University of Southern Queensland

Faculty of Engineering and Surveying

Balancing Wheeled Robot

A dissertation submitted by

Ho, Khoon Chye Randal

in fulfillment of the requirements of

Courses ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Electrical and Electronics)

Submitted: October, 2005

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Ho, Khoon Chye Randal

Student Number: 0050027382

____

Signature

____

Date

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 & ENG4112 Research Project

Limitations of Use

Acknowledgements

I would like to express sincere thanks to my USQ project supervisor Mr. Mark Phythian local supervisor for their guidance and support throughout the duration of the balancing wheeled robot project. I have learned a lot under their guidance, be it practically or theoretically.

In addition to that, I would also like to thank lab supervisor Mr. Voon who has help me to arrange the suitable lab hours to use during the course of the project. He is indeed a nice person as no matter how busy he would offer some time to give some invaluable advice on certain issues.

Other than that I am also grateful to my friends who have help and guide me in searching for important notes and books.

Finally, I would like to thank my parents for their moral support as I can count on them whenever I am down and upset.

Abstract

Inverted pendulum has long been the interest of control engineers. The system is unique in such a way that defies the logic of human brain. Moreover, the system inherited the non-linearity attributes, which is part of every systems on earth.

The concept of two wheeled balancing robot is based on the inverted pendulum theory. A suitable control system is needed to control the system so that it is balanced and stable. The main purpose of this project is to use a good control strategy to keep the body of the robot upright.

This dissertation applies the idea of non-linear control strategy and analyses its effectiveness. The non-linear control strategy requires a good understanding of the inverted pendulum system. The knowledge is then implemented in programming to program the microcontroller. This is particularly important in low-level assembly language, which is used in this project.

Two types of semiconductor sensors to provide tilt information to the robot to balance are applied. The sensors used are gyroscopes and accelerometer. The performance of the sensors is then evaluated based on stability.

In addition to that, it is important to design a reliable motor driver to control the speed and direction of the motors, which is central to the balancing process. The method used is the h-bridge circuit. Since there are two different motors are controlled separately, it is best to control the speed so that the motors work in harmony with each other, thus balancing the robot better.

Nomenclature

The following are the variables being used in modeling the balancing robot.

x Linear displacement

$\dot{x}$ Linear velocity

$\ddot{x}$ Linear acceleration

θ_{rc} Rotation angle of robot chassis

$\dot{\theta}_{rc}$ Angular velocity of the robot chassis

$\ddot{\theta}_{rc}$ Angular acceleration of robot chassis

$\dot{\theta}_{wh}$ Angular velocity of the wheel

$\ddot{\theta}_{wh}$ Angular acceleration of the wheel

C_l Applied torque from motor to left wheel

C_r Applied torque from motor to right wheel

P_l, P_r, H_l, H_r Reaction forces between the wheel and chassis

H_{fr}, H_{lr} Friction forces between the wheels and the ground

g Gravitational acceleration, $9.81 m s^{-2}$

J_{rc} Moment of inertia of robot chassis

J_{wh} Moment of inertia of wheelw

M_{wh} Mass of the robot wheels

M_{rc} Mass of robot chassis

l Distance between the centre of the wheels and robot centre of gravity

r Radius of the wheel

Table of contents

Chapter 3: Hardware

Chapter 4: Sensors

Chapter 5: Stability Analysis

Chapter 6: Motor Control

List of Figures

List of Tables

Chapter 1: Literature Review

This section provides an insight and literature review to the current technology available to construct a two-wheel self balancing robot. It also highlights various methods used by researches on this topic.

1.0 Balancing robots

The concept of balancing robot is based on the inverted pendulum model. This model has been widely used by researches around the world in controlling a system not only in designing wheeled robot but other types of robot as well such as legged robots.

Researches at the Industrial Electronics Laboratory at the Swiss Federal Institute of Technology have built a prototype two wheel robot in which the control is based on a Digital Signal Processor. A linear state space controller using information from a gyroscope and motor encoder sensors is being implemented to make this system stabilise. (Grasser et al.2002).

Another two wheeled robot called 'SEGWAY HT' is available commercially (Dean Kamen ,2001) . It is invented by Dean Kamen who has design more than 150 systems which includes climate control systems and helicopter design. An extra feature this robot has is that it is able to balance while a user is standing on top of and navigate the terrain with it. However, this uses five gyroscopes and a few other tilt sensors to keep it balanced.

Next is the small scale robot, Nbot which is similar to JOE is built by David. P Anderson. (Anderson, David.P) This robot uses a commercially available inertial sensor and position information from motor encoder to balance the system. This robot has won the NASA cool robot of the week in the year 2003.

Steven Hassenplug used a more innovative approach to construct a balancing robot (Steve Hassenplug, 2002). The chassis of the body is constructed by using the LEGO Mindstorms robotics kit. The balancing method of controlling the system is unique with two Electro-Optical Proximity Detector sensors is used to provide the tilt angle information for the controller. This omits the conventional use of gyroscope that has been used by previous robot researchers.

1.1 Control System

Over the years there are only two types of control being used by researchers in controlling a system. The types of control is categorised as linear and non-linear control.

In some instances, the linear control is sufficient to control a system. One of the most widely used is the Proportional Derivative Integral controller or better known as the PID controller (Rick Bickle, 2003). The others are linear quadratic controller (LQR), fuzzy logic controller, pole placement controller etc. It is generally accepted that linear control is more popular than non-linear control.

There are two reasons for this. In all cases, the modeling of a system requires a lot of parameters to be considered and applied. Therefore, the system is complex. However, some of the parameters values needed to model the system are small. That is why most researchers would prefer to model their applications in a linear approximation, which is simpler and in some instances effective.

However, in most cases the linear control theory is not suitable for real life implementation, which mostly exhibit non-linear response. For better performance some non-linear approximation can be applied. (J.J. D'Azzo, pg 11)

Figure below shows how a non-linear response can be approximated to a linear response.

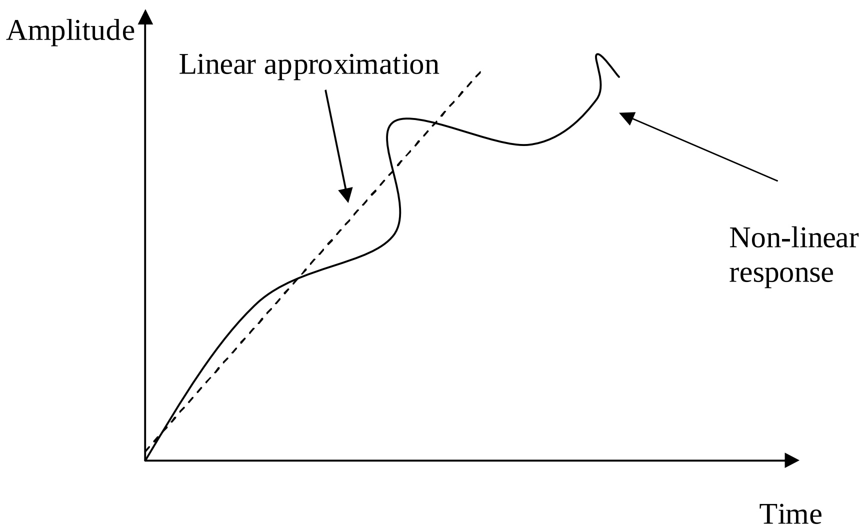


Figure 1: Difference between linear and non-linear response

However, it is better to use the non-linear control theory to control the balancing robot system. This is because the system of the balancing robot, which is based on the inverted pendulum concept, is unique and unbalanced in nature (‘Inverted Pendulum’, Microrobot NA) . Due to this the response is unpredicted.

1.2 Data Acquisition

In a paper ‘Attitude Estimation Using Low Cost Accelerometer and gyroscope’ presented by Young Soo Suh, it shows the two different sensors which is the accelerometer and gyroscope that exhibits poor results when use separately to determine the attitude which is referred as the pitch angle or roll angle. The factor that contributes to the deviation of the desired result of the gyroscope is due to the drift term. Since the drift increases with time error in output data will also increase.

One of the disadvantages of using accelerometer individually is that the device is sensitive to vibration since vibration contains lot of acceleration components. One solution that Young suggested is that a low pass filter is required to limit the high frequency.

However, the gyroscope can combine with accelerometer to determine the pitch or roll angle with much better result with the use of Kalman filter.

1.3 Kalman Filter

The purpose of this filter is to solve problems of statistical nature (Kalman, 1960). Kalman filter is applied based on several mathematical equations that provides computational solution by using the least squares method. In other words, in simple explanation the process is done by averaging a sequence of values. This filter is powerful as it can estimate the past, present and future states. This filter is actually applied in state space equation. Since the program to be used is in assembly this method is not to be used. The averaging method is used instead.

Chapter 2: Introduction

2.0 Balancing Wheeled Robot Concept

Imagine a 2D diagram of an inverted pendulum on the wheel cart. If say, the pendulum falls to the left, the cart should move to the left to try and keep it upwards. This is the same explanation if the pendulum falls to the other side.

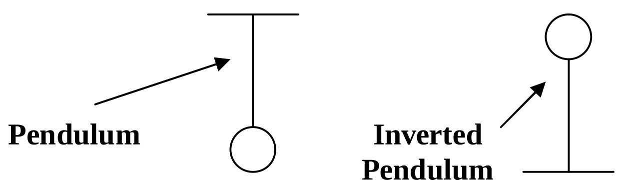


Figure 2: Inverted and non-inverted pendulum

2.1 Balancing Process

The word balance means the inverted pendulum is in equilibrium state, which its position is like standing upright 90 degrees. However, the system itself is not balance, which means it keeps falling off, away from the vertical axis. Therefore, a gyro chip is needed to provide the angle position of the inverted pendulum or robot base and input into the microcontroller, which the program in itself is a balancing algorithm. The microcontroller will then provide a type of feedback signal through PWM control to the H-bridge circuit to turn the motor clockwise or anticlockwise, thus balancing the robot. This can be further explained in the flow chart.

The figures below show the various position of the robot could be in.

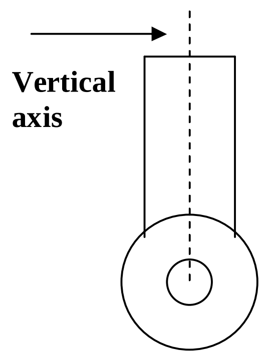


Figure 3: Equilibrium or Balanced State

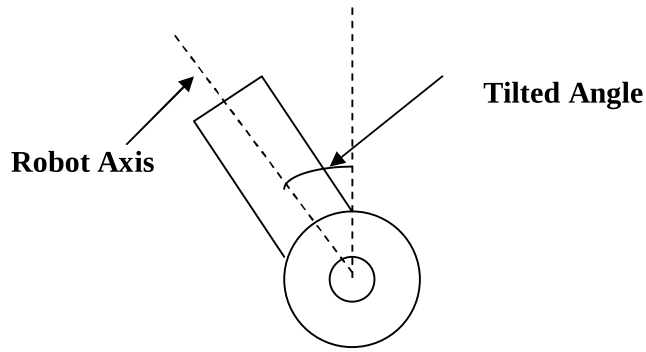


Figure 4: Tilted or Unbalanced State

2.2 System modeling

One of the very first steps for a control system engineer to able to control a system successfully is to understand the system first. An engineer can understand the system better through modeling the system in a free body diagram. In this balancing wheeled robot the modeling is divided into two parts. The first is the wheel model and the second is the chassis model.

After modeling the parameters are combined accordingly in the form of equation. With that the model can be represented by a set of equations in a state space form and controlled using control system theory.

2.2.1 Wheel modeling

Right wheel **Left wheel**

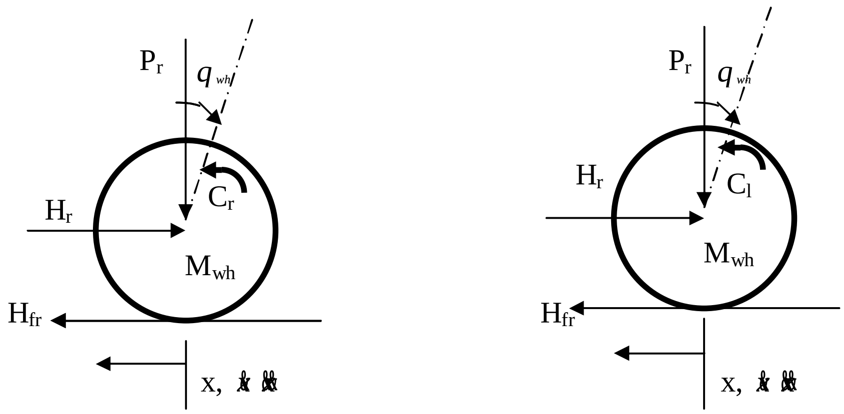


Figure 5: Free body diagram of the wheel

Using the Newton's law of motion, sum of the forces on the x-direction,

$\Sigma F_x = Ma$

$$H_{fr} - H_r = M_{wh}\ddot{x} \quad (10)$$

Summing up the moment around the centre of the wheel,

$\Sigma M = J\alpha$

$$C_r - H_{fr} * r = J_{wh}\ddot{\theta}_{wh} \tag{11}$$

Rearranging equation (11),

$$H_{fr} = \frac{C_r - J_{wh}\ddot{\theta}_{wh}}{r} \tag{15}$$

For the right wheel substitute equation (11) into (10),

$$M_{wh}\ddot{x} = \frac{C_r - J_{wh}\ddot{\theta}_{wh}}{r} - H_r \tag{16}$$

For the left wheel,

$$M_{wh}\ddot{x} = \frac{C_l - J_{wh}\ddot{\theta}_{wh}}{r} - H_l \tag{17}$$

Because the linear motion is acting on the centre of the wheel, angular motion can be transformed into linear motion by simple transformation.

$$\ddot{\theta}_{wh} r = \ddot{x}, \quad \ddot{\theta}_{wh} = \frac{\ddot{x}}{r} \tag{18}$$

$$\dot{\theta}_{wh} r = \dot{x}, \quad \dot{\theta}_{wh} = \frac{\dot{x}}{r} \tag{19}$$

For the right wheel,

$$M_{wh}\ddot{x} = \frac{C_r}{r} - \frac{J_{wh}}{r^2}\ddot{x} - H_r \tag{20}$$

For the left wheel,

$$M_{wh}\ddot{x} = \frac{C_l}{r} - \frac{J_{wh}}{r^2}\ddot{x} - H_l \qquad (21)$$

Summing up equation (20) and (21),

$$2(M_{wh} + \frac{J_{wh}}{r^2})\ddot{x} = \frac{C_r}{r} + \frac{C_l}{r} - (H_l + H_r) \qquad (22)$$

2.2.2 Chassis Modeling

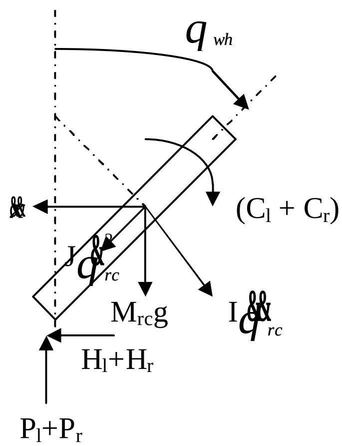


Figure 6: Inverted Pendulum Free body diagram

Sum of forces perpendicular to the chassis or pendulum

$$\Sigma F_{perpendicular} = M_{rc}\ddot{x}\cos\theta_{rc}$$

$$(P_l+P_r)\sin\theta_{rc} + (H_l+H_r)\cos\theta_{rc} - M_{rc}\,l\,\ddot{\theta}_{rc} - M_{rc}g\sin\theta_{rc} = M_{rc}\ddot{x}\cos\theta_{rc} \qquad (23)$$

Sum of forces in the horizontal direction

$$\Sigma F_{horizontal} = M_{rc}\ddot{x}$$

$$(H_l+H_r) + M_{rc}\, l\, \dot{\theta}_{rc}^2 \sin\theta_{rc} - M_{rc}\, l\, \ddot{\theta}_{rc} \cos\theta_{rc} = M_{rc}\ddot{x} \qquad (24)$$

Sum of moments around the centre mass of the pendulum

$$\sum M = J\alpha \qquad \text{, where } \alpha = \ddot{\theta}_{rc}$$

$$-(P_l+P_r)\, l \sin\theta_{rc} - (H_l+H_r)\, l \cos\theta_{rc} - (C_l+C_r) = J_{rc}\ddot{\theta}_{rc} \qquad (25)$$

Rearranging equation 25,

$$-(P_l+P_r)\, l \sin\theta_{rc} - (H_l+H_r)\, l \cos\theta_{rc} = J_{rc}\ddot{\theta}_{rc} + (C_l+C_r) \qquad (27)$$

Multiplying equation (23) by – ell,

$$[-(P_l+P_r)\sin\theta_{rc} - (H_l+H_r)\cos\theta_{rc}]\, l + M_{rc}g\, l \sin\theta_{rc} + M_{rc}\, l^2 \ddot{\theta}_{rc} = -M_{rc}\, l\, \ddot{x} \cos\theta_{rc} \qquad (28)$$

Substitute equation (27) into equation (28) to eliminate (P_l+P_r) and (H_l+H_r) term and rearranging the same terms,

$$J_{rc}\ddot{\theta}_{rc} + (C_l+C_r) + M_{rc}g\, l \sin\theta_{rc} + M_{rc}\, l^2 \ddot{\theta}_{rc} = -M_{rc}\, l\, \ddot{x}\cos\theta_{rc} \qquad (29)$$

To eliminate (H_l+H_r) term for the second equation, equation (24) is inserted into Equation (22) and rearranging the same terms,

$$\left(2M_{wh} + \frac{2J_{wh}}{r^2} + M_{rc}\right)\ddot{x} = \frac{C_r}{r} + \frac{C_l}{r} + M_{rc}\, l\, \dot{\theta}_{rc}^2 \sin\theta_{rc} - M_{rc}\, l\, \ddot{\theta}_{rc}\cos\theta_{rc} \qquad (30)$$

Equation (29) and (30) are non-linear equations, and to get linear equations, some linearising or assumption are taken. Let $\theta_{rc} = \pi + \delta$, where δ is the small angle from the vertical upward direction. Therefore, $\cos\theta_{rc} = -1$, $\sin\theta_{rc} = -\delta$ and $\frac{d^2\theta_{rc}}{dt^2} = 0$.

