Electrical and Electronic Engineering School

EEE2008 Project and Professional Issues

Senior Lecturer: Jeff Neasham

Team 16 – Project Thesis

Aaron Adalian – b1025407
Abdul Al-Faraj – b1014963
Jean Ubertalli – b0472997

Abstract: While there are many ways to navigate through a certain path, our toy car is capable of steering through its path using controls and subsystems that we developed in our lab. In our work, we display the importance of subsystems integration and how our subsystems can be used in real world application.

Contents

Table of Figures

Introduction

This thesis deals with understanding the importance of subsystems integration to create a whole system. This report also reflects on examining three different subsystems that have been developed for our project. And it explains why we chose this way to develop our subsystems.

The project assigned to our team was to create a toy car, which is also referred to as the buggy. It is able to navigate its rout based on a frequency wire implemented in the floor of the electronic laboratory. The wire is going around as a track.

The purpose of the project is to reflect on many aspects of real-time engineering development. Its aims and objectives can be summarised as follows:

Aims:

- Developing the team's project management and planning skills.
- Understanding different designs and choose suitable ones.
- Enhancing team skills and appreciate the responsibility and importance of individual tasks.
- Increasing the team's presentation skills through different orals and written presentations.
- Ultimately, this project gives its team members a chance to understand the value of a real-time assignment, and how engineers in industry work side by side to achieve a goal.

Objectives:

- Constructing three different subsystems:
  - Sensor subsystem, assigned to Jean.
  - Motor drive and steering subsystem, assigned to Aaron.
  - Control subsystem and interfaces to other subsystems, assigned to Abdul.
- Integrating subsystems on stages.
- Developing and enhancing subsystems based on the need of our project.
- Understanding the financial implication and follow a certain budget.
- Demonstrate the car on the track for 5 laps less than 2 minutes.

Subsystems introduction

1- Sensor Subsystem

This system is responsible for detecting the wire which carries 10 kHz sinusoidal frequency of 20 Volts Peek to Peek. The Sensor Subsystem's main job is to feedback into the Control Subsystem with a DC voltage that varies between 0 and 5 voltages. Details of the subsystem can be found under the section called 'Sensor Subsystem'.

2- Motor Drive, steering and power supply Subsystem

As the name suggests, this subsystem is the heaviest subsystem in hardware terms, in the whole system. It has different jobs unlike the previous system: this one is responsible for supplying power to all subsystems; it is also in charge of reading and interfacing with the car's main body (chassis). This system controls the wheels, the drive motor, and it feedback into the control subsystem with three different signals, as 0-5V DC signals.

3- Control Subsystem – Subsystems integration

Finally, the control subsystem is mainly to integrate all subsystems using microprocessor that can read in the entire signals and direct the car accordingly. This subsystem is the only subsystem that takes-in and gives-out signals. It was integrated with each subsystem at different stages before join in the whole system together. It was proved to be effective after testing.

Project Time Line

Our time line plan was executed according to figure 1 shown below (Gantt Chart). As a team, we anticipated that we might run into some difficulties hence we dedicated special time for such occasions.

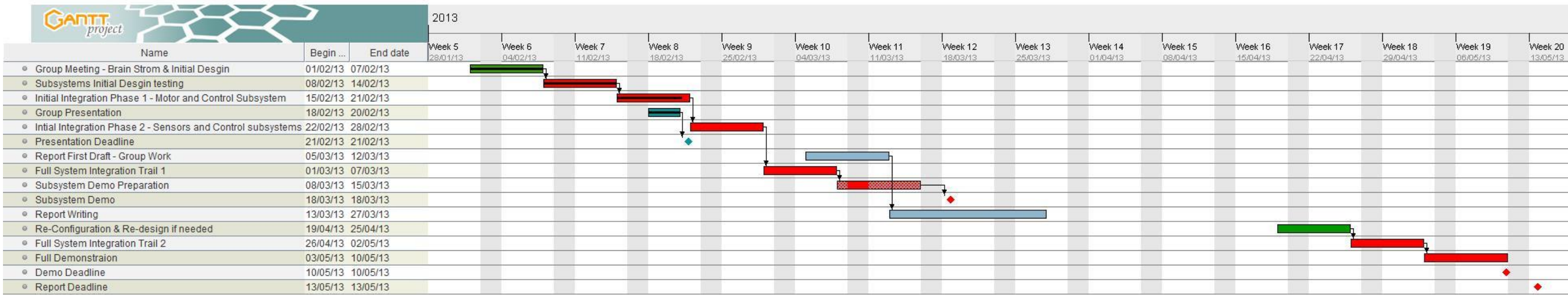

Figure 1 - Project Time Line - Gantt Chart

The red bars represent the crucial dates. IF the team missed them, the outcome of our project would have been majorly affected. The project started on the 28th of January, which is week 5 of the year but also the first week of the life line of the project.

The green bars represent the timer we allocated for preparation and unexpected incidents or delays.

The group's target for the first 3 weeks was to create initial system. The 3rd week also involved an oral presentation regarding our performance and progress. In the third and fourth week (weeks 8-9) the team tried to test the Drive Subsystem with the control subsystem, and the sensor subsystem with the control subsystem. However, issues and problems were identified.

The initial integration phases were not only to complete the physical integration, but also for team discussions regarding the subsystems and how they integrate with one another. Important dates on the Gantt chart represent the final three weeks, of which we left a week (week 17) as a 'green time' line for unexpected issues/problems before the Easter break.

The team followed the above time schedule, and due to some problems which we anticipated, we used the 'green times' to catch up and get the car back on track!

SENSOR SUBSYSTEM

1.1. Presentation

The sensor circuit does not tell you where the wheels are currently on the pointing but they tell you about the track. They provide information about on the track location. We know the magnetic field alternates 60 times per second when it is generated by an AC ferrite core coil. The changing magnetic field induces a voltage in the coil which is sufficient to be amplified, filtered and detected by the analogue circuits which has been constructed in the electronic lab. The induced voltage depends on the magnitude and direction of the magnetic field on the track. The amplitude of the induced signals in each sensor is fed back to the PIC through the analogue circuits.

1.2. How it works on the track

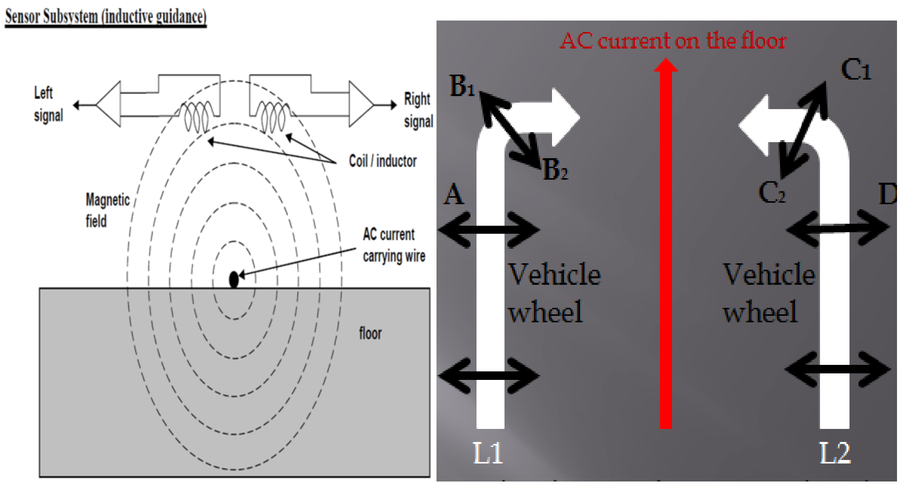

Figure 2 - At left illustrates the AC current carrying wire on the floor producing magnetic field in the coil. Figure2.At right shows magnetic field Ed which is the ratio between induced EMF at L1 andL2.

