very complex and not very accurate. The tachometer circuit that was built on the breadboard (shown in section 4.1. of this project) was not as easy to build, or as easy to program as the Arduino Uno circuit. However, making the circuit without using an Arduino Uno could be significantly cheaper.

It was assumed for this project that for both the tachometer and the speedometer circuit, that a voltage between 0 and 5V is being provided. This was only done for practical reasons, in order to simplify the design of the circuit. As the circuit could not be tested along with a real car, the tests had to be simulated and therefore the power supply was used instead.

The results of this project were experimental, as they were done in a lab at the university. However, for future research, the circuitry could be tested with a real car in order to make sure everything works properly. The frequency to voltage converters that were mentioned in the design section of this report were included in the design of the circuits, but not included in the testing as there was no access to the Student Formula Car.

Future research can also include designing and building the whole circuit on one Arduino Uno board. As mentioned earlier, it is possible to build both circuits designed in this project on the same Arduino Uno board.

7. Project Management, Sustainability and Health and Safety

7.1. *Project Management*

Part of writing this project, a Gantt chart was designed to be followed since the beginning of the academic year. The Gantt Chart is shown below:

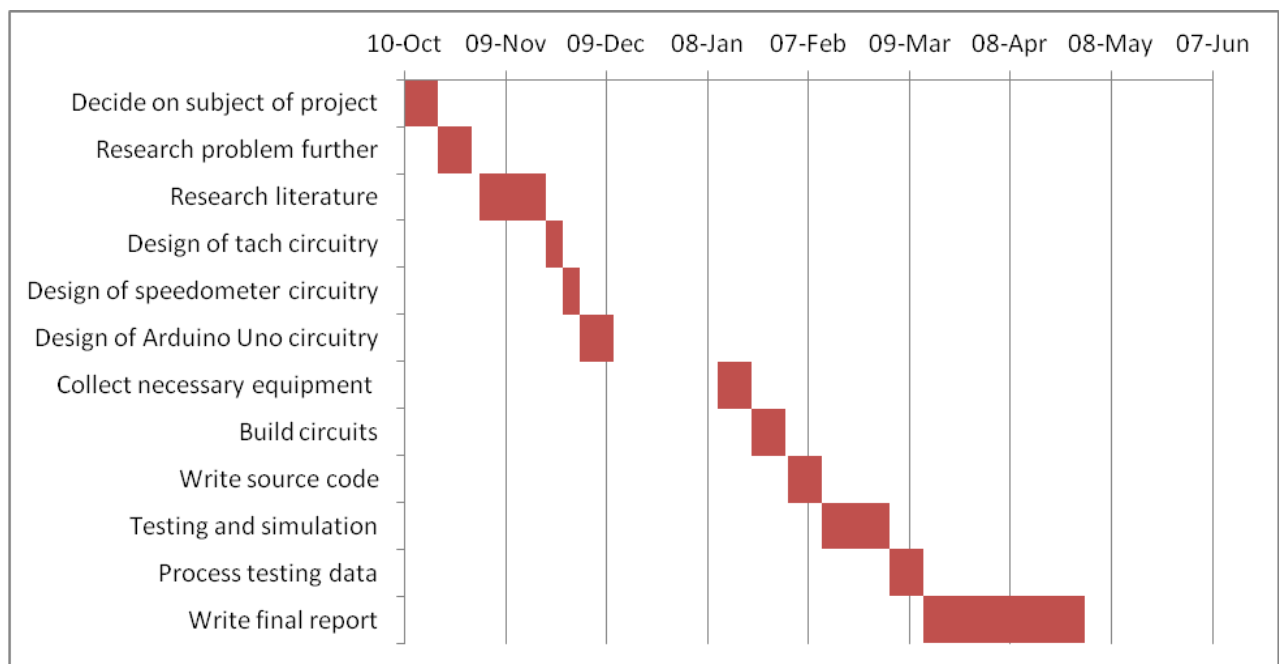


Figure 16: Gantt chart designed for the project

The Gantt chart in figure 16 shows all the steps that were taken in order to obtain all the results shown in this report. The problem was discovered at the beginning of the academic

year, which led to research of literature in order to find out what was done before regarding the subject. Once the research was complete, all the circuitry was designed by hand and then electronically so that they can be implemented later during the year. Once the circuits had been built and the source codes shown in the Appendices were written, the simulation results were noted down and the final step was to write the report.

7.2. *Sustainability*

The use of the Arduino Uno board for the circuitry of this project is very important as it is a relatively low cost device that can be programmed to do many different tasks. It is sustainable because it is highly programmable, and once the circuitry is built, it can be placed in the Student Formula Car to be used for the competition. However, as it is programmable, once the competition is finished, the board can be extracted from the car and can be reused for different tasks. All the integrated circuits are programmable as well, meaning there is no need to dispose of them after the project.