The linearised equation representing the whole system is,

$$J_{rc}\ddot{\delta}_{rc} + (C_l+C_r) - M_{rc} g l \,\delta + M_{rc} l^2 \ddot{\delta}_{rc} = M_{rc} l \,\ddot{x} \tag{31}$$

$$\left(2M_{wh} + \frac{2J_{wh}}{r^2} + M_{rc}\right)\ddot{x} = \frac{C_r}{r} + \frac{C_l}{r} + M_{rc} l \,\ddot{\delta}_{rc} \tag{32}$$

Rearranging the state space representation,

$$\ddot{\delta}_{rc} = \frac{M_{rc} g l}{J_{rc} + M_{rc} l^2}\delta + \frac{M_{rc} l}{J_{rc} + M_{rc} l^2}\ddot{x} - \frac{(C_l + C_r)}{J_{rc} + M_{rc} l^2} \tag{33}$$

$$\ddot{x} = \frac{M_{rc} l}{\left(2M_{wh} + M_{rc} + \frac{2J_{wh}}{r^2}\right)}\ddot{\delta}_{rc} + \frac{C_l + C_r}{r\left(2M_{wh} + M_{rc} + \frac{2J_{wh}}{r^2}\right)} \tag{34}$$

The state space equation is obtained as below after substituting equation (33) into (32) and equation (34) into (31) in the form of:

$$\frac{dz}{dt} = A\bar{z} + Bu$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{d} \\ \ddot{d} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{M_{rc}^{2} g l^{2}}{j - \dfrac{M_{rc}^{2} l^{2}}{J_{rc} + M_{rc} l^{2}}} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{M_{rc}^{2} g l^{2}}{J_{rc} - \dfrac{M_{rc}^{2} l^{2}}{j} + M_{rc} l^{2}} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ d \\ \dot{d} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \dfrac{1}{r} - \dfrac{M_{rc} l}{J_{rc} + M_{rc} l^{2}} & \dfrac{1}{r} - \dfrac{M_{rc} l}{J_{rc} + M_{rc} l^{2}} \\ 0 & 0 \\ \dfrac{M_{rc} l}{rj} - 1 & \dfrac{M_{rc} l}{rj} - 1 \end{bmatrix} \begin{bmatrix} C_r \\ C_l \end{bmatrix}$$

(35)

where $j = 2M_{wh} + \dfrac{2J_{wh}}{r^{2}} + M_{rc}$

and output of $y = C\bar{z}$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ d \\ \dot{d} \end{bmatrix}$$

(36)

The C matrix is oriented this way because the position of the wheel base and the robot chassis are interested.

Applying the feedback equation:

$$u = r - [e \quad f \quad g \quad h]\begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \tag{37}$$

Whereby x1, x2, x3 and x4 are the states of the state space equation

The feedback equation is the equation that is about to be inserted into the main system, with coefficients e,f,g, and h are the arbitrary chosen values to be experimented (by trial and error) in order to get the robot to balance properly.

Finally the overall balancing robot control system is shown in the figure below.

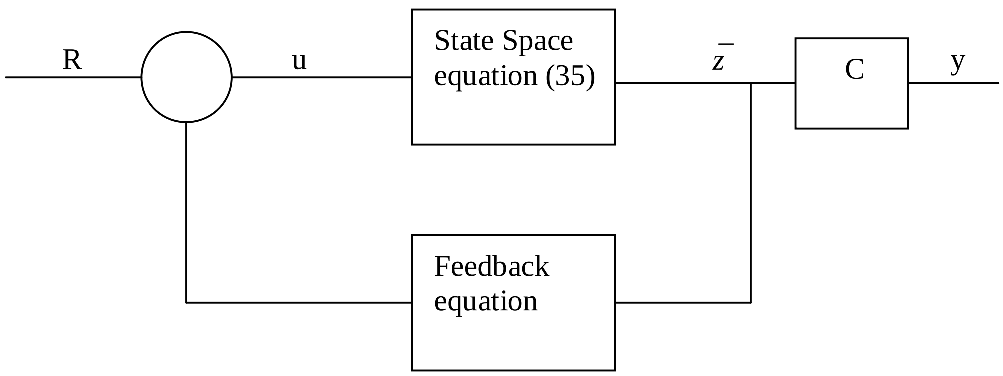


Figure 7: Balancing robot control system

The term or variable u represents the driving speed of the motor to lift the robot chassis upright. In order to drive the robot chassis upright the motor need to have the required minimum speed to do the task.

The non-linear state space equation is difficult to derive as there are many variables that has to be taken care of. However, the equations (29) and (30) can be represented with the state space equation

2.3 Approach

The balancing robot can be controlled using the state space equation with high level language programming such as C. However, after gaining much understanding of the robot system it is logically possible that the balancing robot can be controlled by using the assembly language programming, even though it is quite simplistic as compared to the state space equation previously derived in the modeling section.

The following strategy is applied in the control of balancing robot.

a) Data acquisition time

The time the microcontroller takes to collect and execute the data obtained. Theoretically if the microcontroller can acquire data and applying control strategy faster than the response of the robot chassis tilts the robot will appear to be balanced and more stable.

b) The non-linearity control

Applying a control strategy limits the overshoot. For instance, there is no accurate prediction that the motor would not output more speed when in fact the speed required to control (programmed) is lesser. That is why to be on the safe side a few lines of control to limit that possible non-linearity behavior.

c) Proportional band control

This is based on the requirement of the robot from the motor to appropriately control the tilt of the robot. For instance, when the robot tilts further the more speed is required to lift the chassis and vice versa.

Chapter 3: Hardware

3.0 Motor Selection

There are basically two types of motor that are in consideration initially. There are a few reasons why the car wiper motor, which is one of the permanent magnet DC motor, is chosen over stepper motor.

The first is that the stepper motor does not turn on the shaft fast enough. That is the speed of response is slow. This could be detrimental to the balancing robot as the speed that the robot tilts is quite fast and need a motor that could match or have faster response than the robot chassis to lift the chassis to the upright position.

Secondly, the rating speed (in r.p.m) of the stepper motor is not as high as the permanent magnet DC motor. This aspect is also essential in the balancing robot project as heavier robot needs more motor speed from the motor to lift the chassis balanced state.

3.1 Speed Calculation

This section focuses on the how to calculate the required speed of the motor to lift the chassis of the balancing robot upright. The rated speed of the motor has to be sufficiently high in order to balance the weight of the robot. If the weight of the robot is more than the motor can handle the robot will not be able to standstill.

Acceleration needed = $5ms^{-2}$

Weight of the wheels and motors = 2kg

Force to accelerate the 2kg is F=ma, F= 2*5ms^{-1}
= 10N (5 N per wheel)

Wheel radius = 0.045m

Power required from motor would be P = v * F
= 2 *10
= 20W , that is two 20W motors

Assuming the motors will on average operate at ½ their rated voltage, this leads to a factor of 4 reductions in the power output of the motor (from its rated power).

$$P = V^2/R$$
$$= (1/2V)^2/R$$
$$= 1/4V^2/R$$

So that is 20W (needed) * 4 = 80W motors.

Given some tolerance of 5% of the required power of the motor; therefore a motor power rating of about 84W is required.

Due to the fact that the angular rate of turn is quite high and in order to make it controllable, the speed should be geared down by a ratio of three 3 (1:3). The speed must not be geared down significantly as it will increase the moment of inertia in the drive mechanism and slows its responsiveness (John Billingsley, 2005).

3.2 Motor capacity

After finalising the required speed of a motor needed the next step would be confirming the motor capacity of a motor practically. The information can be obtained by reading the angular rate or speed and power from the motor's datasheet. Since the car wiper motors that obtained is from the spare parts shop another method is needed to get the information. This can be by using a stroboscope/tachometer to measure the angular rate.

3.2.1 Speed measurement

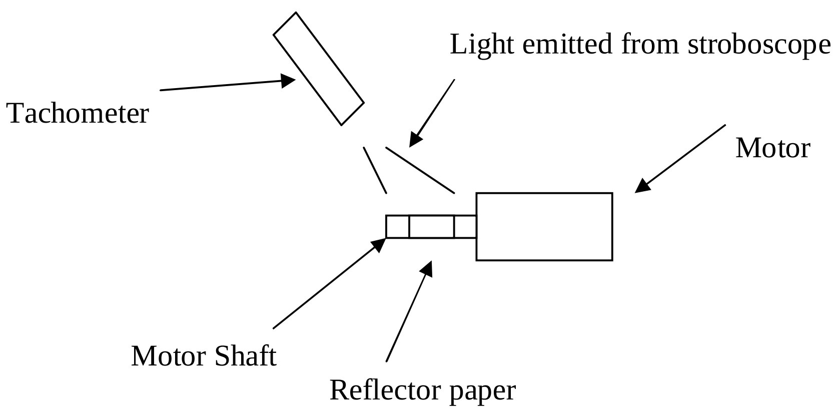


Figure 8: Speed measurement using tachometer/stroboscope

The tachometer is positioned as above and a reflector paper is attached to the motor shaft. As the motor shaft turns the tachometer light output is emitted onto the reflector paper, when then reflect the light back to the tachometer to provide a reading. The reading is the angular rate which is in unit r.p.m.

3.2.2 Results

Determining the characteristics of the wiper motor:

Motor O (Side wire)

Voltage, V	Current , A	Angular speed, rpm
12V	1.3A	3100 rpm
10V	1.26A	2500 rpm
8V	1.05A	1900 rpm

Motor O (Centre wire)

Voltage, V	Current , A	Angular speed, rpm
12V	0.7A	2100 rpm
10V	0.68A	1780 rpm
8V	0.6A	1360 rpm

Table 1 : Motor O speed characteristics

Motor X (Side wire)

Voltage, V	Current , A	Angular speed, rpm
12V	1.8A	3000 rpm
10V	1.7A	2300 rpm
8V	1.6A	1770 rpm

Motor X (Center wire)

Voltage, V	Current , A	Angular speed, rpm
12V	1.22A	2085 rpm
10V	1.2A	1700 rpm
8V	1.12A	1300 rpm

Table 2 : Motor X speed characteristics

The speed is measured by using stroboscope/ tachometer and as noted from the experiment, the side wire connection makes the motor turn faster. For general information, wiper motor is a two speed motor whereby user can select whether the faster or the slower speed.

3.4 Robot Chassis Design

3.4.1 Robot Weight

It is important to consider the overall weight of the robot that one is about to build. This would seriously affect the wheel base design of the robot. The base and the wheel might bend outwards if the overall weight of components is too high for the wheel base to handle.

The drop down list of the total weight contributed:

	Quantity	kg/quantity	Total weight (kg)	
Sealed lead acid battery	2	1	2	
Aluminum	-	0.5	0.5	
Wiper motors	2	0.5	2	
Printed Circuit Boards	4	0.02	0.08	Total = 4.58kg

From the total weight anticipated the robot requires a strong wheel base.

3.4.2 Materials

Since the aluminum is strong, light and affordable it is preferably chosen as the material to build the robot base and chassis.

3.4.3 Robot Response

The robot falls faster when the robot is shorter. Therefore, a simple experiment can be done by first holding the robot up straight, release and measure the time on how much time it needed to tilt. The taller robot will be easier to control as the responds is slower.

3.4.4 Gears Selection

The tooth gear pulley is chosen ahead of the sprocket type. This is because the tooth gear pulley is more durable and the quietest choice. The tooth belt pulley consists of two aluminum gears and a suitable length of belt. This method is easy to use and do not require special skill to mount on the robot.

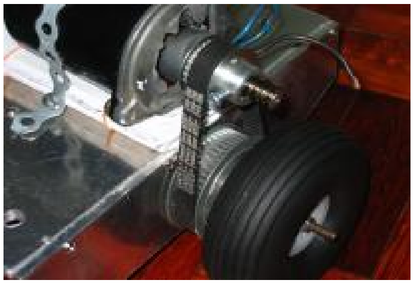


Figure 9: Aluminum Gear mounting

3.4.5 Motor Mounting

Since the wiper motor is quite heavy and a tooth gear pulley is required, it is best that the motor is mounted on top of the robot wheel base. In addition to that ball bearing is embedded into the aluminum side plate in an axis parallel to the motor shaft. This is to reduce the amount of friction imposed on the shaft and this enable the shaft to turn smoothly.

Figure 10: Motor Mounting

3.4.6 Aluminum rod fabrication

This step is taken to ensure the portability of the robot. The top and middle rods can be disconnected anytime the user wants.

Figure 11: Aluminum rod fabrication and mounting

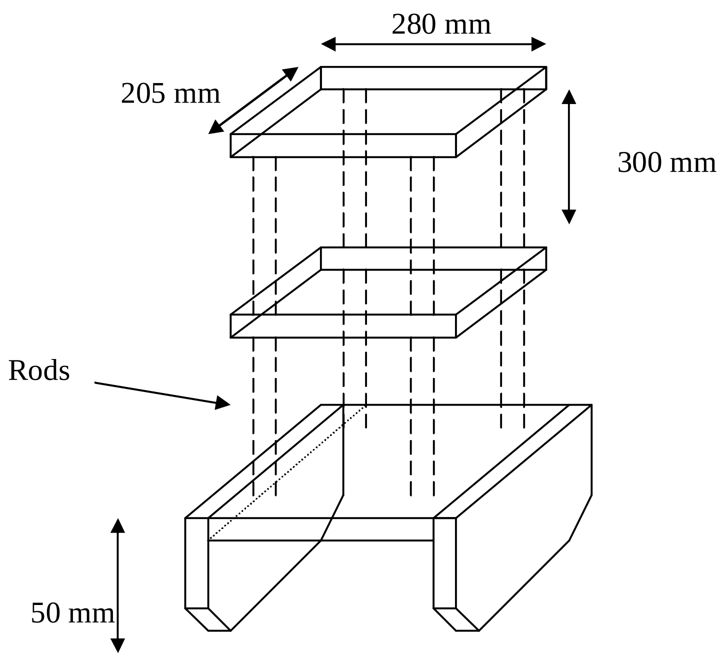
3.4.7 Rigidity

This type of tightening is to ensure that the robot do not wobble while balancing.

Figure 12: Nut tightening

3.4.8 Robot Chassis construction

The chassis of the robot will be constructed of the following designs:



The mounting of the motor to the wheel

Figure 13: Robot Chassis

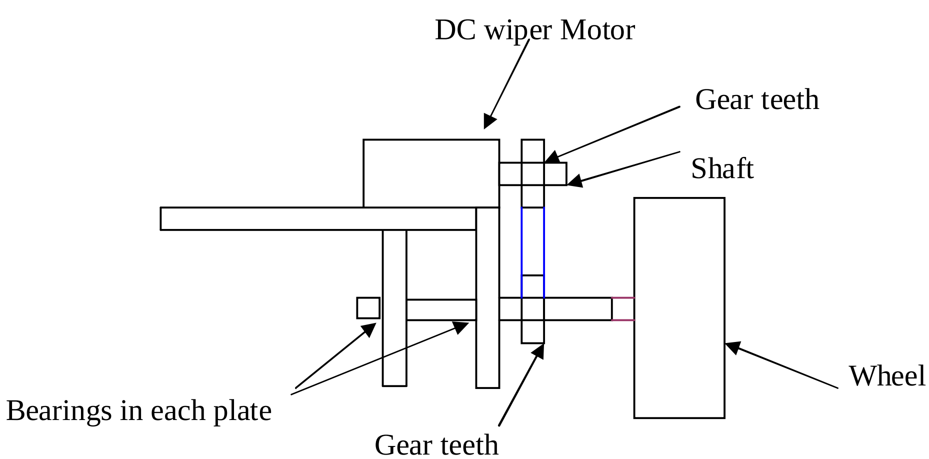


Figure 14: Cut away view of robot base mounting

Chapter 4: Sensors

The feedback block of the balancing robot control system consists of sensors that provide information for the robot to move accordingly. There are two types of sensor used in this project, which are the gyro and the accelerometer.