According to Faraday Law's the voltage induced in the coil is the rate of change of the flux linked to the coil with time. Bearing this in mind, magnetic field diminishes with increasing distance from a strait current carrying wire. What does happen on the track? At A and D, the magnetic field Ed is at unity and flux is stronger hence, the vehicle steers straight, when the steering gets to C, Ed>0. This will weaken the magnetic field strength at B forcing the vehicle's wheel to turn faster at C. H is constant in magnitude around the path and is acting in the direction of the path every point on it. At any given location in the vicinity of a current carrying -wire, the magnetic flux density is directly proportional to the current in amperes.

1.2.1. Position and orientation of sensor

The amplitude of the induced signals in each sensor tells us what angle we have to be. As observed during so many different trials, the vertical orientation is more successful because the magnetic field strength decays when you go away from it. We notice relationship between field on the track and sensor is so complicated.

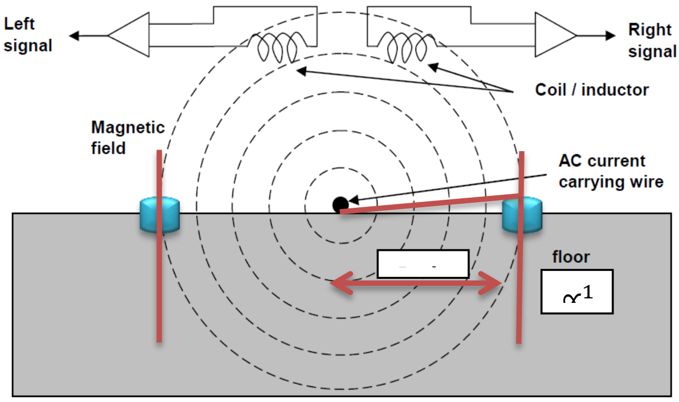

Figure 3 - Illustrates vertical orientation of sensors

The position and orientation of sensors are extremely important as it allows the coils to capture maximum magnetic field when they are in parallel.

1.3. System design and manufacturing techniques

1.3.1 Manufacturing process

Before, we commit our self to producing either a printed circuit board or any other form of hard wired circuit. We firstly test the circuit full working action in the bread board many times. Signals generated are in different waveforms, and oscilloscope, signal generators and millimetres are used to see how the circuit behaves before making any progress. It is worth to use it as it makes easy for creating temporary prototypes and experimenting with the circuit system. It allows us re –use and move components and connections around to correct any malfunction. It enables implementing the circuit using different kind of components and see how are they going to react on the circuit.

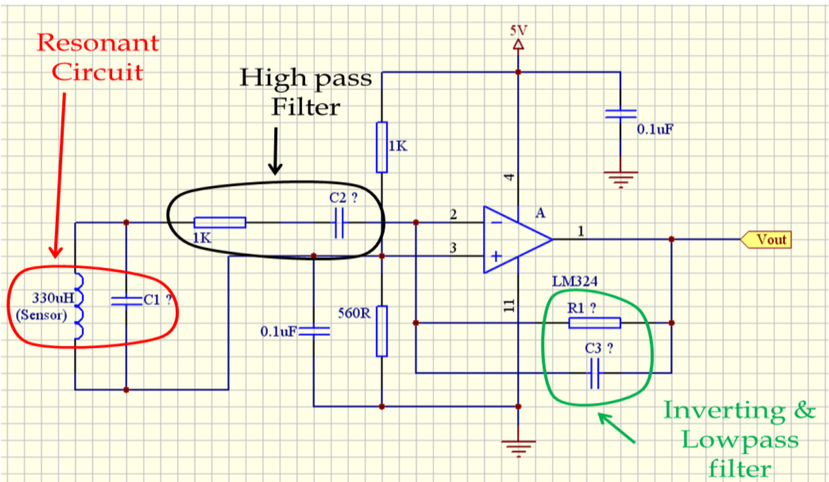

Figure 4 - Illustrates different parts of the sensor circuit to be manufactured

I want to ensure this design system is made up and controlled to the quality standard required in the lab sheet. I am responsible for ensuring the complete sensor effectiveness is designed to operate and deliver required outputs to other subsystems such as subsystem control. How do I know our design system is safe? I can find out if my sensor circuit is safe by carrying out suitable checks, such as inspection and testing. The level of testing this design system is upon any risks. I write down in my lab book according to the project requirement how often each stage of sensor circuit construction should be checked and results write down for next process.

1.3.2. Resonance

This is the natural frequency of this system. In our case, it is establishing the condition of stable frequency in circuits designed to generate AC signal needed to be amplified, filtered and detected. We use a parallel tank circuit for this purpose, with capacitor and inductor directly connected together, exchanging energy between each other. As we notice while building the circuit, the frequency set by the tank circuit is only dependent upon the values of L and C, and not on the magnitudes of voltage or current present in the oscillations. Circuits resonate at national frequency where capacitors are charged. C1is calculated using the formula.

- C1 =$\frac{1}{(2\pi f)^2 l} = \frac{1}{2\pi\times10^4\times330\times10^{-6}} = 767nF \approx 800nF(330+470)nF$

1.3.3. High pass and low pass filter

High pass filter when used with sinusoidal signal as in our case, its purpose is to allow high frequency sine waves to pass from its input to its output, but to reduce the amplitude of lower frequency signals. The high resistance in series tends to block low-frequency signals from getting to the load. Conversely, lower pass filter are used to pass low frequency sine waves and attenuate high frequency sine waves. However, it is possible to design a wide variety of filters with different levels of gain and different roll off patterns using op amp. C1 and C2 are calculated:

- C2= $\frac{1}{(2\pi f)^2 R1} = \frac{1}{2\pi\times10^4\times1000} = 15.9nF \ \approx 15nF$ (high pass)
- C3=$\frac{1}{(2\pi f)^2 R2} = \frac{1}{2\pi\times10^4\times10^5} = 159pF \ \approx 150pF$ (low pass)

We benefit from the use of low pass band as an integrator circuit and high band as a differentiator which results from making the feedback path frequency dependence, we just use single capacitor RC filters. By doing so, we obtain derivative and integral operation mode of the PID. Low and High active filter in inverting mode configuration allows us to adjust Kp, Ki and Kd.