However, the integrated circuits are quite fragile, meaning if care is not taken when handling them, they can break easily. Therefore, this must be taken into consideration when building the circuitry. The Arduino Uno must also be enclosed in a case if it is to be placed inside the car, as debris or water might get into the car, causing it to malfunction or break completely.

7.3. *Health and Safety / Risk Assessment*

There are not too many health and safety issues regarding this project, as most of the simulation is done in the lab using low voltages and currents, therefore not too many accidents are prone to happen. However, the aim of this project is to implement the circuitry within the Student Formula Car, where there are more risks involved, which can include injury due to car battery wiring as the components designed in this project are designed to be powered by the in car battery. When working in the lab, lab procedures were always followed, i.e. at least two people must be in the lab at any one time, and care must be taken when operating all the necessary equipment such as the signal generators and the power supplies. A risk assessment table was made to identify risks which can occur during the project, and ways of minimising the risks. The table is shown on page 24.]

ID	Risk item	Effect	Cause	Likelihood	Severity	Importance	Action to minimise risk
1	Injury due to malfunctioning equipment in lab	Equipment in the lab can malfunction causing injury to user	Electrical malfunction/mechanical malfunction, human error	3	8	24	Always follow rules and regulations of lab, at least 2 people in the lab at any one time. Supervision at all times
2	Faulty integrated circuits	Circuitry would not work if ICs are faulty	Broken ICs, mishandled, or introduced to too much power	2	5	10	Always check the datasheets to avoid breaking ICs. Also, handle with care as they are fragile
3	Faulty Arduino Uno board	Project revolves around Arduino Uno, i.e. circuit would not work otherwise	Mishandling, faulty device, too much power	2	8	16	Handle board with care and check datasheet to avoid introducing too much power to the board
4	Injury due to moving car parts	Can cause injury to personnel	This could be due to procedures being done when the car is on and therefore has moving parts	2	10	20	Avoid working on the car if there are moving parts, and if necessary take precautions

Table 1: Risk assessment table showing risks and ways of avoiding them

References

- [1] IMechE. (2013). Formula Student - About us. Available: <http://events.imeche.org/formulastudent/about-us/thechallenge>. Last accessed 10th December 2014.
- [2] Bloom, D. (2014). How it was when Formula One was simply man and car - not like today: Inside Graham Hill's Lotus as it goes on sale for £1million. Available: <http://www.dailymail.co.uk/news/article-2671189/When-F1-just-man-car-Graham-Hills-1968-Lotus-sale-1m.html>. Last accessed 10th December 2014.
- [3] McLaren. (2014). Through the ages: Steering wheels. Available: <http://www.mclaren.com/formula1/team/through-the-ages-formula-one-steering-wheels/>. Last accessed 10th December 2014.
- [4] Golson, J. (2014). An Inside Look at the Insanely Complex Formula 1 Steering Wheel. Available: <http://www.wired.com/2014/05/formula-1-steering-wheels/>. Last accessed 10th December 2014.
- [5] DTM. (2014). DTM Technical Regulations. Available: <http://www.dtm.com/en/TechnicalRegulations.html>. Last accessed 10th December 2014
- [6] Arshad, S. (2014). *Average Cost of Formula 1 Car (Component Price Breakdown)*. Available: <http://www.tsmplug.com/f1/average-cost-of-formula-1-car/>. Last accessed 29th Apr 2015.
- [7] Prigg, M. (2014). *Think your car's dashboard is complicated? The incredible graphic that reveals the screens, buttons, dials and switches on an Formula 1 steering wheel Read more:* <http://www.dailymail.co.uk/sciencetec>. Available: <http://www.dailymail.co.uk/sciencetech/article-2623514/The-incredible-graphic-reveals-screens-buttons-dials-switches-Sauber-Formula-1-steering-wheel.html>. Last accessed 29th Apr 2015.
- [8] Rowe, R. (2015). *How does a Car Tachometer Work ?*. Available: http://www.ehow.com/info_12132364_car-tachometer-work.html. Last accessed 29th Apr 2015.
- [9] Rezz. (2011). *Arduino RPM Counter / Tachometer*. Available: <http://arduinoprojects101.com/arduino-rpm-counter-tachometer/>. Last accessed 29th Apr 2015.
- [10] amandaghassaei. (2015). *Arduino Bike Speedometer*. Available: <http://www.instructables.com/id/Arduino-Bike-Speedometer>. Last accessed 29th Apr 2015. /
- [11] Arduino. (2015). *Arduino Uno*. Available: <http://www.arduino.cc/en/main/arduinoBoardUno>. Last accessed 19th April 2015..
- [12] Texas Instruments. (2000). *LM2907 Datasheet*. Available: <http://www.ti.com/lit/ds/symlink/lm2907-n.pdf> . Last accessed 29th Apr 2015.