4.1 ADXRS 300 Gyro

A person might be curious on how the ADXRS gyro measure data to provide important information to the balancing wheeled robot. It is a general statement that gyro measures angular rate of turn data. ADXRS gyro obtains this data by measuring the coriolis acceleration.

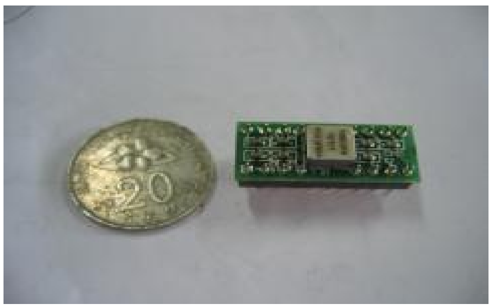


Figure 15: ADXRS 300 Gyro

4.2 Coriolis acceleration

Imagine a person standing on a rotating platform near to the center trying to maintain his position to the ground by walking against the rotation at a given speed. Whereas if a person were to maintain his position at the outer rotating platform (away from the center) he has to increase his movement speed. The increase in speed is termed coriollis acceleration (John Green et al, March 2003).

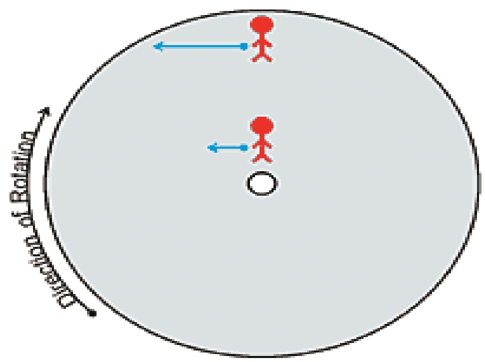


Figure 16: Coriolis acceleration on earth
(Source: John Green et al, March 2003)

4.3 Coriolis acceleration on gyro

The ADXRS gyro applies the similar effect of the above coriolis acceleration concept. Except that the human is replaced by a small block of mass and forming a damping system. The damping system is designed such that it oscillates in one direction when the platform is rotating. When the damping system moves to the outer surface, it experiences a force to the left (John Green et al, March 2003)

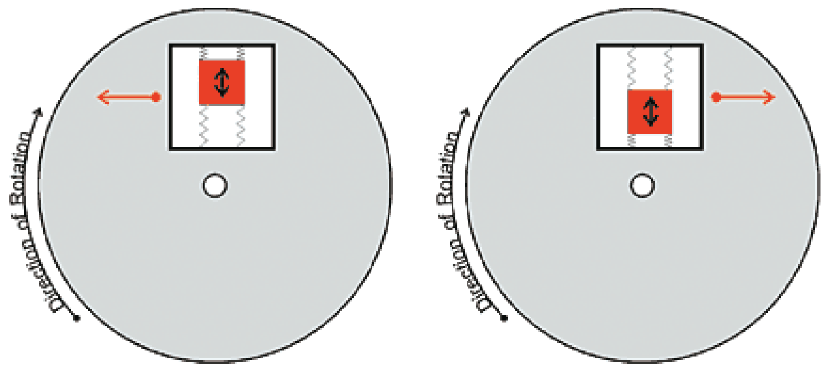


Figure 17: Gyro rotation behaviour
(Source: John Green et al, March 2003)

4.4 Measuring coriolis acceleration

The figure below shows the cut off structure of the ADXRS 300 gyro. The structure frame containing the mass is tethered to the substrate perpendicular to the resonating motion (John Green et al, March 2003). The ADXRS primarily utilises the coriolis sense fingers to 'read' the amount of force applied by the mass to it, which in turn 'translates' the force into the voltage reading.

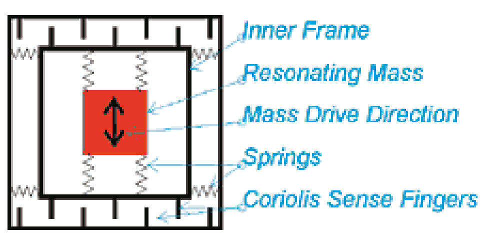


Figure 18: ADXRS inner structure
(Source: John Green et al, March 2003)

Figure below is the complete structure of the ADXRS gyro placed on the moving plate.

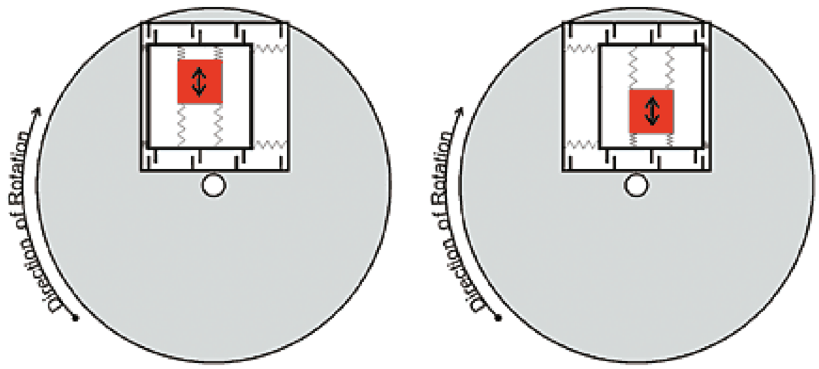


Figure 19: Complete inner gyro structure
(Source: John Green et al, March 2003)

4.5 Advantages

There are quite a number of reasons why this gyro sensor is chosen over the other sensors such as tilt meters and gyro for helicopters. The reasons are as follows:

i) Small in size

There is no need to worry about space. Since the gyro is so small that is only about the size of a coin the user can place the gyro on a small empty space on the robot.

ii) Samples available for free (from www.analogdevices.com)

This is great for any researchers that are keen on researching but short of funding. Analog devices provide this golden opportunity for anyone to try hands-on.

iii) Has internal conditioning on-chip

This special feature improves the data Most of the information is available on the data sheet as attached on the appendix C.

iv) Readily available for interfacing with microcontroller chip

There is no need to include extra circuits to interface the ADXRS 300 gyro with microcontroller chip as the signals are compatible with each other.

4.6 Axis selection

There are a few axes that can be used. There are named Yaw Axis, Roll Axis and Pitch Axis as shown in figure (A) below. In the balancing robot application only one axis is used. This can be the roll axis in figure (B).

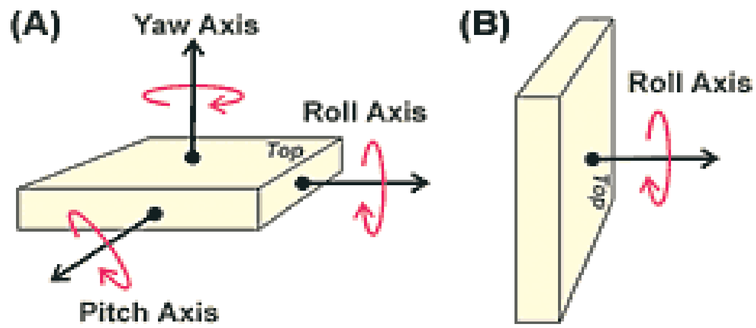


Figure 20: ADXRS 300 Gyro Axes
(Source: John Green et al, March 2003)

4.7 Mounting

Due to the fact that measurement of coriolis acceleration is most sensitive when mounted parallel to the ground surfaced the gyro has to be mounted on like the one in the figure(B) above onto the robot. The mounting can be as shown below:

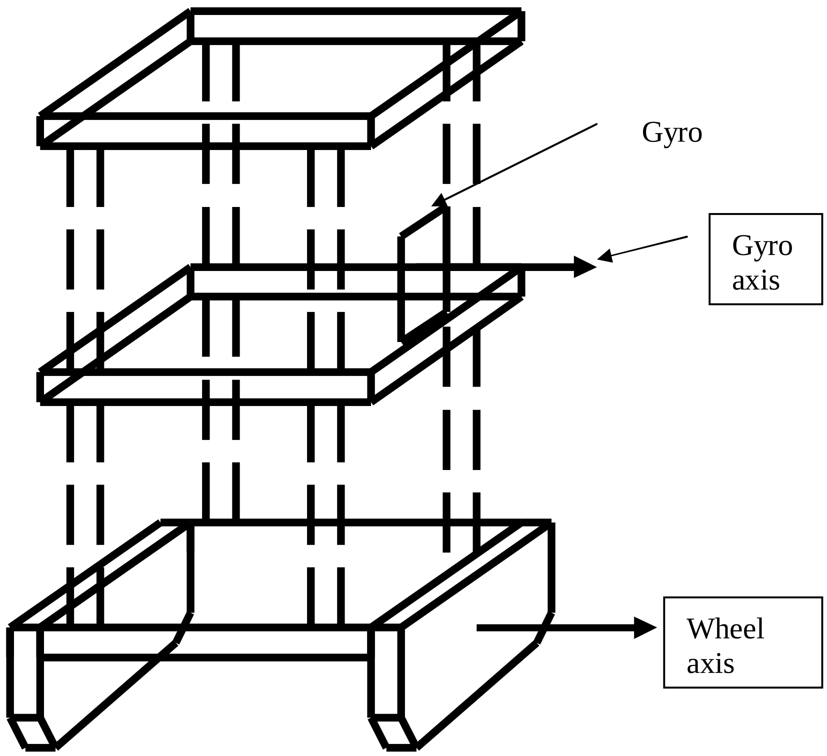


Figure 21: Gyro mounting

4.8 ADXRS Gyro as tilt meter

Based on the graph given below the range of values from 2.5 to 4.75 volts can be assigned for clockwise direction tilt and from 2.5 to 0.25 volts the values can be allocated for anti-clockwise direction. For instance, when the microcontroller receive an input on 4.0 volts the chassis will tilt clockwise direction and motor will turn anticlockwise to lift the chassis back up and vice versa.

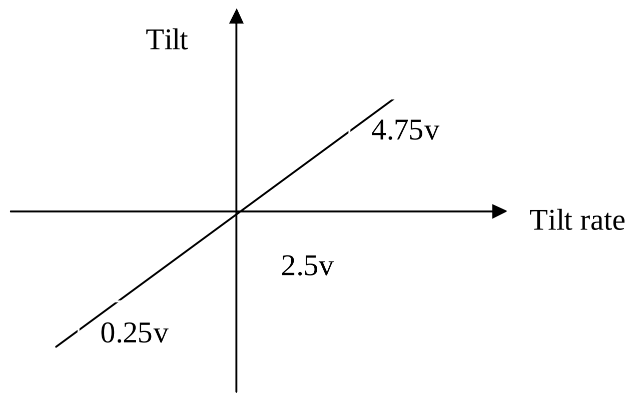


Figure 22: Gyro characteristic curve

4.9 Measurement Range

The ideal measurement range of the graph above does not apply to the practical measurement range. Initially, the measurement range of the gyro is only within 2.3 to 3 volts. Therefore, the method of extending the range is by adding a resistor between the pins CMID and SUMJ of the gyro. This can be found in the data sheet as attached in appendix C4. As found out the measurement range is only from 1.8 volts to 4.9 volts. As a result some adjustment has to be made to enable the microcontroller to measure the voltage values.

4.10 ADXL Accelerometer

The purpose of this accelerometer is to be examined whether it can be used as an alternative to find the tilt angle. The second purpose of accelerometer can be considered to use with the gyro chip to stabilize the angular reading drift. This is because it may have an important property, which is the duty cycle to hold the data for a specified period of time before the next sampling edge to calculate and replace the old value.

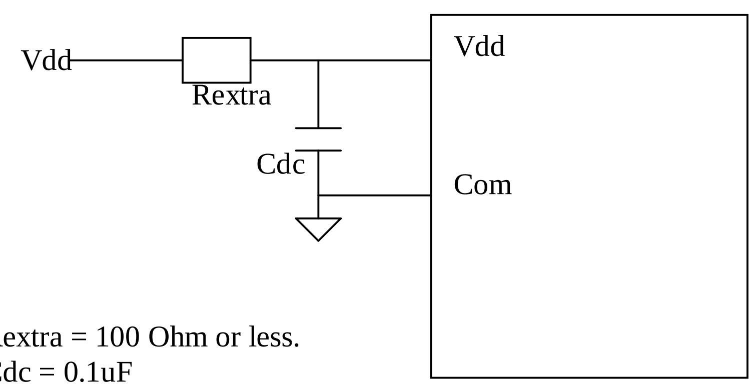
4.11 Calibration

With reference to the data sheet attached at the appendix C3.There are a few electronic components values that needed to be determined before mounting on the printed circuit board designed as attached at the appendix B1.

The primary function of the Rset resistor is to set the PWM period of the accelerometer output, T2. The equation is given in the equation below:

T2 = Rset/125MOhm

4.11.1 Power supply decoupling



Rextra = 100 Ohm or less.
Cdc = 0.1uF

Figure 23: PCB power supply coupling

4.11.2 Capacitors

The capacitors Cx and Cy act as the filtering system to reduce the external noise that might affect the overall performance of the accelerometer.

Choosing the low pass filter bandwidth to be 100Hz

Using the equation: $F_{-3dB}=1/(2p\ (32KOhm)*C(x,y))$

Therefore, the capacitance value would be = 0.05uF

4.11.3 Calculation

The calibration of ADXL 213 is done by utilizing the earth's gravity as a reference input. The X and Y axes are properly aligned horizontal to the earth such that both axes experience 0g. A microcontroller is told to read the duty cycle output T1 and period T2 from each of the axes.

The accuracy can be improved by average the readings of T1 and T2. These values are used as the fixed values to be used in calculating the acceleration after calibration mode.

Zcal = (T1max – T1min)/2

The bit scale factor is to be found next. This is to determine the resolution (in bits) of the acceleration calculation as shown below:

K= (T2*bit scale factor)/(T1max-T1min)

4.11.4 Software Approach

T2actual is the measurement of T2. This formula offsets the 0g value for changes in T2 due to drift. The steps below are actually the method that is used to sample the edges of the output pulse of the accelerometer output.

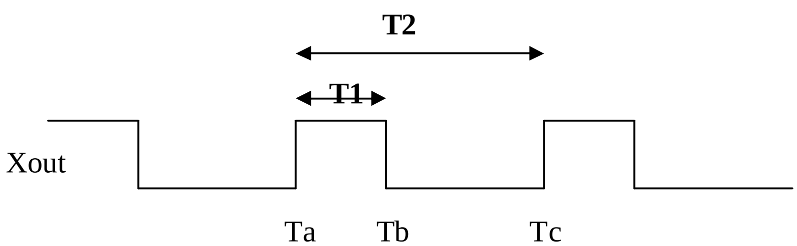
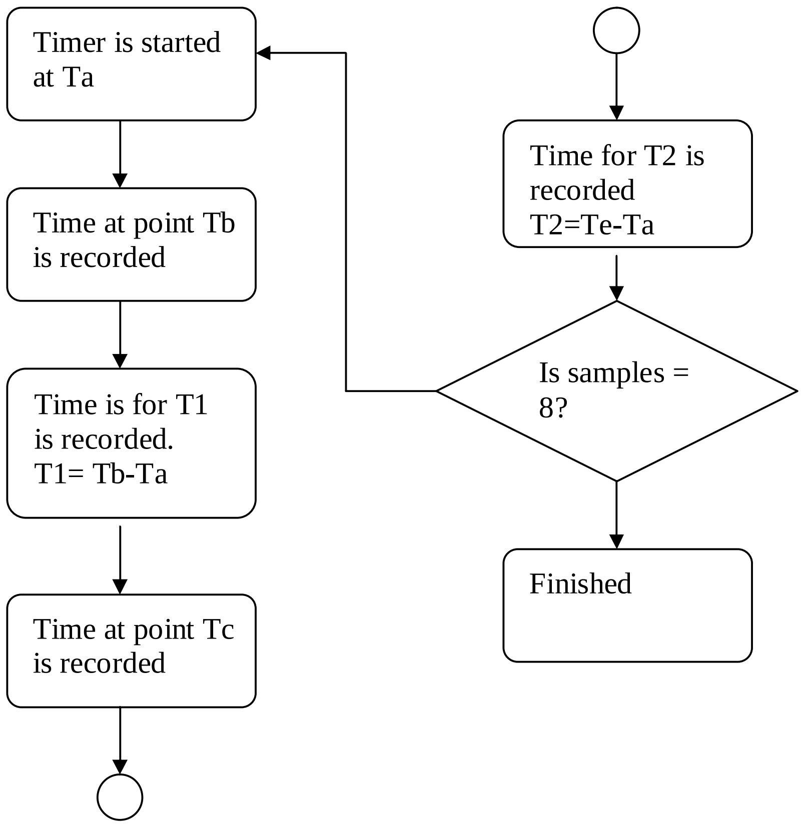


Figure 24: Duty cycle decoding scheme

The flow chart below illustrates the sampling of the output edges



As can be seen above only the output pulse at X pin is being sampled. This is because one axis is sufficient in this application. After sampling the time values the Acceleration and tilt value of axis can be found by using the formula below:

Acceleration = K*(T1-Zactual)/T2actual

Zactual = (Zcal * T2actual)/T2cal

Full program listing is attached at appendix D

4.12 Overall Balancing Program

4.12.1 Programming Approach

The ADXRS 300 Gyro output is in the form of analog values. Therefore, the Analog to Digital Converter (ADC) Module of the microcontroller is being utilised.