1.3.4. Inverting operational amplifier Gain

We know that the open loop DC gain of op amp is extremely high we may therefore try to lose some of this high gain by connecting a suitable resistor across the amplifier from the output terminal back to the inverting input terminal as done the circuit to both reduce and control the overall gain of the amplifier. This produces an effect known as negative feedback. It is and advantage to generate a very suitable operational amplifier system as observed during testing in the electronic lab. Feedback resistance is calculated:

- R2 = R1$\times$ $Gain = 1000 \times 100 = 100K\Omega$ (where R2 = 1000Ω and gain = 100)

The introduction of capacitor modifies the frequency (increases or decreases)

1.3.5. Precision rectifier

Why precision rectifier is it used in the design system of this sensor circuit. As the sensors must feed the control system with analogue signals of 1.8V DC from each sensor that is the main reason we use the half wave rectifier, I can choose to keep one polarity while discarding the other. The half –wave rectification receives an incoming waveform from the output of sensor and as usual with op amps, inverts it. However, only the positive –going portions of the output waveform, which corresponds to the negative-going portions of the input signal, actually reach the output. The direct feedback diode shunts any negative going output back to the input, preventing it from being reproduced.

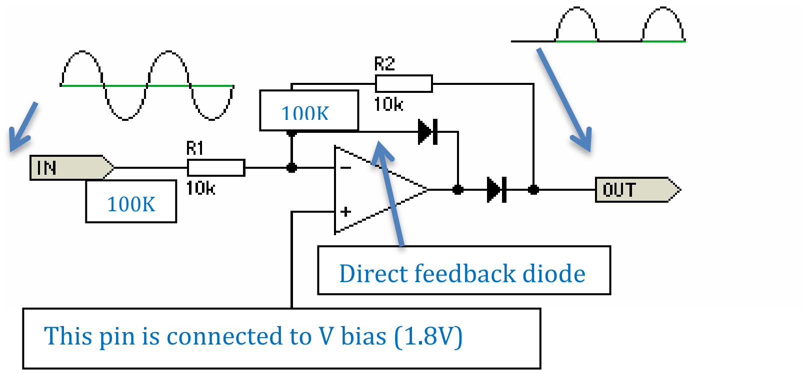

Figure 5 - Shows the half wave precision rectifier designed to get rectified voltage

Furthermore, since the output voltage is taken from beyond the output diode itself, the op amp will necessarily compensate for any non-linear characteristics of the diode itself. As a result the output voltage is true and accurate (but inverted) reproduction of the negative portions of the input signals. If want to keep the positive –going portion of the input signal instead of the negative-going portion, simply reverse the two diodes.

1.3.6. Filter

Finally in this design system I add at the output of a precision half-wave rectifier a low pass filter to get rid of ripples, so I can have DC signal to feed to the control subsystem.

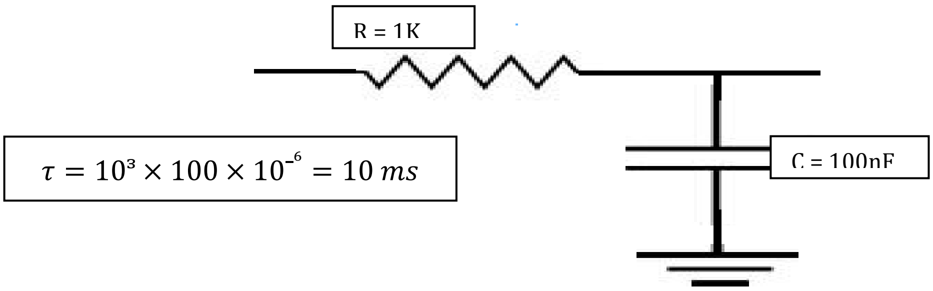

Figure 6 - It shows low pass filter used in the circuit for smoothing

It is a big deal to get the right value of R and C for this circuit. I observe a big capacitor filter reduces ripples but it can also affect the time reaction of the sensor, so the time reaction can become very slow, contrary to a small capacitor will not reduce the ripple very well. I get it by trial on the bread board and observing the reaction of my sensor on the oscilloscope.

1.4. Evaluation and Testing

We evaluate our design system through experimentation and trials using bread board. We try to our best to interpret correctly obtained waveform and values to progress with the subsystem. Being able to evaluate your own work is difficult we believe we have done in clear and concise way. Results from testing different parts of the sensor are shown:

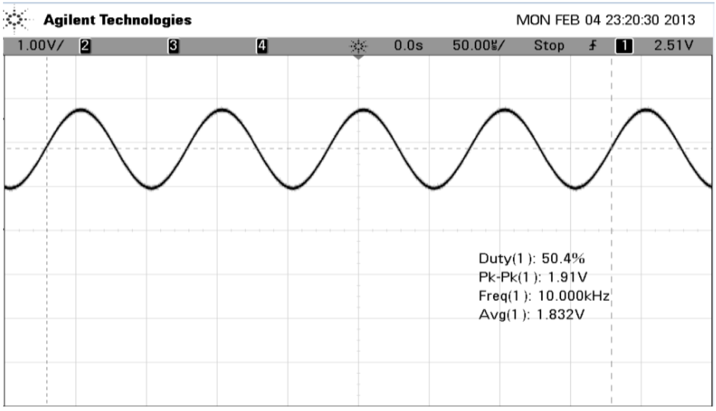

Figure 7 - waveform at the output of the amplifier circuit 1.83V

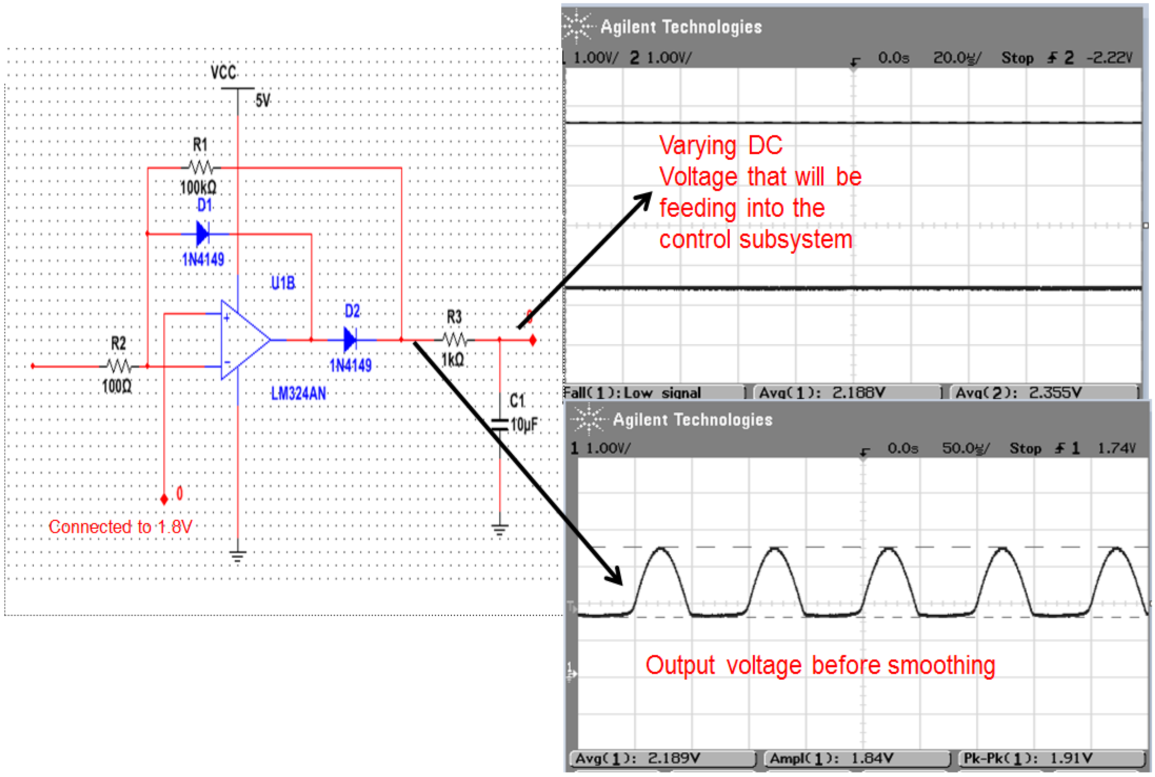

Figure 8 - different stages of the sensor waveforms obtained

We would say we had so many testing before obtaining good and strong waveforms from various parts of this circuit.

1.4. Implementation

The question is what we have done to implement the circuit design. The implementation of the circuit design has been achieved from our part of the work. The sensors are really sensitive and sending information in real time. They are behaving on the track as expected by providing information about the location. We choose this implementation:

- For the feedback diodes of the precision half wave rectifier we use Schottky diode to get out some of its advantage such as: high speed, high frequency, low forward voltage drop, low heat dissipation and low loss energy.
- Through carefully design of the power supply network, correct chip functionality can be ensured. In order to achieve an acceptable level of voltage fluctuation in the power supply network, a sufficient amount of decoupling capacitance must be allocate. The capacitor acts as local reservoir for high-frequency circuits and reduces the effects of power supply noise on neighbouring circuits. Thus, we connect the decoupling capacitor near the chip to make the circuit very effective.

Motor Drive, Steering and power supply subsystem

In this subsystem of the project the motor drive, steering and power supply subsystem was designed and constructed. The purpose of this subsystem is to give an exceptional control of the motor drive which controls the speed and the direction (forward or reverse) of the buggy. The motor steering part, which is controlled by the same motor control (IC) as the drive, is controlling the direction of the front wheels and hence the direction of the buggy turning left or right. Also the power supply subsystem was designed to enable or disable bits of the motor control (IC) and to supply the other two subsystems of the buggy with the needed voltage.

The basic circuit design for the motor control IC, A 4973, was supplied by the school. This diagram was duplicated as the need for constructing two motors was necessary for controlling the drive and steering motor.

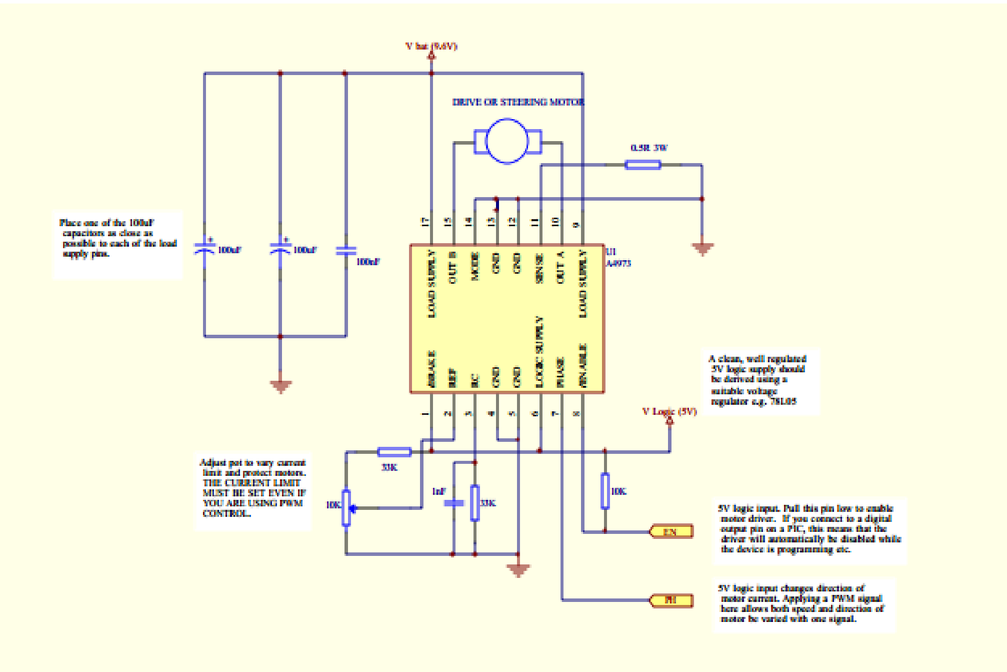

Figure 9 - circuit diagram given from the school for the control of the motors

The A4973 IC is a full bridge PWM (Pulse Width Modulation) motor driver. The output of the IC, which are pins 10 and 15, are controlled by a PWM signal which is applied at pin 7, the phase pin. Basically the average voltage of the PWM signal is read by the phase and as that voltage varies the output voltages on pin 10 and 15 are linearly varied thus controlling the steering or drive motor that are connected on. This way a full bi-directional speed and steering control is achieved. The voltage outputs of 10 and 15 pins are linearly inverse, when the PWM is on 50% both have the same value but with the increase or decrease of the PWM, the voltage of pin 15 increases while the voltage of pin 10 decreases and vice versa. At this point a variable resistor is used to set the reference pin, pin 2, to limit the peak load current so that the motor or steering mechanism is not damaged by exceeding its limit. This can happen as trying to drive the steering past full lock. Varying the variable resistor hence the dc voltage on the ref input is also varied and thus control of the speed of the motor drive is gained as well.

As the control system can supply only one PWM signal, that signal (the PWM of the control subsystem) is supplied to the motor IC in control of the motor steering as it needs a precise handling for turning on the track. Instead at the motor drive, a high or low (0 or 1) is applied, from the control subsystem, in order to control only the direction of the buggy, forward or reverse. The speed is controlled by the variable resistor. It is more important to have a precise control of steering instead of the drive motor.

Three pins needed to be controlled, the enable pin, the brake pin and the mode pin.

The enable pin, pin 8, enables and disable the IC control, if 0 is applied to the enable pin then the motor is enabled thus the motors are running and if a 1(in this case a 5v is applied) is applied the IC control is disabled stopping the motor to run and hence the buggy stops. The control of this pin was under the control of the control subsystem in order to get the buggy start moving (supplying pin with low) or in the off chance that went off track to stop it supplying it with a high.