- [13] Texas Instruments. (2000). *LM3914 Datasheet*. Available: <http://www.ti.com/lit/ds/symlink/lm3914.pdf> . Last accessed 29th Apr 2015.
- [14] Texas Instruments. (1997). *High-Speed CMOS Logic 4- to 16-Line Decoder/Demultiplexer*. Available: <http://www.ti.com/lit/ds/symlink/cd74hc154.pdf>. Last accessed 29th Apr 2015.
- [15] Arduino. (2015). *Arduino Software*. Available: <http://www.arduino.cc/en/main/Software>. Last accessed 29th Apr 2015.
- [16] Electro Sensors. (2015). *Stainless Steel Hall Effect*. Available: <http://www.electro-sensors.com/products/speed-sensors/1101-hall-effect-sensor/>. Last accessed 29th Apr 2015.
- [17] Fitzgerald, S. Shiloh, M. Igoe, T. (2012). *Arduino Projects Book*. 2nd ed. Torino, Italy: Arduino. p46-47.

8. APPENDIX A - Arduino Uno Source Code for Tachometer Circuit

```
1  const int voltagePin = A0;
2  const int count = 0;
3
4  void setup () {
5      Serial.begin(9600);
6  }
7
8  void loop() {
9      int voltagePin = analogRead(voltagePin);
10     float realVoltage = (voltagePin/1024.0) * 5;
11     int RPM = realVoltage*2400;
12     Serial.print("RPM: ");
13     Serial.println(RPM);
14
15     for (int pinNumber = 2; pinNumber<7; pinNumber++) {
16         pinMode(pinNumber,OUTPUT);
17     }
18
19     if (realVoltage < 1.65) {
20         digitalWrite(6,HIGH);
21     }
22
23     if (1.65 < realVoltage && realVoltage < 2.52) {
24         digitalWrite(2,HIGH);
25         digitalWrite(3,LOW);
26         digitalWrite(4,LOW);
27         digitalWrite(5,HIGH);
28         digitalWrite(6,LOW);
29     }
30
31     if (2.52 < realVoltage && realVoltage < 3.35) {
32         for (int count=0; count<2000; count++) {
33             digitalWrite(2,HIGH);
34             digitalWrite(3,LOW);
35             digitalWrite(4,LOW);
36             digitalWrite(5,HIGH);
37             digitalWrite(6,LOW);
38
39             digitalWrite(2,LOW);
40             digitalWrite(3,LOW);
41             digitalWrite(4,LOW);
42             digitalWrite(5,HIGH);
43             digitalWrite(6,LOW);
44         }
45     }
46
47     if (3.35 < realVoltage && realVoltage < 4.17) {
48         for (int count=0; count<2000; count++) {
49             digitalWrite(2,HIGH);
50             digitalWrite(3,LOW);
```

```

51     digitalWrite(4, LOW);
52     digitalWrite(5, HIGH);
53     digitalWrite(6, LOW);
54
55     digitalWrite(2, LOW);
56     digitalWrite(3, LOW);
57     digitalWrite(4, LOW);
58     digitalWrite(5, HIGH);
59     digitalWrite(6, LOW);
60
61     digitalWrite(2, HIGH);
62     digitalWrite(3, HIGH);
63     digitalWrite(4, HIGH);
64     digitalWrite(5, LOW);
65     digitalWrite(6, LOW);
66 }
67 }
68
69     if (4.17 < realVoltage && realVoltage < 5) {
70         for (int count=0; count<2000; count++) {
71             digitalWrite(2, HIGH);
72             digitalWrite(3, LOW);
73             digitalWrite(4, LOW);
74             digitalWrite(5, HIGH);
75             digitalWrite(6, LOW);
76
77             digitalWrite(2, LOW);
78             digitalWrite(3, LOW);
79             digitalWrite(4, LOW);
80             digitalWrite(5, HIGH);
81             digitalWrite(6, LOW);
82
83
84             digitalWrite(2, HIGH);
85             digitalWrite(3, HIGH);
86             digitalWrite(4, HIGH);
87             digitalWrite(5, LOW);
88             digitalWrite(6, LOW);
89         }
90         digitalWrite(6, HIGH);
91         delay(100);
92     }

```

9. APPENDIX B - Arduino Uno Source Code for Speedometer Circuit

```
1  #include <LiquidCrystal.h>
2  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
3
4  const int voltagePin = A0;
5
6  void setup() {
7      lcd.begin(16, 2);
8  }
9
10 void loop() {
11     int voltagePin = analogRead(voltagePin);
12     float realVoltage = (voltagePin/1024.0) * 5;
13     int speed = realVoltage*20;
14     Serial.println(speed);
15     Serial.print("KPH");
16
17     lcd.clear();
18     lcd.setCursor(0, 0);
19     lcd.print("Speed: ");
20     lcd.setCursor(0, 1);
21     lcd.print(speed);
22     lcd.print("KPH");
23
24 }
```