Since the resultant registers storage is of 10 bits wide and the maximum voltage used is 5V therefore the assigned analog voltage to digital form is as shown in table 3.

One bit value is : (2^10)/5 = 1023/5 = 204.6 or equal to 205

Analog (v)	Multiplier	Digital
1	205	205
1.5	205	308
2	205	410
2.5	205	513
3	205	615

Table 3: Analog to digital conversion

The resultant value is being stored in ADRESH and ADRESL registers.

After that the computer program will calculate the reading error from the reference, which is 3V. Then the direction to control will be determined. The program will appropriately choose the correct subroutine, based on error reading calculated to send the appropriate amount of PWM signal to control the motor. In the way it is programmed there are quite a number of PWM subroutines to choose from. This might sound to be a crude technique. There are two reasons for this. Firstly, it has got to do with the design of the h-bridge, which makes the programming even more complex. That is why the internal PWM function of the PIC microcontroller not used. Otherwise the programming could be done better. The second reason would be the robot chassis does not tilt much, therefore does not need much reading for control.

Chapter 5: Stability Analysis

There are some points need to be taken into consideration. There are a few things that could be done in order to properly balance the robot.

5.1 Differential Drive Motor Control (While Standstill)

The method being used here is to manually calibrate the speed of the two differential drive motors such that the shaft not only turns in one direction but with the same speed as well.

The calibration of the two wiper motors are done prior to mount the gyro and accelerometer onto the robot chassis. This is one important step before balancing the robot.

The robot will be quite impossible to balance and might wobble around.

5.2 Sensor Fusion

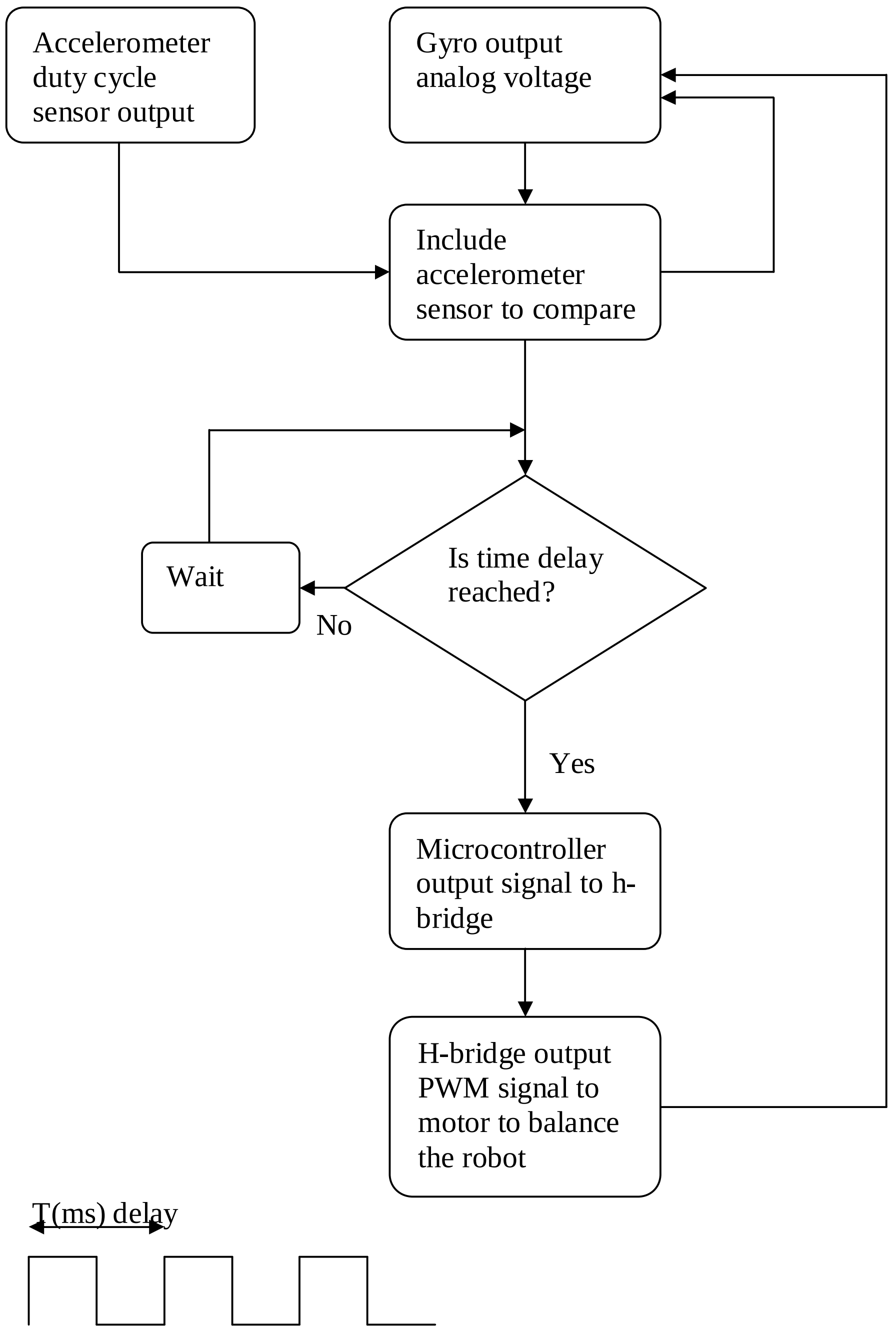
Both the gyro and accelerometer can be used to measure tilt data. However, only the gyro is used to measure the tilt while the accelerometer can be used as a 'stabiliser' to provide stability. There are three important properties that can be utilised from the accelerometer.

First and foremost, the response of the accelerometer sensor is quite slow as compared to the gyro, which only outputs a signal every two cycles (Appendix C, pg). The other reason is that the output of the accelerometer is in pulse width digital form it could be stated that the output of the accelerometer is more stable than the gyro analog output.

The accelerometer provides a true measure of tilt. For example, the accelerometer will output a digital pulse when the device is in steady state condition. Similarly, accelerometer will change its output accordingly when the device rotates. In other words, the accelerometer will only change its output when the device changes direction.

Unlike the accelerometer the gyro is sensitive to drift, which the value of the output changes with respect to time. For instance, when the gyro rotate to a position, the gyro will give a reading, however, the reading will quickly return to steady state value as there is no angular turn.

Therefore, the accelerometer can be combined with the gyro to offset the drift. The strategy is to utilise the delay in accelerometer to 'hold' the gyro data and only output a signal to the motor driver when the accelerometer 'permits' it. The general process can be better understood through the flowchart


Accelerometer duty cycle sensor output
Gyro output analog voltage
Include accelerometer sensor to compare
Is time delay reached?
Wait
No
Yes
Microcontroller output signal to h-bridge
H-bridge output PWM signal to motor to balance the robot
T(ms) delay

5.3 Averaging

Since the gyro outputs data at a high rate the sensor data is quite unstable. In other words, the reading is not linear and it oscillates. That is why the reading needs to be averaged.

This is one method of reducing the non-linearities. For example let the original data output at a rate of 400Hz and if the data is averaged for eight samples. Therefore the output rate would be 400Hz/8 = 50Hz.

5.4 Differential drive motor control (While traveling)

There could still be deviation of the actual trajectory when the robot is traveling, even though initial calibration as in section 5.1. This is analogous to a situation whereby when a person is driving a car with the steering wheel set such that the car moves in a straight direction. The person just sits there without holding the steering wheel while the car is in motion. After some time, the steering will move and the car moves side ways.

Therefore, some control algorithm is needed to sense that if one of the two motors move faster than the other the computer program will slow the faster moving motor down such that both of the speeds will be synchronised.

Chapter 6: Motor Control

6.1 Pulse Width Modulation

PWM output is basically a series of pulses with varying size in pulse width. This PWM signal is output from the h-bridge circuit to control the wiper motor. The difference in pulse length shows the different output of h-bridge circuit controlling the output speed of the motor.

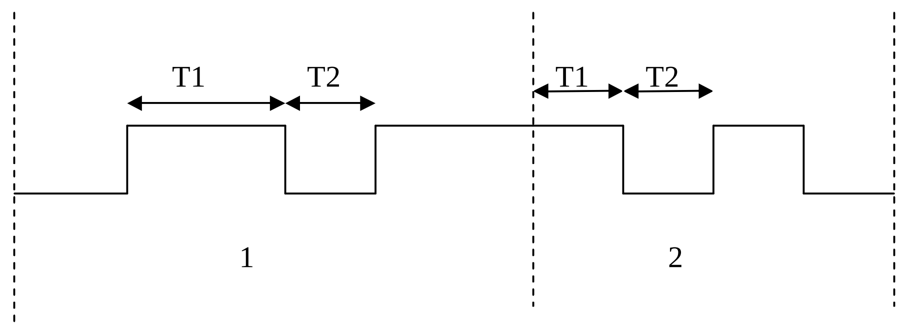


Figure 25: Pulse Width Modulation waveform

Figure above shows the varying pulse length of the pulse width modulation (PWM) scheme. Let's say that the PWM frequency is about 50 Hertz, with a period cycle of 20ms. Therefore assuming that the T1 and T2 length values are 15ms and 5ms respectively, the duty cycle can be calculated as below:

Duty cycle = T1/(T1+T2) * 100%
= 15/20 * 100%
= 75%

Therefore if the maximum rating speed of the wiper motor is about 1000 rpm, then the controlled speed would be 750 rpm.

If suddenly the h-bridge circuit wants to control the half the speed of the motor as in part 2 of figure (25) then the duty cycle value can be calculated as:

Duty cycle = T1/(T1+T2) * 100%
= 10/20 * 100%
= 50% Therefore the speed would be 500 rpm.

6.2 H-bridge design

6.2.1 Introduction

H-bridge circuit is a widely known circuit for controlling the direction spin and speed of DC motors. This is how the H-bridge circuit works. Let's denote one rotation is clockwise and the other direction spins as counter clockwise. Basically, the circuit consists of two p-channel MOSFETS (A & B) and two N-channel (C & D) MOSFETS. In order to turn the motor clockwise, the MOSFETS A and D are turn on while MOSFETS B and C are turn off at one instant. The same goes for counter clockwise direction whereby MOSFETS B and C are turn on while the MOSFETS A and D are turn off. This is best illustrated in the figures below.

Step 1: Motor clockwise turn

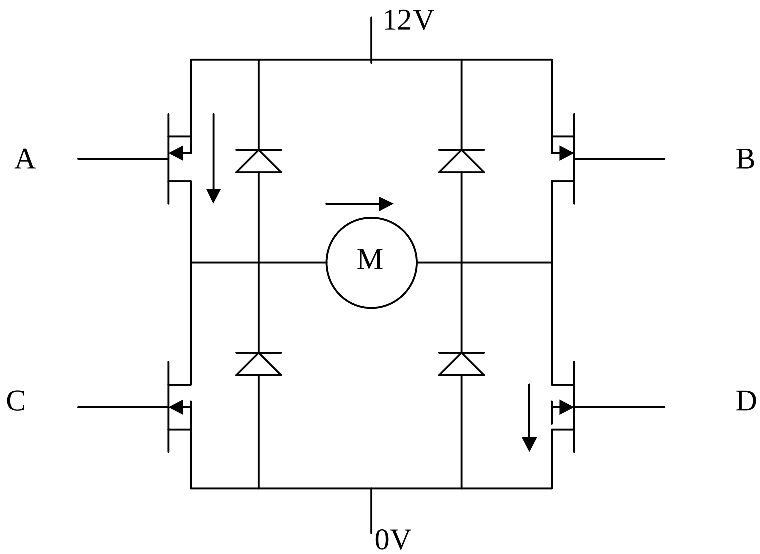


Figure 26: Motor clockwise turn

→ Current flowing clockwise ┈► CEMF current

When the motor starts to turn clockwise, the current will flow as above figure.

Step 2 : CEMF current flow

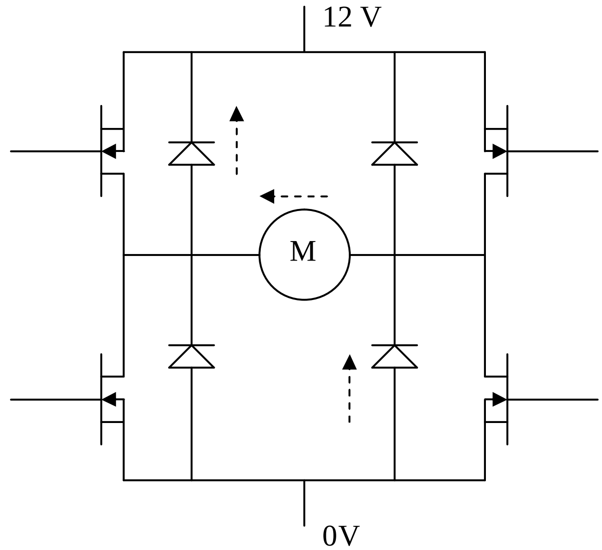


Figure 27: CEMF current flow

If the motor stops and starts to turn the other direction, which is counter-clockwise the opposite CEMF current will flow subsequently as above (Dennis Clark, pg191) . As can be seen the freewheeling diodes serves to protect the MOSFETS from being damage by the CEMF current. That is why there is a tendency that one vertical parallel ‘leg’ shorting when the motor starts to turn the motor the other direction immediately. This is not an ideal situation as it will damage the parallel MOSFETS. Therefore some delay time is needed to allow the CEMF current to finish flowing before starting the counter-clockwise turn as shown in figure 28.

Step 3: Motor counter-clockwise turn

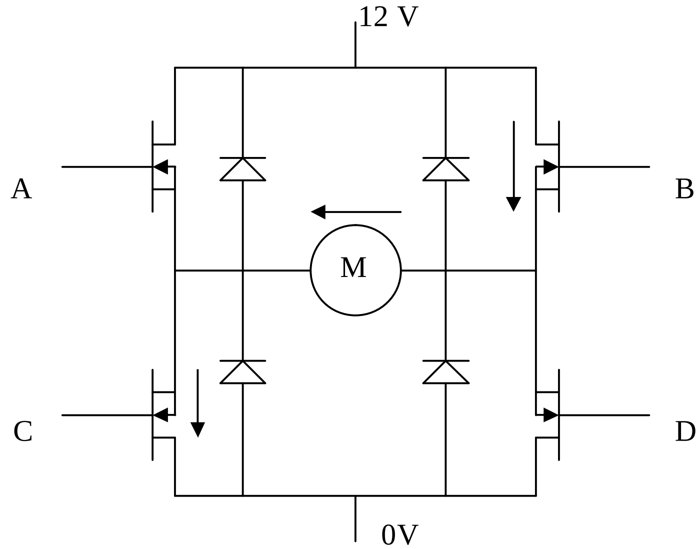


Figure 28: Motor counter clockwise turn

6.3 Design consideration

Design consideration 1: Free-wheeling diodes

Dc motors are very powerful motors and because of that the motors can be term as powerful inductors. Inductors in nature tend to resist change in current. When turning off an inductor, current will gradually go to zero. However, the inductor will try to keep the current flowing. If the current is goes to zero faster, the harder the inductor will try to keep the current. This means current may shoot up higher thus resulting in high voltage across the inductor. The voltage, termed Counter Electro Motive Force (CEMF) is undesirable for the switching transistors which may in the end malfunction.

Since there is CEMF voltage there will also be CEMF current flowing in the transistors. Therefore clamping diodes (4 of them) are inserted as shown in figure. Instead of flowing into the transistors the will flow along the diodes path to ground. The clamping diodes can help limit the amount of high CEMF voltages to a low and desirable 0.3 to 0.7 volts.

Design consideration 2: Transistors or MOSFET

Consider the motor operate at 12V is about 2A.The value 2A can be compared with the rated drain current Id of MOSFET, which is 23A and has as can be seen, has lots of tolerance so that it will not overheat.

Design consideration 3: Opto-isolator

Opto-isolator is being considered here for high current design. This is to completely isolate circuitry of the microcontroller from the noisy motor circuitry. It is also used to switch the gate of a MOSFET on and off. The important characteristics of the Opto-isolator is the fast rise and fall times. This is to make sure that the MOSFET is fully switched on fast. The below calculation can be used to check whether an opto-isolator is suitable for an application.

Let's say that the wanted PWM frequency is 50Hz, therefore the period = 1/Freq = 20ms

10% duty cycle = 20ms *0.1 = 2ms
100% duty cycle = 20ms * 1 = 20ms

Given the specification from the opto-isolator 4n26 (used in this project):

Turn on time = 10us
Turn off time = 10us
Total delays = 20us

Therefore, since the total delays of the opto-isolator is able to switch on and off the MOSFETS fully and is suitable for use.