The brake mode is enabled when low (0) is applied to pin1. When this function is applied it overrides enable and phase to turn off both source drivers and turn on both sink drivers, which stops the car moving right away. To turn the brake function off, a high is applied to pin1. The brake function was constantly disabled in this particular project as it was unnecessary for the time being to be used. If the buggy went off the track, enable pin (pin8) would be supplied a high from the control system thus the buggy would stop.

The sense pin (pin11), which is connected to the ground through a low value resistor, is a current sensing pin. The current detect through this pin is taken into account, by the IC control, when determining the impedance of the external circuit that sets the reference voltage value. A small value resistor is used to reduce heating and power loss and would keep the voltage rating below the maximum rating allowed to the sense terminal. The value of the resistor (0.5Ohms 3W) is given from the manufacturer of the IC control.

The mode pin, pin 14, in forward or reverse mode the current-control circuitry limits the comparator resets a latch that turns off the selected source driver or selected sink and source driver pair depending on whether the device is operating in slow or fast current-decay mode, respectively.

Decoupling capacitors are used on the power supply rail to reduce interference with other subsystems

For the power supply subsystem the voltage regulator 78L05 which lowers the input voltage supplied to it by 9.6Volts to 5Volts is used. This particular regulator is used as the two other subsystems, sensors and control, both need a 5Volts supply. Also a 5Volts logic supply is used to the pins needed to be high on the IC control.

The steering motor assembly includes a potentiometer to provide position feedback; this feedback must be used to enable precise steering adjustment as it gives information of the exact direction of the steering. This feedback signal is supplied to the control subsystem, which is in control of the positioning and the steering of the buggy, and hence can determine the exact position that the buggy is heading to.

Subsystem Testing

Furthermore the design and strategy of approaching the project was done the actual circuit was constructed on a protoboard for testing and fully understanding the properties and applications of the IC control. Also testing was carried out to check if any alterations in the circuit or the interactive parts of the three subsystems (control, sensor and motor) were needed.

When the motor circuit was successfully constructed, a 9.6Volts (the same voltage as the battery used on buggy) was supplied to the circuit. With the use of a signal generator, which it was tuned to generate a PWM at 10kHz the steering and motor drive was tested applying this PWM to the phase pin (pin 7) and varying the PWM. As expected the speed and direction of motor drive and of steering motor was also controlled. Each motor was controlled by a different generator. If the PWM was more or less than 50% of duty cycle, the drive spins forward or reverse and the steering turns right or left. The way that this happens (70% forward, 30% reverse) dependent on the way that the output of the IC were connected to the motor. If we change the connections to the motor from the output of the IC then the direction of the motor, right or left would change. Also the output of the voltage regulator was checked to ensure that outputs 5Volts.

Then at the potentiometer, which is supplied with 5V, the values of the feedback were taken. When the steering was full lock to the right the feedback voltage was found to be 4.04V, at canter where the buggy goes straight is was 2.6Volts and at full lock left was 0.94Volts.

After the adjustments of how the interconnections between the subsystems would be done and decided, the circuit was constructed on a veroboard. When it was successfully constructed on the veroboard, the same tests were carried again. The circuit seemed to work properly but at some point the speed of the motor drive it was not constantly controlled. After some tests the problem was solved as some soldering was not good and the signal was sometime lost. When the circuit was finally placed on the buggy the screws used to screw the board on the buggy was short circuiting the circuit. To face this problem the holes were widened on the board and also plastic screws were used. Continuing the testing with the control subsystem the microchip of the control was burned. Investigating why this happened it was found out that the voltage regulator was defected and at some point it started exceeding the 5Volts output it should have been supplying and thus the microchip was burned. The voltage regulator was replaced by a new one. Last but not least when finally all the subsystems were put together the steering was turning on the left very slow comparing it with the right turn. Furthermore some tests were carried out and the conclusion that there must be something wrong with the steering motor of the buggy. The assistance of the technicians was asked. After inspecting the buggy they found out that the steering motor had a problem and it was fixed.

Control Subsystem

All subsystems are needed to be integrated to make one functional system. This is where the control subsystem important role lies. It uses information from different sources to behave accordingly to the input data. It controls other subsystems based on feedback that either been fed from the same system or another.

The subsystem uses a microchip, also known as Microcontroller, PIC18F1220. This PIC features good characteristics that allowed the signals from the other two subsystems to feedback. Figure below shows the inputs and outputs of the microcontroller. The important features that worth mentioning of this PIC:

- It contains analogue inputs with 10 bits A/D.
- It generates PWM (Pulse Width Modulation) of frequency of my choice.
- It have varies output that been used for different purposes.
- Programmed using C, with C18 compiler.
- To programme the PIC, we used SIL header, which then interfaced with ICD3 programmer.

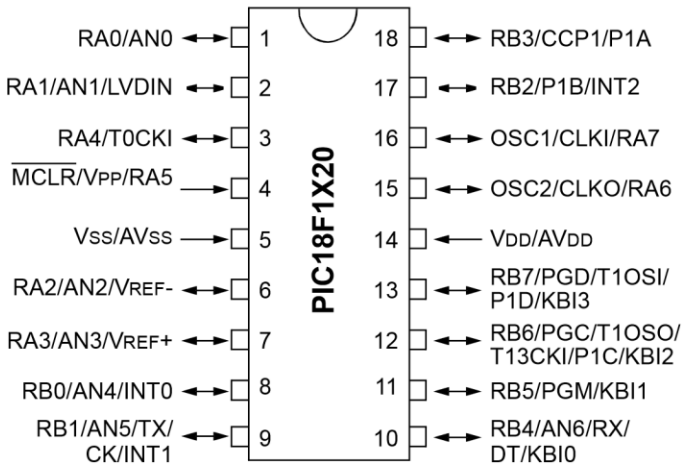

Figure 10 - Microcontroller PIC18F1220

The Microcontroller is the most important hardware kit in this subsystem, and before looking into the circuit designs of this subsystem, the following points explains the importance and the roles of this subsystem:

- Receiving three different DC signals which varies between 0-5 Volts,
  - Left sensor, fed into analogue channel 0
  - Right sensor, which is fed into analogue channel 1
  - Wheel's potentiometer, fed into analogue channel 2.
  - Extra potentiometer for testing, fed into analogue channel 3.
- Control the wheels direction using PWM
  - Pulse Width Modulation signal is fed back to the Drive Motor Subsystem using RB3, or as it's called P1A, which is the first channel for PWM output.
- Enable/Disable the back wheels
  - First signal fed back to the Motor Drive Subsystem for output RB0. This signal disables the motor if it is high. For live debugging I connected it also to green LED.
- Second signal, which is also fed back to the same subsystem, controls the direction of the back wheels. High logic output means forward, and zero logic output will result to make the wheels go backward. It uses output RB1 which is in turn connected to red LED for debugging.