Design consideration 4: totem pole transistor as MOSFET driver

This is normally used for driving and switching along with the MOSFET. The transistor considered is the fast switching transistor which is the 2N3904for NPN and 2N3906 for PNP.

6.4 Totem-pole circuit

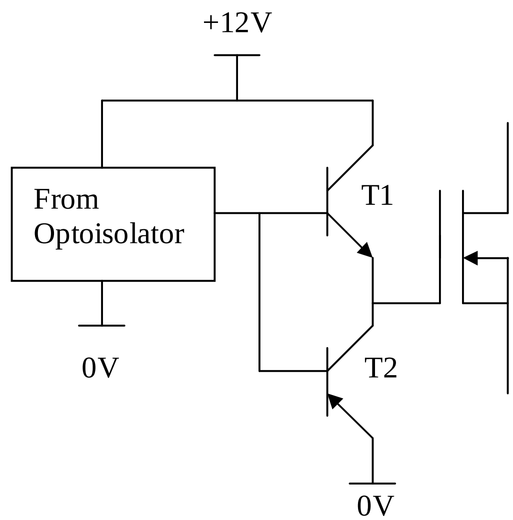


Figure 29**:** Totem-pole and P-channel Mosfet

N- Channel Mosfet	OFF	ON	BUZ10
P- Channel Mosfet	ON	OFF	MT8
Totem-pole	T1-ON,T2-OFF	T2-OFF,T2-ON	T1-NPN 2N3904 T2- PNP 2N3906
Optoisolator	ON	OFF	4N26

Table 3: H-bridge circuit components operation

‘high’ -5V/12V , ‘low’ -0V

From the above configuration the P-channel MOSFET will conduct or turn on when the opto-isolator outputs a ‘high’ signal, which then turn on the transistor T1 and the transistor T2 will then automatically switches off.

Meanwhile, the N-channel MOSFET will turn on when the opto-isolator outputs a ‘low’ signal. This signal will then turn off the transistor T1 and the transistor T2 will then switch on. As can be observed, when the N-channel MOSFET is turn on, the P-channel MOSFET turns off and vice-versa.

P-channel MOSFET Vg = 0V , Vs =12V	ON (Vgs = 0V)	OFF (Vgs = 12V)
N-channel MOSET Vg = 12V , Vs =0V	ON (Vgs = 12V)	OFF (Vgs = 0V)

Table 4: MOSFET's operation

The operations of MOSFETs are summarised as above table 4.

6.5 Unwanted situation

Referring to the figure 29 above MOSFET B and D should not turn on at the same time. Even though the programmer can program exactly such that only MOSFET A and D turn on and MOSFET B and C turn off there is a possibility that the MOSFET will fail in two situations below:

i) MOSFET does not turn on and off properly.

ii) MOSFET does not turn on and off fast enough.

6.6 Experiment

Bread boarding one n and p channel MOSFET circuit to test whether can turn the motor on or off. This experiment is also to determine which PWM frequency signal that can be used to drive the motor. Appropriate frequency is needed such that motor shaft would turn on smoothly without any jerky movement. In other words, the frequency must not be too low.

The MOSFET gate input is measured using oscilloscope to note whether pulse train shape is formed.

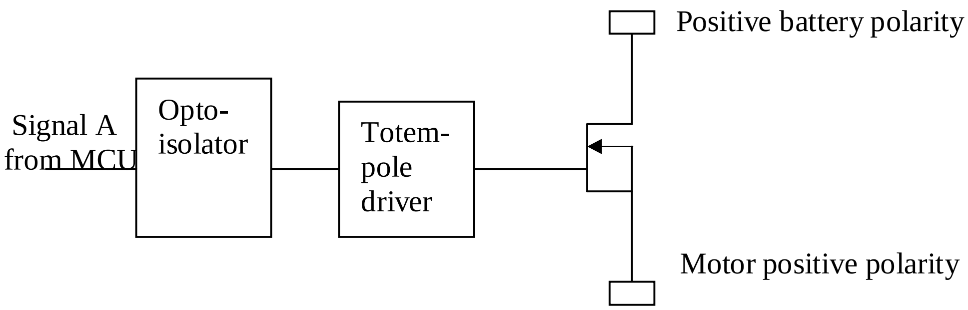
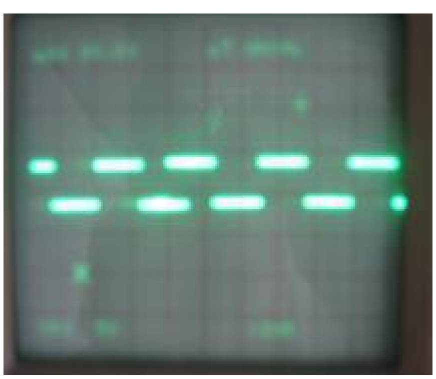


Figure 30: N-channel MOSFET motor driver

6.7 Results

6.7.1 Pulse Train



The left hand side photo indicates the pulses output from the microcontroller pin.

Voltage/div = 2.5 volts
Time/div = 10ms

Figure 31: Pulse train waveform

6.7.2 Current Surge

During the operation of the wiper motors there is a current surge whenever the motor starts to turn. Instead of the usual pulse waveform voltage spikes can be seen as 4 division space high. This is the motor running at rated 12v without gearing down. Imagine what would happen if the motor is geared down such as the ratio of 3. Current would shoot up three times higher. This would seriously damage the h-bridge circuit. Therefore, it is suggested that the voltage is ramped up to the voltage required within the short period of time such as 500ms.

6.7.3 Noise isolation

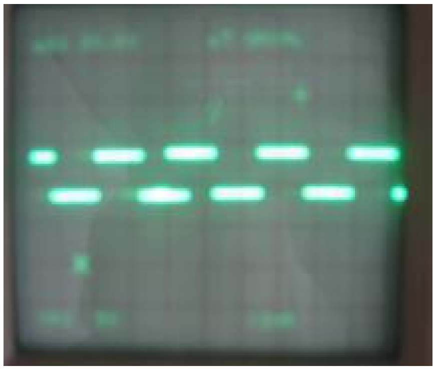


Photo on the left shows the cleaner pulse train on the microcontroller circuitry.

Figure 32: Clean pulse train waveform

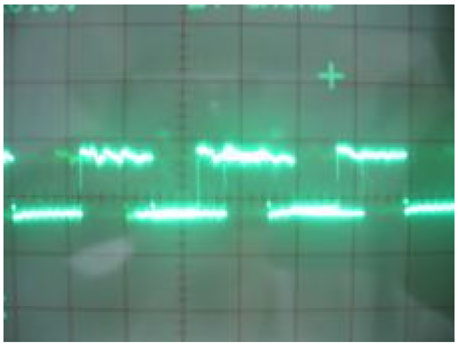


Photo on the left shows the noise corrupted pulse train on the MOSFET's circuitry.

Figure 33: Noise corrupted pulse train waveform

Chapter 7: Microcontroller & Software

The type of microcontroller used is the PIC 16F877A that can be bought from microchip website www.microchip.com. The diagram of the microcontroller is as shown below:

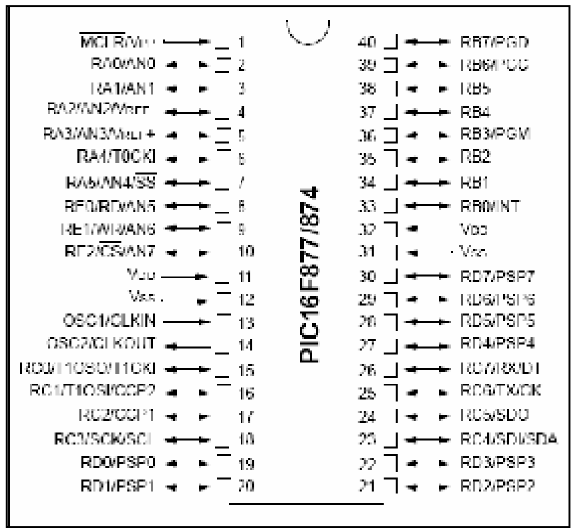


Figure 34: Microcontroller schematic diagram
(Source: PIC16F877A, Microchip)

The table below some of the important features of the PIC16F877A

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

Table 5: Comparison of microcontroller features
(Source: PIC 16F877A, Microchip)

7.1 Reasons

Reasons why PIC16F877A is chosen:

i) Many input output ports to choose from – 5 all together

ii) Huge memory space – Nearly four hundred bytes

iii) Has Analog to Digital modules to convert analog voltage (from sensors) to digital value.

iv) Interrupt feature

This function is very useful in programming as it enables two programs running at the same time, with one running in the background. When one of the programs is of higher priority and after some time the microcontroller for instance will service the higher priority program first and later return the handling to the other lower priority program.

7.2 Memory

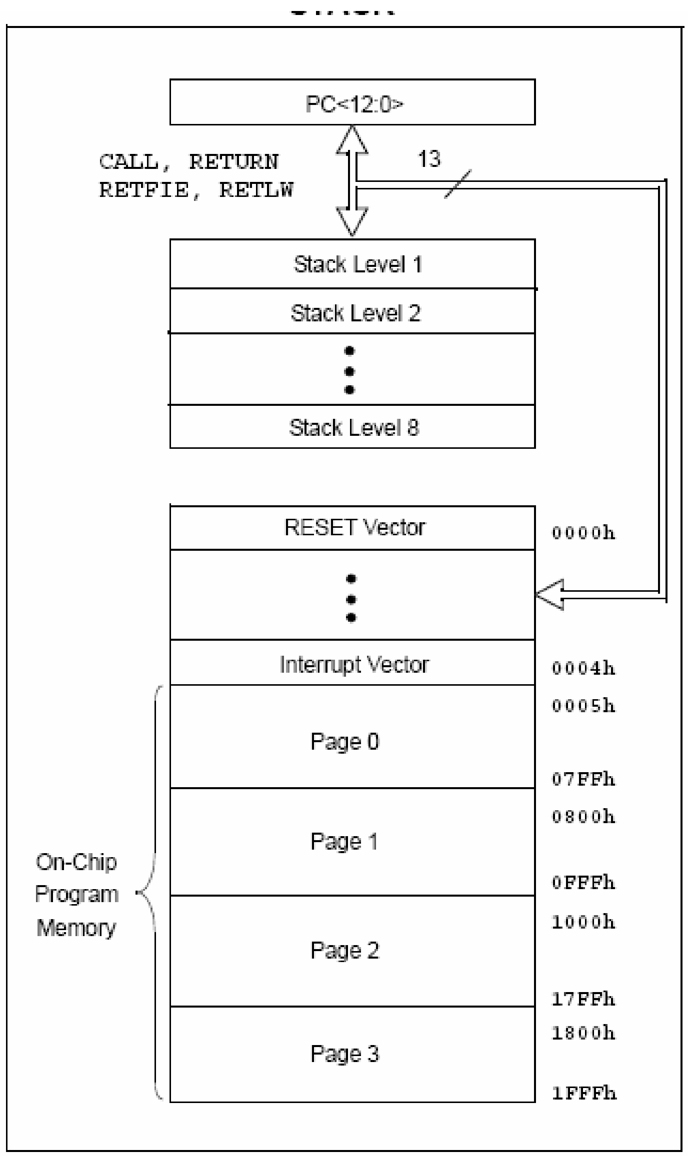


Figure 35: Microcontroller memory
(Source: PIC16F877A, Microchip)

PIC microcontrollers have two separate blocks of memory. One of them is the program memory and memory for file registers (PIC16F877A, Microchip). The program memory shown in the figure above represents the total amounts of program programmable into the PIC16F877. While the memory for file registers is the address for built-in registers. (i.e: STATUS, OPTION_REG) (PIC16F877A, Microchip).

7.3 Time consideration

7.3.1 Oscillators

There are four different types of clock oscillators that can be used with the PIC microcontroller. It is listed as follows:

a) RC - resistor/capacitor

b) XT – crystal or ceramic resonator

c) HS – high speed crystal or ceramic resonator

d) LP – low power crystal

The crystal or ceramic resonator is used because it is accurate and reliable. An example value for this crystal oscillator is 4 MHz.

Although the time to process one cycle looks like 1/4x10^6 = 0.25us it is actually not. The actual time for the processor to execute one instruction cycle is 1us. In other words, executing each instruction line takes 1us.

This is because the processor needs to undergo a few stages in order to successfully execute one machine cycle. They are fetch, decode, execute and store.

7.4 Software

One of the advantages of using the assembly program (MPLAB) is that the user gets to know in detail how much time used for every cycle executed.

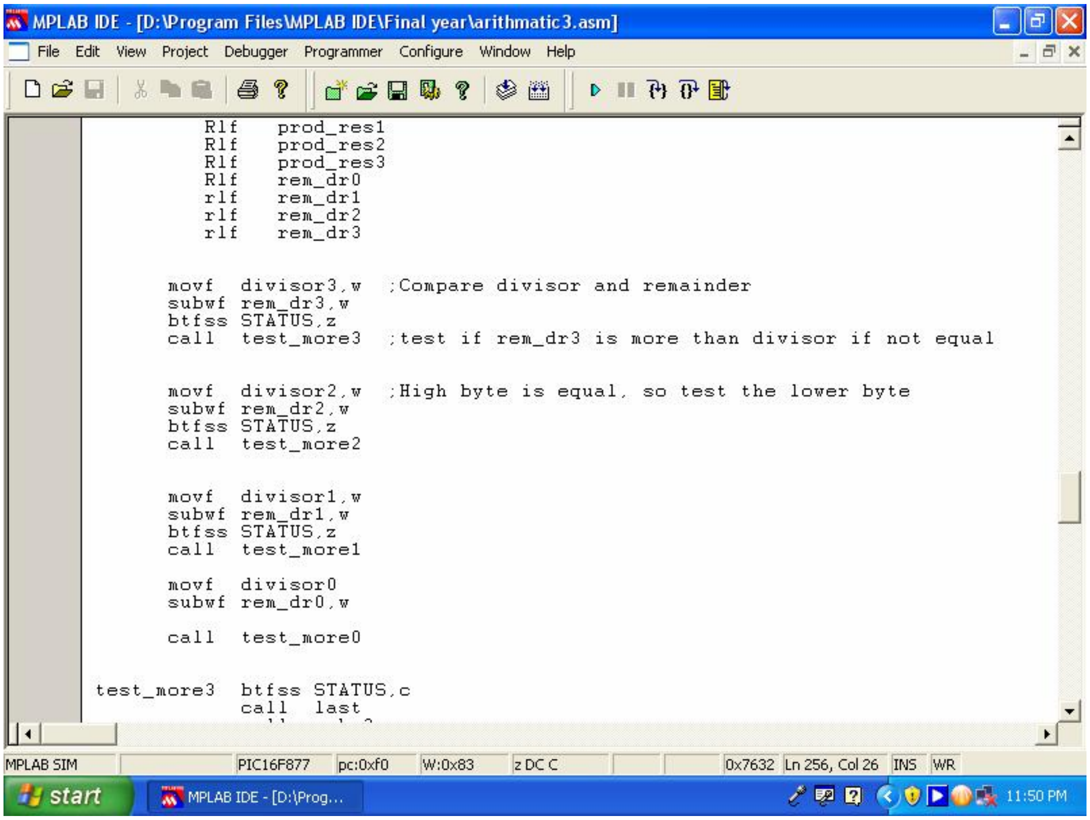


Figure 36: MPLAB interface program

7.5 Arithmetic Operation

Arithmetic Operation in assembly is different from what is used in high level language such as C. The four types of operations are shown and work out below. There will be a comparison of the C language and the respective addition, subtraction, multiplication and division.

The main difference is that programming is C is easier. However, by programming in assembly language a person can get to know how the computer operates the arithmetic operations.

Subtraction in C
Answer = 280 -8 = 272

Subtraction in Assembly

```
   00000001 00011000
-  00000000 00001000
--------------------
   00000001 00010000   (= 272)
--------------------
```

Multiplication in C
Answer =16*150 =2400

Multiplication in assembly

Assuming the multiplication in the 150*16

W=16 (00010000)
Lo=150 (10010110)

Rrf hi&lo (00000000 X1001011) , Carry = 0
(answer : 0)

No carry
Rrf hi & lo (00000000 0X100101), Carry =1
(answer:0)

carry! Hi+w (=00100101)
Rrf hi & lo (00010010 10X10010), Carry = 1
(answer:16*2= 32)

carry! Hi+w (=10110111)
Rrf hi & lo (01011011 110X1001), Carry = 0
(answer:16*2+16*4= 96)

no carry
Rrf hi & lo (00101101 1110X100), Carry = 1
(answer:0)

carry! Hi+w =(00010001)
Rrf hi & lo (00001000 11110X10), Carry=0
(answer: 16*2+16*4+ 16*16=352)

No carry
Rrf hi & lo (00000100 011110X1), Carry =0

No Carry
Rrf hi & lo (00000010 0011110X), Carry = 1

Carry! Hi+w (=00111110)
Rrf hi & lo (00011111 00000000), Carry=X
(answer: 16*2+16*4+ 16*16 +16*128 = 2400)

In assembly it is just shifting the registers of 150 (8 bit) value in 9 times and multiply the result as shown above. Values with more than 255(8 bit) can also be done the similar way as above. It is just that have to shift 17 times and multiply.

Full program listing is in the appendix D.

7.5.1 Trade-off

There are two major trade-offs that the programmer might face in using the assembly language. One is the level of tediousness. Programming the arithmetic operations can be time consuming and difficult to check for errors.

Secondly, it is the level of accurateness. It is not accurate to use the above simple methods to calculate the floating point (or decimal point). Any results that has decimal point the program will round them up and output integer values.

Chapter 8: PCB design consideration

There are several rules to be followed in designing a Printed Circuit Board (PCB). This is to make sure that the circuit to be created works properly by not having it affected by any unwanted interference (i.e noise) or current (in Amperes) free-flowing without any obstruction.

The main considerations are:

i) Components Arrangements

Components are arranged uniformly parallel or 90 degrees to each other.

ii) Conductor width

The copper tracks on the circuit board use the suitable track width. The current flow through the tracks is the main consideration. The higher current needed to flow the wider

track width is needed. (Electronic Measurement Workbook, USQ)

iii) Conductor length

Conductor's connection from one point to another should be as short as possible. The longer the conductor length the more disadvantage to the circuit be. This is because longer conductor length will act a transmission lines, which is susceptible to interference and noises from the external environment.(Electronic Measurement Workbook, USQ)

iv) Pad sizes

There is a standard rule of thumb the pad sizes should be about 2 to 2.5 times the diameter of the hole.

The designs for the microcontroller, accelerometer and h-bridge circuit boards are attached in the appendix B

Chapter 9: Discussion & Conclusion

9.1 Results

There are a few experiments that are tried and tested. It is as follows:

9.1.1 Gyro mounting

When testing the gyro at the laboratory the gyro exhibits a fast response. For instance, the faster the gyro rotates the faster the reading being output. Due to the fact that the gyro measures the rate of turn, and when the gyro stay still the gyro will output a steady state reading, which is 3.0V. Initially, the motor buzzing and after a while it keeps buzzing.

9.1.2 Accelerometer

In the experiment with the accelerometer alone the robot could not balance. There is a possibility that the response time is slow and could not output the signal to the motors to turn.

9.2 Reasons

The robot does not manage to balance by itself. There might be a few reasons for this.

i) Unsuitable Control

The control might be too simplistic even though the control strategy seems logical at. first. This issue is not realised initially as the programming language used in

assembly

ii) Does not provide the actual angle

As for the gyro as the sensor and it might not be too good in directly reading the values. Instead, we could try in adding an integrator block in whereby after integration the angular rate data would become the angular data. With this, the accelerometer might not be needed. This could be done in assembly such that when the time the robot takes to tilt the microcontroller is used to record the time and subsequently control the robot by lifting the chassis back up in the amount of time.

iii) Programming skill

There is a possibility that the programming skill may not up to the mark since this is the first time such a tedious program is being written.

9.3 Incomplete tasks

9.3.1 Gyro-Accelerometer Combo

Due to the time constrains this experiment is not able to be executed even though the program is already written. More time is needed to troubleshoot. If this problem is fixed then theoretically the user might get the robot balance more stable.

9.4 Conclusion

9.4.1 Recommendation for Future Work

Researchers could build on what is researched until now. There are a few experiments that are unaccomplished. That is the main drawback that hampers the overall project as concrete results is unable to attain. Therefore appropriate conclusions are not able to achieve.

Programming in assembly is tedious and time consuming. Even a slightest error may result in whole program could not work. Detecting the mistake is hard as there are so many lines of commands. Therefore, in future researchers might want to switch to C programming for future research and development.

References

Anderson, D.P, ‘Nbot, a two wheel balancing robot’, Viewed 28 February 2005
<http://www.geology.smu.edu/~dpa-www/robo/nbot>

Steve Hassenplug, 2002, ‘Steve’s Legway’, Viewed 4 March 2005
<http://www.teamhassenplug.org/robots/legway/>

Dean Kamen ,2001, Viewed 7 March 2005
<http://www.segway.com>

John Green,David Krakauer, March 2003, New iMEMS Angular Rate Sensing Gyroscope, Viewed 12 April 2005,
<http://www.analog.com/library/analogDialogue/archives/37-03/gyro.html>

Peter Hemsley, 32-bit signed integer maths for PICS, Viewed 2 April 2005
<http://www.piclist.com/techref/microchip/math/32bmath-ph.htm>

30 Jan2002, Mosfets and Mosfet's drivers,
<http://homepages.which.net/~paul.hills/SpeedControl/Mosfets.html>

Rick Bickle, 11 July 2003,‘DC motor control systems for robot applications’, Viewed 15 April
<http://www.dprg.org/tutorials/2003-10a/motorcontrol.pdf>

Carnegie Mellon, 26 August 1997, ‘Control Tutorials for Matlab’, The University of Michingan, Viewed 25 February 2005
<http://www.engin.umich.edu/group/ctm/PID/PID.html>

<http://www.boondog.com/tutorials/mouse/mouseHack.htm>

Martin Rowe, 11 January 2001, Measuring PWM motor efficiency, Test & Measurement World, Viewed 14 April 2005
<http://www.reed-electronics.com/tmworld/article/CA180848.html >

Gerry, 6 Ferbruary , Tilt sensors for your Robot, Viewed 2 March 2005
<http://www.roboticsindia.com/modules.php?name=News&file=article&sid=90>

(c) 1998, 2001 EME Systems, Berkeley CA U.S.A., Pulse Width Modulation, Viewed 29 February 2005
<http://www.emesystems.com/BS2PWM.htm>

2001 Microchip Technology Inc, PIC 16F87X data sheet, Viewed 14 February 2005 <www.microchip.com>

2004 Analog Devices, Inc, All rights reserved, ADXLS213, viewed 15 February 2005 <www.analogdevices.com>

2004 Analog Devices, Inc, All rights reserved, ADXRS300, viewed 15 February 2005 <www.analogdevices.com>

David Bension, version 4, Easy Microcontrol'n, A Beginners' guide to using PIC Microcontrollers, Square 1.

Dennis Clark and Michael Owings, 'Building Robot Drive Trains', McGraw Hill Companies.

D.W.Smith, 2002, PIC in Practice, Newnes, An imprint of Elsevier Science

Naoji Shiroma, Osamu Matsumoto, Shuji Kajita, Kazuo Tani, 'Cooperative Behavior of a Wheeled Inverted Pendulum for Object Transportation', Proceedings of the 1996 IEEE/RSJ International conference on Intelligent Robots and Systems '96, IROS 96, volume:2, 4-8Nov. 1996 Pg(s): 396-401 vol.

Grasser, Felix, Alonso D'Arrigo, Silvio Colombi & Alfred C. Rufer, 2002, 'JOE: A Mobile, Inverted Pendulum', IEEE Transactions on Industrial Electronics, vol 49.

Young Soo Suh, 'Attitude Estimation using Low Cost Accelerometer and Gyroscope', Proceedings of the 7th Korea-Russia International Symposium, KORUS 2003,Pg(s) 423-427.

Albert-Jan Baervaeldt and Robert Klang, 'A Low-cost and Low-weight Attitude Estimation System for an Autonomous Helicopter', Halmstad University, Sweden, Pg(s) 391-395.

Yongjun Hou, Greg R.Luecke, October 5-8 2003, 'Control of the Tight Rope Balancing Robot', Proceedings of the 2003 IEEE International Symposium on Intelligent Control, Houston, Texas, Pg(s): 896-901.

Alessio Salerno and Jorge Angles, 'The Control of Semi-Autonomous Two-Wheeled Robots Undergoing Large Payload-Variations', Proceedings of the 2004 IEEE International Conference on Robotics & Automation, New Orleans, LA, Pg(s): 1740-1745.

Kiyoshi Komoriya and Eimei Oyama, 'Position Estimation of a Mobile Robot Using Optical Fiber Gyroscope (OFG)', Pg(s): 143-149.

Prof. John Billingsley, Mechatronics Practice Unit, Viewed 11 May 2005 <http://www.usq.edu.au/course/material/eng3905/>

Dr.Tony Ah Fock, Power Electronics study book 1 and 2, University of Southern Queensland.

Paul E.Sandin , “Robot Mechanisms and Mechanical Devices Illustrated” , The McGrawHill companies.

J.J. D’Azzo, C.H. Houpis, Feedback Control System Analysis and Synthesis, second edition, McGraw-Hill International Editions. (ISBN 0-07-Y85150-6) Pg11

‘Inverted Pendulum’, Microrobot NA, Viewed 27 February 2005 http://www.microrobotna.com/pendulum.htm

R.E Kalman, 1960, ‘A New Approach to Linear Filtering and Prediction Problems’, Transactions of the ASME- Journal of Basic Engineering, 82(Series D), pp35-45.

Appendix A:
Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	**<u>HO</u>, KHOON CHYE (RANDAL)**
TOPIC:	BALANCING WHEELED ROBOT
SUPERVISORS:	Mr. Mark Phythian (USQ) Dr. Izham Bin Zainal Abidin (Uniten, Malaysia)
ENROLMENT:	ENG4111-S1, XP, 2005 ENG4112-S2, XP, 2005
PROJECT AIM:	This project aim is to build a robot that can balance itself on two wheels without falling.
SPONSORSHIP:	Individual / USQ
PROGRAMME:	Issue A, 21 March 2005

1. Research information on designing a two wheel balancing robot by identifying the type of programming software to use and selecting control methods to be applied on the robot.

2. Design and assemble two-wheel robot base.

3. Understand, write and troubleshoot MATLAB programming software on determining the behavior of robot and torque needed by motor to balance the robot in the upright position.

4. Analyze and understand the process flow of the motor and chassis controlling program.

5. Write and troubleshoot PIC programming software on the controlling of motor and chassis of robot.

6. Design Printed Circuit Board for motor controller and sensors.

7. Evaluate the performance of balancing robot by implementing two types of sensors.

As time permits:

8. Add and design extra feature for the robot that can follow lines while traveling.

AGREED:

________________(Student) ______________, _____________ (Supervisors)

(dated) ___/___/___

Appendix B

Source Codes

Accelerometer Acquisition program (MPLAB)

```
List      p=16F877a
              include  "p16f877a.inc"
              __config _cp_off & _wdt_off & _xt_osc & _pwrte_on


;Reading Accelerometer duty cycle value

;This subroutine collects and calculates T1X, T1Y and T2
;T1X is represented by registers T1XHi and T1Xlo
;T1Y is represented by registers T1YHi and T1Ylo
;T2 is represented by registers T2Hi and T2lo

T1XEndlo            equu    46h
T1XEndHi            equ     47h
T1Ybeginlo          equ     48h
T1YbeginHi          equ     49h
T1YEndlo            equ     50h
T1YEndHi            equ     55h
T1YHi               equ     56h
T1YLo               equ     81h
T1XHi               equ     82h
T1XLo               equ     83h
T2Hi                equ     84h
T2Lo                equ     85h
ZXcalHi             equ     86h
ZXcalLo             equ     87h
ZXActualHi          equ     88h
ZXActualLo          equ     89h
u_term_lo_acce      equ     95h
u_term_hi_acce      equ     96h
KHi                 equ     97h
KLo                 equ     98h

; Start at the reset vector
org     0x000
goto    start

org 0x0004
            incf        Timer1H
            bcf         INTCON,T0IF
            bcf         INTCON,RBIE
            retfie

Start
```

```
        bcf     STATUS,RP0
        clrf    PORTA
        clrf    PORTB
        bsf     STATUS,RP0                      ;Bank1
        movlw   B'00000011'          ;Set up the I/O ports
        movwf   TRISA
        movlw   B'00010000'
        movwf   TRISB
        movlw   B'00001111'
        movwf   OPTION_REG
        bcf     STATUS,RP0                      ;Bank0
        bsf     INTCON,GIE

        Movlw   b'00100011'
        Movwf   T1CON
        Movlw   b'00000101'
        Movwf   CCP1
        bsf     INTCON,GIE


EdgeA           btfsc   PORTA,0
                Goto    EdgeA

EdgeB           btfss   PORTA,0         ;Look for the high transmission at
Ta
                Goto    EdgeB           ;Keep looking for high

transmission
                Clrf    TMR1L                           ;Start timing
                Clrf    TMR1H
                Bcf     PIR1,TMR1IF     ;Enabling the timer1 overflow interrupt
                bsf     PIE1,TMR1IE


EdgeC           btfsc   PORTA,0     ;Look for the low transmission at Tb
                Goto    EdgeC       ;Keep looking for low transmission
                Movf    TMR1L,w     ;Record and save the time in register T1X
                MovwfT1XEndlo
                Movf    TMR1H
                MovwfT1XEndHi


EdgeD           btfsc   PORTB,2
                goto    EdgeD

EdgeE           btfsc   PORTB,2     ;Look for the high transmission at Tc
                Goto    EdgeE       ;Keep looking for high transmission
```

```
                Movf   TMR1L,w      ;Record and save the time in T1Ybeginlo
                Movwf   T1Ybeginlo
                Movf    TMR1H,w
                Movwf   T1YbeginHi

EdgeF           btfsc    PORTB,2              ;Look for the low transmission at
Td
                Goto    EdgeF                ;Keep looking for low transmission
                Movf   TMR1L,w               ;Record and save the time in
                Movwf        T1YEndlo        ; T1YEndlo
                Movf         TMR1H,w
                Movwf        T1YEndHi

                Movf         T1YEndHi,w
                Movwf        Arg_hi
                Movf         T1YEndlo,w
                Movwf        Arg_lo
                Movf         T1YbeginHi,w
                Movwf         Sum_Hi
                Movf         T1YbeginLo,w
                Movwf        Sum_Lo
                call         Subtract
                Movf         Sum_Hi,w
                Movwf        T1YHi
                Movf         Sum_Lo,w
                Movwf        T1YLo


;CALCULATE T2
;2*(T2Hi,T2Lo) = (T1YEndHi:T1YEndLo)+
;(T1YStartHi:T1YStartLo)-(T1XHi:T1XLo)

                movf         T1YEndHi,w
                movwf        Arg_Hi
                movf         T1YEndLo,w
                movwf         Arg_Lo
                movf         T1YbeginHi,w
                movwf        Sum_Hi
                movf         T1YbeginLo,w
                movwf        Sum_lo
                call         add

           ;Sum_hi,Sum_lo=(T1YEndHi:T1YEendLo)+
                movf         T1XEndHi,w          ;
(T1YBeginHi:T1YBeginLo)
                movwf        Sum_Hi
                movf         T1XEndLo,W
```

```
                movwf       Sum_lo
                call        Subtract            ;Sum_hi:Sum_lo = 2*T2
                bcf         STATUS,C
                rrf         Arg_hi,F            ;rotate one bit means
multiply
                rrf         Arg_lo,F            ; by two
                movf        Arg_hi,W
                movwf       T2Hi
                movf        Arg_lo,W
                movwf       T2Lo
                return


; Calulation of the Z value based on the formula
; Zactual = (Zcal * T2actual)/T2cal

ZActual_value   movf        ZXcalHi,w
                Movwf       Arg_Hi
                Movf        ZXcalLo,w
                Movwf       Arg_Lo
                Movf        T2Hi,w
                Movwf       Sum_Hi
                Movf        T2Lo,w
                Movwf       Sum_Lo
                Call        Mul

                Movf        T2calHi,w
                Movwf       Divisor1
                Movf        T2calLo,w
                Movwf       Divisor0
                Call        Division
                Movf        Quo_1,w
                Movwf       ZXActualHi
                Movf        Quo_0,w
                Movwf       ZXActualLo



; The x-axis acceleration value is programmed based on the formula
; Acceleration = K*(T1-Zactual)/T2actual


X_Accel_value   movf        ZXActualHi,w ; This is to check whether the
                Subwf       T1XHi,w      ; numerator  is positive or negative
                Btfss       STATUS,c
                Goto        Num_negx
                Btfss       STATUS,z
```

```
                Goto        Num_posx
                Movf        ZXActualLo,w
                Subwf       T1XLo,w
                Btfss       STATUS,c
                Goto        Num_posx
```