Circuit Design

My choice of design of the circuit was meant to be simple. Most of the work involved in this circuit is simply wiring outputs and inputs correctly. Previously, I mentioned the roles of the PIC alongside what inputs and outputs are used for the circuit. The figure below shows the actual hardware coming in and out from my subsystem.

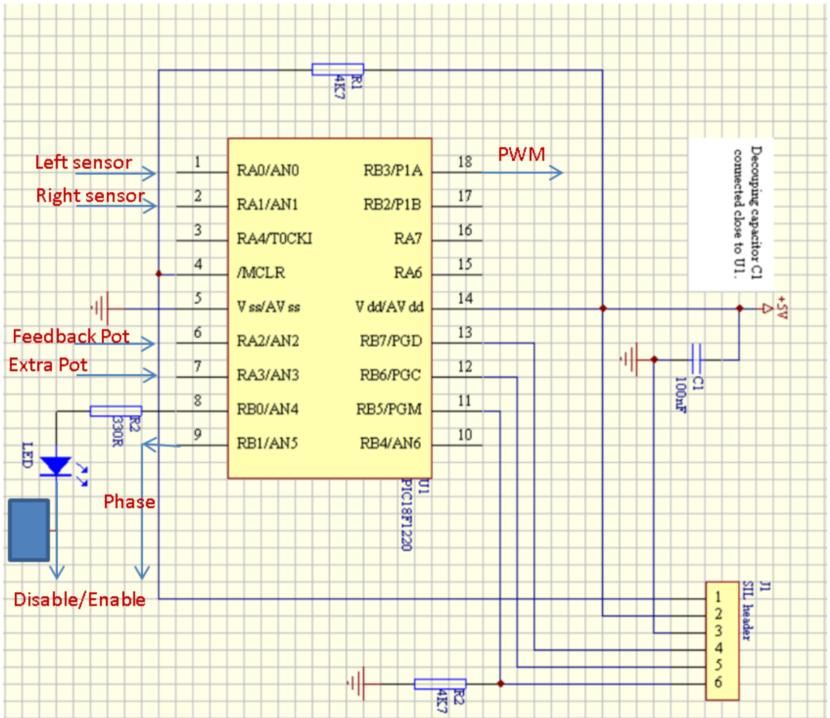

Figure 11 - Control Subsystem Circuit Design

There are extra components on the circuit which are not needed for the application but only for the debugging of the application. Pin RA3 connected to extra potentiometer which was used to test the response behaviour of the PWM before testing it on the wheels. Extra two LEDs were installed on the circuit that are connected to two outputs, RA0 and RA1. They helped during the first stages were no other subsystems were ready.

Software Design – and Software testing

This system depends heavily on the software design rather than the hardware. Throughout the life of the project, the programme was written gradually to accommodate certain tests and conditions; however the final code is attached in the index.

This section is intended to explain different phases of the programme and to explain different parts of the final code and the reasons behind certain choices.

1- Initial programme

The initial code was given, also known as the Skelton code. The only reason behind this code is to understand the basics of inputs and initializations of output pins. It was only capable of switching the LEDs of and on.

2- Pulse Width Modulation

PWM is fed back into the phase of the microchip which is used to control the motor of the front wheels. To explain how PWM works, before explaining the programme, figure below displays how can varying the output effect the direction of the wheels. 50% duty cycle this results in roughly 2.5V output. On the other hand, either extreme of duty cycles can results in either high voltage or low voltage.

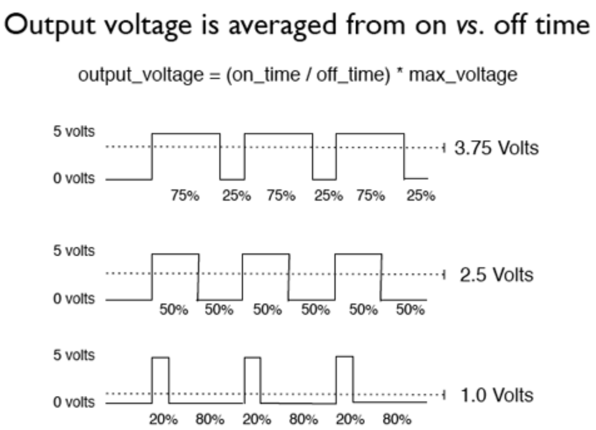

Figure 12- PWM Duty Cycle On/Off

In our design, over 50% of duty cycle can affect the car to move right, on the other hand anything less than 50% duty cycle effects the car to be moved to the left, of course other important variable are taken in consideration such as feedback position of the wheel and their current position.

For the buggy, we chose the frequency of the PWM to be 30 kHz. This can guaranty that no effect on the performance of the sensor. This particular choice of frequency is well above 10 kHz and it have a safe rang between the two frequencies. The PIC Manual gave us the formula to calculate the frequency as follow:

$$PWM\ Period = [(PR2 + 1] \times 4 \times TOSC \times (TMR2\ Prescale\ Vaule)$$

Whereas, PR2 is the value which will set within the programme to give the required frequency and TOSC is the internal 8 MHz clock. Finally the presale value by default is 1. Hence, we know the required PWM Period 30 kHz which results in the period of 33.3 microseconds. Therefore, the only left variable is the PR2, which is needed for the programme to be set.

After arranging the formula above, PR2 ≈ 66

Finally calculating duty cycle at 50%, this is very essential for the PID Control to work correctly as explained in the following section.

$$16.6\mu s = duty \times TOSC \times (TMR2\ Prescale\ Vaule)$$

The reason why 16.6 microseconds was chosen is because it is the half of the period of the frequency, which results in 50% duty cycle.

Duty Cycle	Voltage out	Programme Value
100%	≈ 3.8	266

50%	2.5	133
0%	1.0	0

3- Analogue to Digital

As mentioned before, this PIC can change analogue DC signal to sampled Digital signal. This property been used in four functions:

Function	Minimum DC, V	Maximum DC,V	Others	A/D range	Comments
Left_Sonsor()	1.7 (Not sensing)	3.3V (Sensing)		375 - 645	
Right_Sensor()	1.7 (Not sensing)	3.1 (Sensing)		375 - 645	
getPot()	1.3 (Left)	3.7 (Right)	2.7V (Straight)	312-712	
SP()	0	5		312-700 (limited for safety)	For testing

To find any value A/D value, the PIC samples at 10 bits/s: $2^{10} - 1 = 1023 \rightarrow \frac{5v}{1023} = 0.0048$ Then multi any DC value with this result gives the value of the A/D conversion. This has helped dramatically throughout the project period and without ADC within the PIC it would have been very difficult to make this car working.

4- Bang-Bang Programme

For initial programme, we used the bang-bang approach. It was slow and gets off the track easy but it gave good insight on the car behaviour. The bang-bang does not use the feedback on the front wheels position, and only sets the PWM to the either max based on one of the sensors. The main programme of the bang-bang is shown below:

```
While(TRUE){

    PORTB=0x02; // to enable the motor and put it forward

    if(Left_Sensor()>600){
        SetDCPWM1(312);
    }else{
        SetDCPWM1(712);
    }
}
```

This approach disallowed us from increasing the speed of the car, and it was very week because it only deepened on 1 feedback component. On the second line of the programme, we used to set more than one output port together, as we have found problems setting them separately.