;This subroutine is chosen if the x-axis acceleration value is negative

```
Num_negx        movf        ZXActualHi,w
                Movwf       Arg_Hi
                Movf        ZXActualLo,w
                Movwf       Arg_Lo
                Movf        T1XHi,w
                Movwf       Sum_hi
                Movf        T1XLo,w
                Movwf       Sum_lo
                Call        Subtract
                Movlw       KHi
                Movwf       Arg_Hi
                Movlw       KLo
                Movwf       Arg_Lo
                Call        Mul

                Movf        T2Hi,w
                Movwf       Divisor1
                Movf        T2Lo,w
                Movwf       Divisor0
                Call        Division

                Movf        T2Hi,w
                Movwf       Divisor1
                Movf        T2Lo,w
                Movwf       Divisor0
                Call        Division
                movf        Quo_0,w
                movwf       u_term_hi_acce
                movf        Quo_1,w
                movwf       u_term_lo_acce


                movf        u_term_hi_acce,w
                sublw       b'00000010' ; Upper byte for analog value 3V
                call        no_drive
                movwf       temp_lo
                btfsc       STATUS,c
                call        ccw
                call        cw
```

```
no_drive        movwf   temp_lo     ;No Signal is output to the h-bridge
                btfsc   STATUS,z
                call    next_byte3
                return

;This section is to determine which direction the motor should turn
next_byte3
                bcf     STATUS,c
                bcf     STATUS,z
                bsf     STATUS,RP0
                movf    u_term_lo_acce,w
                bcf     STATUS,RP0
                sublw   b'00000001'   ;lower byte for analog value 3V
                call    no_drive_test2  ;This is the balanced state value
                btfsc   STATUS,c
                call    ccw           ;If  no carry then go to ccw
subroutine
                call    cw          ;If carry then go to cw subroutine


no_drive_test2  movwf   temp_lo
                btfsc   STATUS,z
                call    clr_drive   ;No Signal is output to the h-bridge
                return


;This ccw subroutine contains a few individual error values, which in turn will call the
;appropriate duty cycle subroutines.

ccw
                movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3 – 1.8)v
                btfsc   STATUS,z
                call    upper_byte


valtest1        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3-1.9)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upper_byte2
                movwf   test_bytelo
                btfss   STATUS,c
                call    set_cycle1
```

```
valtest2        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3-2)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upper_byte3
                movwf   test_bytelo
                btfss   STATUS,c
                call    set_cycle2


valtest3        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3-2.1)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upper_byte4
                movwf   test_bytelo
                btfss   STATUS,c
                call    set_cycle3


valtest4        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value(3-2.2)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upper_byte5
                movwf   test_bytelo
                btfss   STATUS,c
                call    set_cycle4



valtest5        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3-2.3)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upper_byte6
                movwf   test_bytelo
                btfss   STATUS,c
                call    set_cycle5


valtest6        movf    u_term_hi_acce,w
                sublw   b'00000000'   ; upper byte error value (3-2.4)v
                movwf   test_bytelo
                btfsc   STATUS,z
```

```
                call     upper_byte7
                movwf    test_bytelo
                btfss    STATUS,c
                call     set_cycle6


valtest7        movf     u_term_hi_acce,w
                sublw    b'00000000'    ; upper byte error value (3-2.5)v
                movwf    test_bytelo
                btfsc    STATUS,z
                call     upper_byte8
                movwf    test_bytelo
                btfss    STATUS,c
                call     set_cycle7

valtest8        movf     u_term_hi_acce,w
                sublw    b'00000000'    ; upper byte error value (3-2.6)v
                movwf    test_bytelo
                btfsc    STATUS,z
                call     upper_byte9
                movwf    test_bytelo
                btfss    STATUS,c
                call     set_cycle8

valtest9        movf     u_term_hi_acce,w
                sublw    b'00000000'    ; upper byte error value (3-2.7)v
                movwf    test_bytelo
                btfsc    STATUS,z
                call     upper_byte8
                movwf    test_bytelo
                btfss    STATUS,c
                call     set_cycle9

valtest10       movf     u_term_hi_acce,w
                sublw    b'00000000'    ; upper byte error value (3-2.8)v
                movwf    test_bytelo
                btfsc    STATUS,z
                call     upper_byte9
                movwf    test_bytelo
                btfss    STATUS,c
                call     set_cycle10

valtest11       movf     u_term_hi_acce,w
                sublw    b'00000000'    ; upper byte error value (3-2.9)v
                movwf    test_bytelo
                btfsc    STATUS,z
                call     upper_byte10
```

```
                movwf     test_bytelo
                btfss     STATUS,c
                call      set_cycle11
                call      clr_drive


upper_byte      movf      u_term_lo_acce,w
                sublw     b'11110110'   ; lower byte error value (3 – 1.8)v
                movwf     test_bytelo
                btfss     STATUS,c
                call      set_cycle1

upper_byte2     movf       u_term_lo_acce,w
                sublw     b'11100010'   ; lower byte error value  (3 – 1.9)v
                movwf     test_bytehi
                btfss     STATUS,c
                call      set_cycle1
                call      valtest2

upper_byte3     movf       u_term_lo_acce,w
                sublw     b'11001101'   ; lower byte error value  (3 – 2.0)v
                movwf     test_bytehi
                btfss     STATUS,c
                call      set_cycle2
                call      valtest3

upper_byte4     movf      u_term_lo_acce,w
                sublw     b'10111001'   ; lower byte error value (3 – 2.1)v
                movwf     test_bytehi
                btfss     STATUS,c
                call      set_cycle3
                call      valtest4

upper_byte5     movf      u_term_lo_acce,w
                sublw     b'10100100'   ; lower byte error value  (3 – 2.2)v
                movwf     test_bytehi
                btfss     STATUS,c
                call      set_cycle4
                call      valtest5


upper_byte6     movf      u_term_lo_acce,w ;lower byte error value (3 –
2.3)v
                sublw     b'10010000'
                movwf     test_bytehi
                btfss     STATUS,c
                call      set_cycle5
```

```
                call        valtest6


upper_byte7     movf        u_term_lo_acce,w ;lower byte error value (3 –
2.4)v
                sublw       b'01111011'
                movwf       test_bytehi
                btfss       STATUS,c
                call        set_cycle6
                call        valtest7

upper_byte8     movf        u_term_lo_acce,w ;lower byte error value  (3 –
2.5)v
                sublw       b'01100111'
                movwf       test_bytehi
                btfss       STATUS,c
                call        set_cycle7
                call        valtest8

upper_byte9     movf        u_term_lo_acce,w ;lower byte error value (3 –
2.6)v
                sublw       b'01010010'
                movwf       test_bytehi
                btfss       STATUS,c
                call        set_cycle8
                call        valtest9

upper_byte10    movf        u_term_lo_acce,w ;lower byte error value  (3 –
2.7)v
                sublw       b'00111110'
                movwf       test_bytehi
                btfss       STATUS,c
                call        set_cycle9
                call        valtest10

upper_byte11    movf        u_term_lo_acce,w ;lower byte error value (3 –
2.8)v
                sublw       b'00101001'
                movwf       test_bytehi
                btfss       STATUS,c
                call        set_cycle10
                call        valtest11


upper_byte12    movf        u_term_lo_acce,w ;lower byte error value  (3 –
2.9)v
                sublw       b'00010101'
```

```
                movwf   test_bytehi
                btfss   STATUS,c
                call    set_cycle11
                goto    $-1
```

;The duty cycle values based on ON and OFF Pulse Width Modulation

```
set_cycle1      movlw   h'E5'           ; 100% duty cycle
                movwf   fr_cnt
                movlw   h'FE'
                movwf   fr_cnt2
                call    dir_chg
                call    ramp_rou

set_cycle2      movlw   h'E7'           ;95% duty cycle
                movwf   fr_cnt
                movlw   h'FD'
                movwf   fr_cnt2
                call    dir_chg
                call    ramp_rou

set_cycle3      movlw   h'E8'           ;90% duty cycle
                movwf   fr_cnt
                movlw   h'FC'
                movwf   fr_cnt2
                call    dir_chg
                call    ramp_rou

set_cycle4      movlw   h'E9'           ;85% duty cycle
                movwf   fr_cnt
                movlw   h'FB'
                movwf   fr_cnt2
                call    dir_chg
                call    ramp_rou


set_cycle5      movlw   h'EA'           ;80% duty cycle
                movwf   fr_cnt
                movlw   h'FA'
                movwf   fr_cnt2
                call    dir_chg
                call    ramp_rou


set_cycle6      movlw   h'EC'           ;75% duty cycle
                movwf   fr_cnt
```

```
                movlw       h'F9'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set_cycle7      movlw       h'ED'           ;70% duty cycle
                movwf       fr_cnt
                movlw       h'F7'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set_cycle8      movlw       h'EE'           ;65% duty cycle
                movwf       fr_cnt
                movlw       h'F6'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set_cycle9      movlw       h'EF'           ;60% duty cycle
                movwf       fr_cnt
                movlw       h'F5'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set_cycle10     movlw       h'F1'           ;55% duty cycle
                movwf       fr_cnt
                movlw       h'F4'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set_cycle11     movlw       h'F2'           ;50% duty cycle
                movwf       fr_cnt
                movlw       h'F2'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou


clr_drive       bcf         PORTB,5
                bcf         PORTB,7
                bcf         PORTB,1
                bcf         PORTB,2
                movlw       h'e5'
                movwf       fr_cnt
```

```
                call        delay1
                call        dir_chg2
```

; Time delay in ensuring the current is fully flowed to ground before
; Starting to turn the other direction of motor. This is to prevent short circuit from happening. The delay is about 500ms

```
dir_chg2        movlw       h'FF'
                movwf       cnt4
Con2            movlw       h'FF'
                Movwf       cnt5
                decfsz      cnt5,1
                goto        $-1
                decf        cnt4,1
                movf        cnt4,w
                sublw       h'be'
                btfss       status,z
                goto        Con2
                incf        num_times
                movf        num_times,w
                subwf       fr_cnt1,w

                btfss       status,z
                goto        dir_chg2
                call        EdgeA
```

;The other time delay for motor to turn the other direction. The function is same as above
;subroutine

```
dir_chg         movlw       h'FF'
                movwf       cnt4
Con3            movlw       h'FF'
                movwf       cnt5
                decfsz       cnt5,1
                goto        $-1
                decf        cnt4,1
                movf        cnt4,w
                sublw       h'be'
                btfss       status,z
                goto        Con3
                incf        num_times
                movf        num_times,w
                sublw       d'20'
```

```
                btfss       status,z
                goto        dir_chg
                return

; Thie Ramp_rou subroutine is to start the ramping up of motor voltage
ramp_rou        movlw       h'fc'
                movwf       ramp_cnt
                movlw       h'ea'
                movwf       ramp_cnt2
                call        PWM_ramp
                decf        ramp_cnt
                movf        fr_cnt,w
                subwf       ramp_cnt,w
                btfss       STATUS,c
                call        PWMbegin
                incf        ramp_cnt2
                call        PWM_ramp
                goto        ramp_rou

PWM_ramp        bcf         PORTB,7
                bsf         PORTB,4
                call        delay1
                bcf         PORTB,2
                bsf         PORTB,1
                call        delay2
                bcf         PORTB,7
                bcf         PORTB,4
                call        delay1
                bcf         PORTB,2
                bcf         PORTB,1
                call        delay2
                return

; Output the PWM signal to respective PORTS

PWMbegin        bcf         PORTB,7
                bsf         PORTB,4
                call        delay1
                bcf         PORTB,2
                bsf         PORTB,1
                call        delay2
                bcf         PORTB,7
                bcf         PORTB,4
                call        delay1
                bcf         PORTB,2
                bcf         PORTB,1
                call        delay2
```

```
                incf        cnt3
                movf        cnt3,w
                sublw       d'50'
                btfss       status,z
                goto        PWMbegin
                clrf        cnt3
                clrf        num_times
                 call       EdgeA
```

;Below are the subroutines to select the appropriate duty cycle values based on the on and
;off time of pulses.

```
set2_cycle1     movlw       h'E5'
                movwf       fr_cnt
                movlw       h'FE'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set2_cycle2     movlw       h'E7'
                movwf       fr_cnt
                movlw       h'FD'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set2_cycle3     movlw       h'E8'
                movwf       fr_cnt
                movlw       h'FC'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set2_cycle4     movlw       h'E9'
                movwf       fr_cnt
                movlw       h'FB'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou

set2_cycle5     movlw       h'EA'
                movwf       fr_cnt
                movlw       h'FA'
                movwf       fr_cnt2
                call        dir_chg
                call        ramp_rou
```

```
set2_cycle6     movlw     h'EC'
                movwf     fr_cnt
                movlw     h'F9'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou


set2_cycle7     movlw     h'ED'
                movwf     fr_cnt
                movlw     h'F7'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou

set2_cycle8     movlw     h'EE'
                movwf     fr_cnt
                movlw     h'F6'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou

set2_cycle9     movlw     h'EF'
                movwf     fr_cnt
                movlw     h'F5'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou

set2_cycle10    movlw     h'F1'
                movwf     fr_cnt
                movlw     h'F4'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou

set2_cycle11    movlw     h'F2'
                movwf     fr_cnt
                movlw     h'F2'
                movwf     fr_cnt2
                call      dir_chg
                call      ramp_rou


delay1          movlw     h'FF'    ; PWM modulation based on the
                movwf     cnt2     ;values given by fr_cnt and fr_cnt2
```

```
Con         movlw   h'FF'
            movwf   cnt1
            decfsz  cnt1,1
            goto    $-1
            decf    cnt2,1
            movf    cnt2,w
            subwf   fr_cnt,w
            btfss   status,z
            goto    Con
            return

delay2      movlw   h'FF'
            movwf   cnt2
Con5        movlw   h'FF'
            movwf   cnt1
            decfsz  cnt1,1
            goto    $-1
            decf    cnt2,1
            movf    cnt2,w
            subwf   fr_cnt2,w

            btfss   status,z
            goto    Con5
            return


cw
            comf    u_term_hi_acce,f
            comf    u_term_lo_acce,f
            movf    u_term_hi_acce,w
            sublw   b'00000000'     ;Upper byte error value (3.1-3)v
            btfsc   STATUS,z
            call    upperbyte


val_test1   movf    u_term_hi_acce,w
            sublw   b'00000000'     ;Upper byte error value (3.2-3)v
            movwf   test_bytelo
            btfsc   STATUS,z
            call    upperbyte9
            movwf   test_bytelo
            btfsc   STATUS,c
            call    set2_cycle11



val_test2   movf    u_term_hi_acce,w
```

```
                sublw     b'00000000'   ;Upper byte error value (3.3-3)v
                movwf     test_bytelo
                btfsc     STATUS,z
                call      upperbyte11
                movwf     test_bytelo
                btfsc     STATUS,c
                call      set2_cycle10


val_test3       movf      u_term_hi_acce,w
                sublw     b'00000000'   ;Upper byte error value  (3.4-3)v
                movwf     test_bytelo
                btfsc     STATUS,z
                call      upperbyte10
                movwf     test_bytelo
                btfsc     STATUS,c
                call      set2_cycle9


val_test4       movf      u_term_hi_acce,w
                sublw     b'00000000'   ;Upper byte error value (3.5-3)v
                movwf     test_bytelo
                btfsc     STATUS,z
                call      upperbyte9
                movwf     test_bytelo
                btfsc     STATUS,c
                call      set2_cycle8



val_test5       movf      u_term_hi_acce,w
                sublw     b'00000000'   ;Upper byte error value (3.6-3)v
                movwf     test_bytelo
                btfsc     STATUS,z
                call      upperbyte8
                movwf     test_bytelo
                btfsc     STATUS,c
                call      set2_cycle7


val_test6       movf      u_term_hi_acce,w
                sublw     b'00000000'   ;Upper byte error value (3.7-3)v
                movwf     test_bytelo
                btfsc     STATUS,z
                call      upperbyte7
                movwf     test_bytelo
                btfsc     STATUS,c
```

```
                call    set2_cycle6

val_test7       movf    u_term_hi_acce,w
                sublw   b'00000000'   ;Upper byte error value (3.8-3)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upperbyte6
                movwf   test_bytelo
                btfsc   STATUS,c
                call    set2_cycle5

val_test8       movf    u_term_hi_acce,w
                sublw   b'00000000'   ;Upper byte error value (3.9-3)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upperbyte5
                movwf   test_bytelo
                btfsc   STATUS,c
                call    set2_cycle4

val_test9       movf    u_term_hi_acce,w
                sublw   b'00000000'   ;Upper byte error value (4.0-3)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upperbyte4
                movwf   test_bytelo
                btfsc   STATUS,c
                call    set2_cycle3

val_test10      movf    u_term_hi_acce,w
                sublw   b'00000000'   ;Upper byte error value (4.1-3)v
                movwf   test_bytelo
                btfsc   STATUS,z
                call    upperbyte3
                movwf   test_bytelo
                btfsc   STATUS,c
                call    set2_cycle2

val_test11      movf    u_term_hi_acce,w ;Upper byte error value (4.2-
3)v
                sublw   b'00000000'
                movwf   test_bytelo
                btfsc   STATUS,z
```

```
                call    upperbyte2
                movwf   test_bytelo
                btfsc   STATUS,c
                call    set2_cycle1
                call    clr_drive


upperbyte       movf    u_term_lo_acce,w
                sublw   b'00010100'   ;lower byte error value (3.1-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle11


upperbyte2      movf    u_term_lo_acce,w
                sublw   b'00101001'   ;lower byte error value (3.2-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle11
                call    val_test11

upperbyte3      movf    u_term_lo_acce,w
                sublw   b'00111101'   ;lower byte error value (3.3-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle10
                call    val_test10

upperbyte4      movf    u_term_lo_acce,w
                sublw   b'01010010'   ;lower byte error value (3.4-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle9
                call    val_test9

upperbyte5      movf    u_term_lo_acce,w
                sublw   b'01100110'   ;lower byte error value (3.5-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle8
                call    val_test8

upperbyte6      movf     u_term_lo_acce,w
                sublw   b'01111011'   ;lower byte error value (3.6-3)v
                movwf   test_bytehi
                btfsc   STATUS,c
```

```
                call        set2_cycle7
                call        val_test7


upperbyte7      movf         u_term_lo_acce,w
                sublw       b'10001111'    ;lower byte error value (3.7-3)v
                movwf       test_bytehi
                btfsc       STATUS,c
                call        set2_cycle6
                call        val_test6

upperbyte8      movf        u_term_lo_acce,w ;lower byte error value (3.8-
3)v
                sublw       b'10100100'
                movwf       test_bytehi
                btfsc       STATUS,c
                call        set2_cycle5
                call        val_test5

upperbyte9      movf        u_term_lo_acce,w ;lower byte error value  (3.9-
3)v
                sublw       b'10111000'
                movwf       test_bytehi
                btfsc       STATUS,c
                call        set2_cycle4
                call        val_test4


upperbyte10     movf         u_term_lo_acce,w     ;lower byte error value (4-
3)v
                sublw       b'11001101'
                movwf       test_bytehi
                btfsc       STATUS,c
                call        set2_cycle3
                call        valtest3

upperbyte11     movf        u_term_lo_acce,w;lower byte error value (4.1-3)v
                sublw       b'11100001'
                movwf       test_bytehi
                btfsc       STATUS,c
                call        set2_cycle2
                call        valtest2
```

```
upperbyte12     movf    u_term_lo_acce,w ;lower byte error value (4.2-
3)v
                sublw   b'11110110'
                movwf   test_bytehi
                btfsc   STATUS,c
                call    set2_cycle1
                call    valtest1
```