5- PID Control

The final control method we explored was Proportional Control. This technique used all the variables correctly and made use of the wheels position feedback. As first step I used the extra pot to set the Set Point in our programme and limited it for safety as follow:

```
SetPoint = SP();

if (SetPoint > 712){
    SetPoint = 712 ;
}else if(SetPoint < 312){
    SetPoint = 312 ;
}
```

Furthermore, using the SP function to set where the wheels turn instead of using the sensor, it helped testing the wheels response. It also eliminated any chance of errors that the sensor might have had at the time. After testing, the programme been changed to accommodate the feedback from the sensors as the following:

```
SetPoint = 512 -0.7*(Left_Sensor()-Right_Sensor());

if (SetPoint > 712){
    SetPoint = 712 ;
}else if(SetPoint < 312){
    SetPoint = 312 ;
}
```

The set point was constructed based on the equation above. It simply takes the difference of both sensors multiply it my constant which effects the response sensitivity for the difference between both then subtract it by 512. 512 is the value for straight. The response of the difference was set under 1 because on the track it was proved effective.

On the other hand, Proportional Control was the next step. PID Control is widely used in so many applications that are valuable to so many engineering industries. This type of control is popular because of the features that it offers; most importantly accuracy. For the purpose of our design, we only chose to use Proportional Control and not full PID. The reason being is that it was good enough for this task, and we have to consider time. After a research that been carried out we accepted that this design can use PID but we decided if there was time, we would have implemented.

For Proportional Control, I used the following formulas and codes to implement it:

```
error = SetPoint - getPot();
duty = 133 + (error * kp);

if(duty > 266){ duty = 266; }
else if (duty < 0){ duty = 0;}


SetDCPWM1(duty);
```

Once again, we limited the duty cycle in case it gets out of the PWM's capacity. This also helped us to change Kp (which is the constant) as we please without worrying to damage the car. The error is calculated by subtracting our destination from the current position which is fed into the PIC by getPot() function. The proportional constant (kp) was chosen to be 0.9. This constant affects the speed at which the wheels would change from one direction to another. This value proved that it will give the least overshoot and the fastest and safest outcome.

6- Extra Safety

As extra safety measurement, I used the two sensors to disable the motors' chip if neither of the sensors are sensing, this is established as follow:

```
while(Left_Sensor()<370 && Left_Sensor()<370){
    PORTB = 0x01;
}
```

Testing & Results

Each subsystem's testing was done respectively in that section. This section deals with the overall testing. The car was complete by the final two weeks. However, adjustments to the sensors' position been carried out and changing the constants (e.g. KP) to achieve an overall good performance.

1- High sensitivity and low KPs

The KP donated for the proportional constant that affects the reaction of the wheels to the SP and the sensitivity is the constant that controls the sensitivity to the difference between the two sensors.

This configuration resulted in losing the SP because of the slow reaction of the car's wheels to the set point. The car could not react fast enough to reach certain set point. However, it is important to notice that it works for a very slow speed.

2- High sensitivity and high KP

However, raising the proportional constant to high do not solve the problem of slow reaction; it caused the car to overshoot, and even react faster which lead the car to go off the track. Once again, this tactic did work with a very slow speed: over 25 seconds per lap. This is not good enough as the minimum timer per lap is only 24 second.

3- Low Sensitivity and High KP

Conversely, by trial and error, we tried to lower the sensitivity value of the difference between the sensors, and keep the wheels response high. We were very close, but even though, this produced very poor behaviour occasionally on high speed. We achieved 2 laps each less than 14 seconds but the buggy left the track because it picked up even more speed and the Set Point lost track of the sensor difference.

4- Low Sensitivity and low KP

Finally, the best choice we were left with is lowering the KP gradually to achieve the most stable laps. The sensors' position was very important and they were adjusted accordingly in every step of testing different variables. The final configuration gave us the opportunity to compete in the race. In our second attempted we train to increase the sensitivity and the speed, which cause the car to lost sight of the track and hence we lost second attempt in the track leaving us with a record of roughly 38 seconds for 2 laps. Full configuration of the code can be found in the index.

Conclusion

For over one complete term, team 16 strived to prove themselves on the track. Many challenges we overcame and many other issues we solved. As first-hand experience and first time to work in group to establish such important task, the team developed skills and enhanced our engineering experience.

The team best un-official record on the track was 13 seconds per a lap. The official timing was 17 seconds per a lap. The reason for such difference was because we wanted to guarantee 5 laps less than 2 minutes without risking the car to get off the track.

Objectives achieved:

- ✓ Constructing three working subsystems.
- ✓ Constructing a car that preformed 5 laps less than 2 minutes.
- ✓ Participating in the race, performance of 38 seconds for 2 laps.
- ✓ Perform oral and written presentation skills.
- ✓ Follow financial budget and remain under the cost.

Aim Achieved

- ✓ Enhancing our team work, throughout this project team work was the first and most important element for the success of our project.
- ✓ Ensuing time plan, predicating any problems and giving each issue its valuable time.
- ✓ Increasing the team's understanding of team work and accepting the individual tasks assigned for each member. Without such attitude, this project would not have been able to continue until the end.

Not only did we achieve our intended aims and objectives, but we also achieved awareness of the systems that we used and which will undeniably be widely implemented in our fields. The project might have been just a toy car, but the integral systems and methods used, such as PID Control and Sensors Systems are vital to everyday life aspects. Hence, dealing with the latter and learning how to manipulate them taught us bigger and vaster lessons beyond the academic aspects that we thought we would learn.

To conclude, we feel very honoured to have had the chance of undertaking this important task that gave us a crucial insight of real-time engineering projects and development stages. From designing phases to implementations until final moments of demonstration and racing; our team held the responsibility of delivering what been assigned to us, and proudly delivered the optimum subsystem in the little time that we had.

Costing

	COMPONENTS	Unit Price	Total price
	MOTOR SUBSYSTEM		
4	Electrolytic 100μF	£0.05	£0.20
1	Ceramic 100nF	£0.10	£0.10
1	Ceramic 1nF	£0.10	£0.10
1	Resistors 33K	£0.10	£0.10
1	Resistors 10K	£0.10	£0.10
1	Resistors0.5R3W	£0.10	£0.10
2	Potentiometer	£0.35	£0.70
2	Chip A4973	£2.75	£5.50
	Stabilisation		
1	Bead cap100nF	£0.15	£0.15
1	TransistorL5S	£0.20	£0.20
1	Bead cap1μF	£0.10	£0.10
	Control subsystem		
1	PIC 18F1220	£3.00	£3.00
1	SIL header 8Way	£0.15	£0.15
2	Connector	£0.15	£0.30
2	LED	£0.10	£0.20
2	Ceramic cap 100nF	£0.10	£0.20
2	Resistor 330K	£0.10	£0.20
2	Resistor 4.7K	£0.10	£0.20
Sensor subsystem			
2	Ferrite coil	£0.60	£1.20
2	Polyester cap330nF	£0.15	£0.30
2	Polyester cap470nF	£0.20	£0.40
1	Op ampLM321	£0.45	£0.45
1	Ceramic cap15nF	£0.10	£0.10
1	Ceramic cap150nF	£0.10	£0.10
3	Resistor100K	£0.10	£0.30
2	Resistor1K	£0.10	£0.20
2	Ceramic cap0.1μF	£0.10	£0.20
2	Scott diode	£0.15	£0.30
2	Normal diode	£0.10	£0.20
1	Ceramic capacitor 10μF	£0.10	£0.10

	Extra materials used		
1	Rj12 connector	£0.70	£0.70
2	M3 threaded 20mm	£0.15	£0.30
2	M3 threaded 10mm	£0.20	£0.40
4	M3 Nylon washers	£0.20	£0.40
3	Vero board	£0.80	£3.20
	Total price		£20.45

Index

Final Code

```
//****************************************************
//*** Final Prgramme for EEE2008 Buggy project ***
//*** For PIC18F1220 and Microchip C18 compiler  ***
//*** Team 16- Written by Abdul Al-Faraj          ***
//****************************************************
#define USE_OR_MASKS
#include "p18f1220.h"          //include file for port definitions etc
#include "delays.h"            //include file for C library functions
#include "pwm.h"
#include "adc.h"


//****************************************************
// InitIO function (initialises ports etc)
//****************************************************

//Prototybes
signed int Right_Senor(void);
signed int Left_Sensor(void);
signed int getPot(void);

//Declaring Variables
signed float SetPoint=0.0,kp = 0.8;
signed double error,duty;
```

```c
//rightt sensor A/D
signed int Right_Sensor(void) {
    signed int ADCResultR = 0;

    unsigned char channel = 0x00, config1 = 0x00, config2 = 0x00, config3 = 0x00, portconfig = 0x00, i = 0;

    config1 = ADC_FOSC_2 | ADC_RIGHT_JUST | ADC_2_TAD;
    config2 = ADC_CH1 | ADC_INT_OFF | ADC_REF_VDD_VSS;
    OpenADC(config1, config2, portconfig);

    ConvertADC();
    while (BusyADC())
        ;
    ADCResultR = ReadADC();
    CloseADC(); //turn off ADC
    return ADCResultR;

}
//left sensor A/D
signed int Left_Sensor(void) {
signed int ADCResultL = 0;
unsigned char channel = 0x00, config1 = 0x00, config2 = 0x00, config3 = 0x00, portconfig = 0x00, i = 0;

config1 = ADC_FOSC_2 | ADC_RIGHT_JUST | ADC_2_TAD;
config2 = ADC_CH0 | ADC_INT_OFF | ADC_REF_VDD_VSS;
OpenADC(config1, config2, portconfig);

ConvertADC();
while (BusyADC())
    ;
ADCResultL = ReadADC();
CloseADC(); //turn off ADC
return ADCResultL;

}
```

```
//Extra variable resistor for testing and debugging - NOT needed for the acutal application
int SP(void) {

    signed int ADCResultSP = 0;
    unsigned char channel = 0x00, config1 = 0x00, config2 = 0x00, config3 =
    0x00, portconfig = 0x00, i = 0;
    config1 = ADC_FOSC_2 | ADC_RIGHT_JUST | ADC_2_TAD;
    config2 = ADC_CH3 | ADC_INT_OFF | ADC_REF_VDD_VSS;
    OpenADC(config1, config2, portconfig);

    ConvertADC();
    while (BusyADC())
        ;
    ADCResultSP = ReadADC();

    CloseADC(); //turn off ADC
    return ADCResultSP;
}
// feedback from the wheels dirction
signed int getPot(void) {
    signed int ADCResult = 0;
    unsigned char channel = 0x00, config1 = 0x00, config2 = 0x00, config3 =
    0x00, portconfig = 0x00, i = 0;

    config1 = ADC_FOSC_2 | ADC_RIGHT_JUST | ADC_2_TAD;
    config2 = ADC_CH2 | ADC_INT_OFF | ADC_REF_VDD_VSS;
    OpenADC(config1, config2, portconfig);

    ConvertADC();
    while (BusyADC())
        ;
    ADCResult = ReadADC();
    CloseADC(); //turn off ADC
    return ADCResult;
}
//PWM initliaztion
void initPWD(void) {

    char period = 0x00;
    unsigned char outputconfig = 0, outputmode = 0, config = 0;
    unsigned int duty_cycle = 0;

    //----Configure pwm ----
    period = 66;
    OpenPWM1(period); //Configure PWM module and initialize PWM period

    //-----set duty cycle----
    duty_cycle = 133; //initially set it to 50% duty cycle, cacluation of this can be found in the report.
    SetDCPWM1(duty_cycle); //set the duty cycle

    //----set pwm output----
    outputconfig = SINGLE_OUT;
    outputmode = PWM_MODE_1;
    SetOutputPWM1(outputconfig, outputmode); //output PWM in respective modes
}
```

```c
//Ports initilaizations
void InitIO(void) {

    OSCCON = 0x72; //set clock to internal 8MHz (2MHz instruction cycle)
    TRISBbits.TRISB0 = 0; //set RB0 to output (all ports are inputs by default)
    PORTBbits.RB0 = 0; //set RB0 to logic zero

    TRISBbits.TRISB1 = 0; //set RB1 to output (all ports are inputs by default)
    PORTBbits.RB1 = 0; //set RB1 to logic zero

    PORTB = 0x01;   //TO DISABLE THE CAR AT THE BIGGING
    initPWD();
}
/**
//important information
//Pot Left DC = 355 --> 1.3V
//Pot Right DC = 755 --> 3.7V
//Pot straight = 470 --> 2.3V
//LEFT Sensor = 3.3V while sensing therefore 653 DC level
//Right Sensor= 3.1 while sensing therefore 633 DC level
//both are 1.9V no sensing which is 388 DC level
**/

 //*****************************************************
 // main function
 //*****************************************************

 void main(void){
     InitIO();
     kp = 0.9;
     //forever for loop
     for(;;) {

         //enable the the car and put it in forward dirction
         PORTB = 0x02;

         //calcuate the set point
         SetPoint = 512 -0.7*(Left_Sensor()-Right_Sensor());

         //limit the set point for safety
         if (SetPoint > 712){
             SetPoint = 712 ;
         }else if(SetPoint < 312){
             SetPoint = 312 ;
         }

         error = SetPoint - getPot();
         duty = 133 + (error * kp);

         //limit the duty for safety
         if(duty > 266){ duty = 266; }
         else if (duty < 0){ duty = 0;}

         SetDCPWM1(duty);

         // in case the car gets off the track, it will disable the motors.
         while(Left_Sensor()<370 && Left_Sensor()<370){
             PORTB = 0x01;
         }
     }
 }
```