Overall balancing program (Gyro only) (MPLAB)

```
        List    p=16F877a
                include  "p16f877a.inc"
                __config _cp_off & _wdt_off & _xt_osc & _pwrte_on



cnt1                        equ     2AH
cnt2                        equ     2CH
cnt3                        equ     2DH
cnt4                        equ     2EH
cnt5                        equ     3BH
num_times                   equ     3DH
fr_cnt                      equ     3FH
fr_cnt2                     equ     7AH
fr_cnt1                     equ     7BH
test_bytehi                 equ     6AH
test_bytelo                 equ     6CH
gy_calc_angle_vel_new_hi    equ     20H
gy_calc_angle_vel_new_lo    equ     21H
gy_calc_angle_new_hi        equ     22H
gy_calc_angle_new_lo        equ     23H
tilt_temp_hi                equ     24H
tilt_temp_lo                equ     25H
Kt_tilt_hi                  equ     26H
Kt_tilt_lo                  equ     27H
Kv_tilt_hi                  equ     28H
Kv_tilt_lo                  equ     29H
Ka_tilt_hi                  equ     30H
Ka_tilt_lo                  equ     31H
tilt_rate_hi                equ     32H
tilt_rate_term_lo           equ     33H
gy_calc_angle_old_hi        equ     34H
gy_calc_angle_old_lo        equ     35H
gy_calc_angle_vel_old_hi    equ     36H
gy_calc_angle_vel_old_lo    equ     37H
u_term_hi                   equ     38H
u_term_lo                   equ     39H
Time_lo                     equ     40H
Time_hi                     equ     41H
set_point_lo                equ     43H
set_point_hi                equ     45H
prod_res0                   equ     54H
prod_res1                   equ     53H
```

```
prod_res2               equ     52H
prod_res3               equ     51H
Arg_hi                  equ     57H
Arg_lo                  equ     58H
Sum_hi                  equ     59H
Sum_lo                  equ     60H
var_hi                  equ     61H
var_lo                  equ     62H
pwm_bit_cnt             equ     63H
pwm_bit                 equ     64H
pwm_bit_cnt2            equ     65H
pwm_bit2                equ     66H
count                   equ     67H
TILT_RATE_LO            equ     68H
tilt_term_hi            equ     69H
tilt_term_lo            equ     70H
tilt_rate_term_hi       equ     71H
temp_lo                 equ     72H
gy_temp_angle_old_hi    equ     73H
gy_temp_angle_old_lo    equ     74H
tilt_hi                 equ     75H
tilt_lo                 equ     76H
ramp_cnt                equ     77H
ramp_cnt2               equ     78H
```

```
        ; Start at the reset vector
    org     0x000
    goto    start

Start
        bsf         STATUS,RP0          ;Bank 1
        bcf         STATUS,RP1
        clrf        TRISB               ;PORTB [7-0] outputs
        movlw       b'10000000'
        movwf       ADCON1              ;Right justified, all A/D
        movlw       b'00000000'
        movwf       OPTION_REG
        bcf         STATUS,RP0      ;Bank 0
        movlw       B'01011001'     ;Fosc/8 [7-6], A/D ch3 [5-3], A/D on [0]
        movwf       ADCON0
        clrf        PORTB
        clrf        cnt3
        clrf        num_times
        movlw        d'8'
        movwf       num_samples
```

```
Main
        call    ad_portb
        goto    Main


ad_portb
                                        ;wait for acquision time (20uS)
          bsf     ADCON0,GO             ;Start A/D conversion
Wait
        btfsc   ADCON0,GO               ;Wait for conversion to complete
        goto    Wait


;tilt error value = (ADC(x)-3.0)

comp_ute
                        movwf           Arg_lo
                        movf            ADRESH,w ; Store in ADC 10-bit result
                        movwf           Arg_hi          ; register
                        movf            temp_acchi,w
                        movwf           Sum_hi
                        movf            temp_acclo,w
                        movwf           Sum_lo

                        call            add
                        movf            Sum_hi,w
                        movwf           gy_calc_angle_new_hi
                        movf            Sum_lo,w
                        movwf           gy_calc_angle_new_lo
                        decfsz          num_samples
                        call            wait
                        movlw           d'8'                    ;Averaging of 8 samples
                        movwf           num_samples

                        movf            gy_calc_angle_new_hi,w
                        movwf           Arg_hi
                        movf            gy_calc_angle_new_lo,w
                        movwf           Arg_lo
                        movlw           b'00000010'
                        movwf           Sum_hi
                        movlw           b'00000001'
                        movwf           Sum_lo
                        call            substract
                        movf            Sum_hi,w
                        movwf           tilt_temp_hi
                        movf            tilt_temp_hi,w
```

```
movwf       Arg_hi
movf        Sum_lo,w
movwf       tilt_temp_lo
movf        tilt_temp_lo,w
movwf       Arg_lo

movlw       b'00000000'
movwf       Kt_tilt_hi
movf        Kt_tilt_hi,w
movwf       var_hi
movlw       b'00000001'
movwf       Kt_tilt_lo
movf        Kt_tilt_lo,w
movwf       var_lo
call        mul
movf        prod_res1,w
movwf       tilt_hi
movf        prod_res0,w
movwf       tilt_lo


movlw       b'00000000'
movwf       Kv_tilt_hi
movlw       b'00000001'
movwf       Kv_tilt_lo
movf        Kv_tilt_hi,w
movwf       Arg_hi
movf        Kv_tilt_lo,w
movwf       Arg_lo
movf        tilt_hi,w
    movwf       var_hi
movf        tilt_lo,w
    movwf       var_lo
call        mul
movf        prod_res1,w
movwf       u_term_hi
movf        prod_res0,w
movwf       u_term_lo

movf        ADRESH,w
sublw       b'00000010' ; Upper byte for analog value 3V
call        no_drive
movwf       temp_lo
btfsc       STATUS,c
call        ccw
call        cw
```

The following program is similar to the list of program from pg 94, no-drive subroutine to pg 109 .

Multiplication program

```
Mul
            ;movlw              b'00010010'
            ;movwf              Arg_lo
            ;movlw              b'00011000'
            ;movwf              Arg_hi
            ;movlw              b'10010100'
            ;movwf              var_lo
            ;movlw              b'01110000'
            ;movwf              var_hi
            clrf        prod_res0
            clrf        prod_res1
            clrf        prod_res2
            clrf        prod_res3
            clrf        count
            movlw       d'17'
            movwf       count

            movf        var_lo,w              ; Place the value from other
program                                                   ;loop
movwf       prod_res0           ;into the variable of this
                                ;multiplication loop.
            btfss       status,z      ;Check whether the product_res0 is zero
            call        Check_n
            movf        var_hi,w              ;the product_res0 is zero
            movwf       prod_res1             ;Check the next upper byte.
            btfsc       status,z
            call        equal_zero

Check_n     btfss       status,c
            Call        add_var
            movf        var_hi,w
            Movwf       prod_res1


Check_Arg   movf        Arg_Lo,f
            btfss       status,z      ;Test if value of Arg_Lo is zero
```

```
                call        test_lsb            ;Arg_lo is not zero
                movf        Arg_Hi,w            ;Arg_lo is zero
                btfsc       status,z        ;Test if value of Arg_Hi is zero
                call        equal_zero          ;Arg_Hi is equal to zero
                call    test_lsb

add_var         incf    prod_res1
                movf    var_hi,w
                movwf   prod_res1

test_lsb        bcf         status,c
                rrf         prod_res3
                rrf         prod_res2
                rrf         prod_res1
                rrf         prod_res0
                btfss       status,c        ;Test if there is carry bit
                call        shift               ; There is no carry
                movf        Arg_lo,w        ;Upper two bytes is added with the bytes
                addwf       prod_res2,f         ;of the Arg_Hi:Arg_Lo.
                Btfsc       status,c
                call        Add_Hi
                movf        Arg_Hi,w
                addwf       prod_res3,f
                call        shift

equal_zero      clrf        prod_res0       ; the multiplication result is zero.
                clrf        prod_res1
                clrf        prod_res2
                clrf        prod_res3
                call        stop                ;Exit from the mul subroutine

Add_Hi          incf        prod_res3
                Movf        Arg_Hi,w
                Addwf       prod_res3


shift           decfsz      count
                call        test_lsb
                goto        stop

stop clrf       count
```

Division program

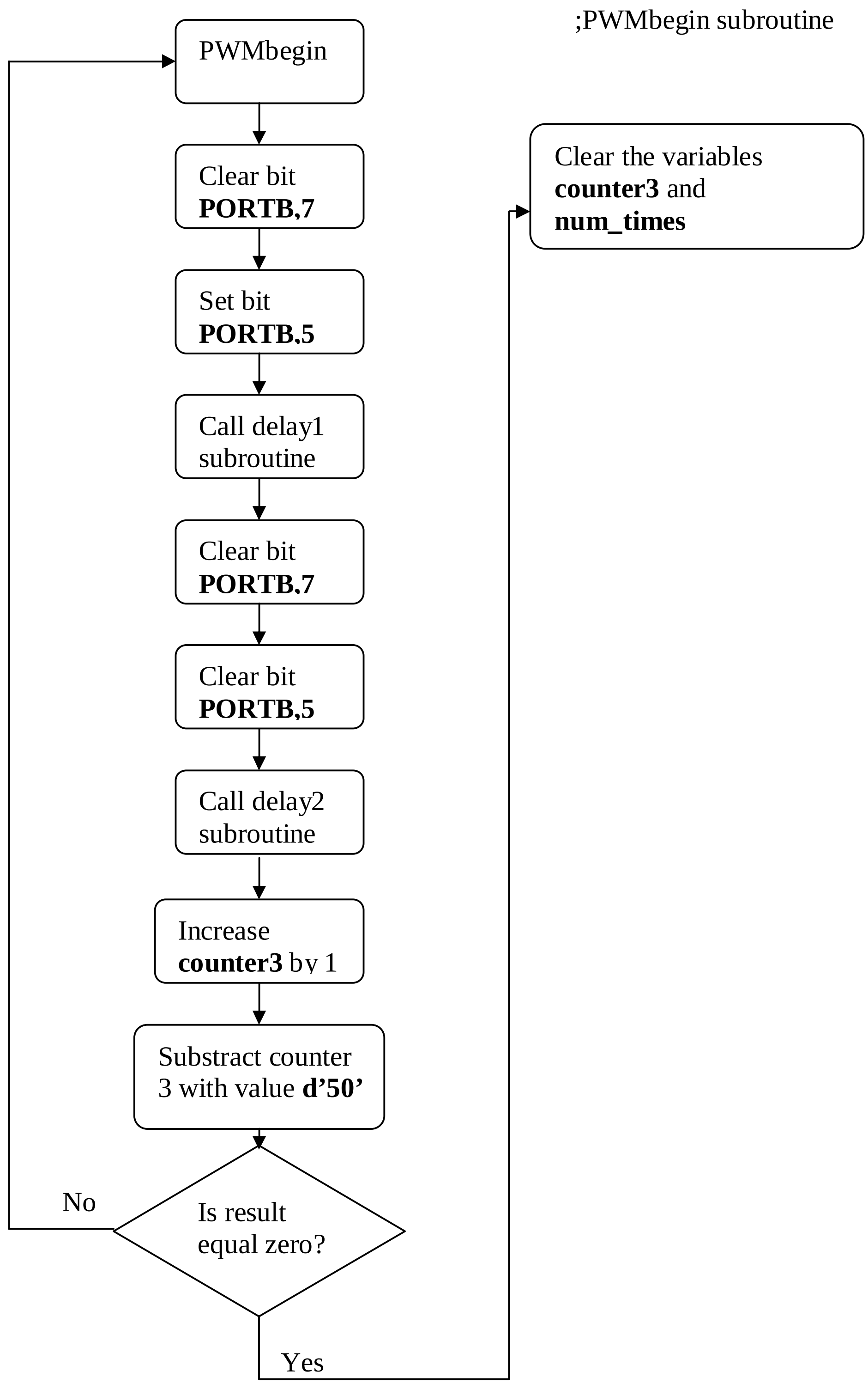
```
Division
        ;movlw   b'00010010'   ;Inserting the values into divisor and the
        ;movwf   divisor0      ;number to be divided.
        ;movlw   b'00011000'
        ;movwf   divisor1
        ;movlw   b'00111000'
        ;movwf   divisor2
        ;movlw   b'01001000'
        ;movwf   divisor3
        ;movlw   b'10010100'
        ;movwf   prod_res0
        ;movlw   b'01110000'
        ;movwf   prod_res1
        ;movlw   b'00000100'
        ;movwf   prod_res2
        ;movlw   b'11100000'
        ;movwf   prod_res3

Clrf    rem_dr0  ;Initialise variables by clearing all the values
        Clrf     rem_dr1
        clrf     rem_dr2
        clrf     rem_dr3
        clrf     Quo_0
        clrf     Quo_1
        clrf     Quo_2
        clrf     Quo_3
        Movlw    32
        Movwf    bitcnt



Loop    rlf      prod_res0     ;Clear the 32 bit result registers
        rlf      prod_res1
        rlf      prod_res2
        rlf      prod_res3
        rlf      rem_dr0
        rlf      rem_dr1
        rlf      rem_dr2
        rlf      rem_dr3
```

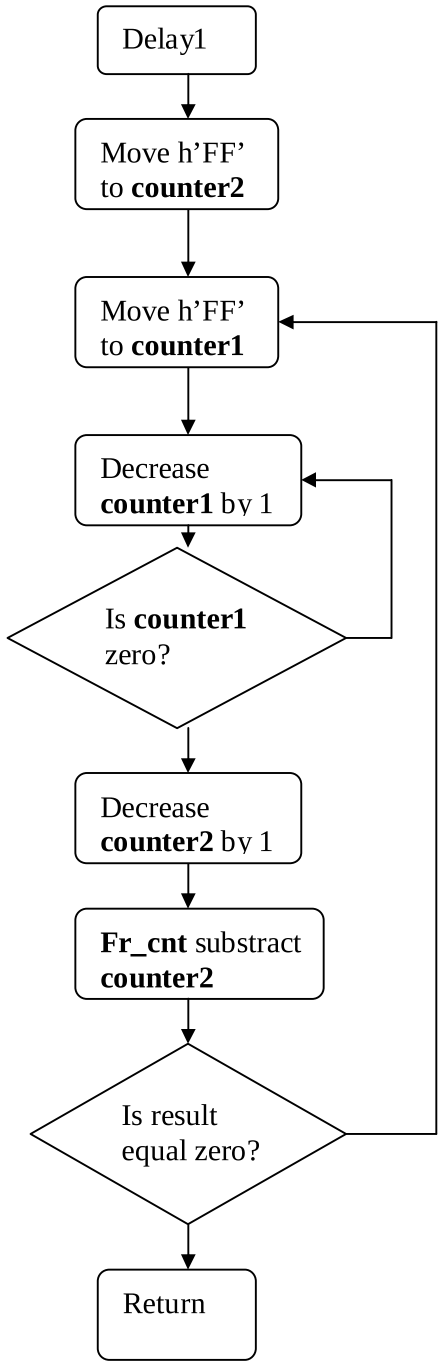
```
movf        divisor3,w    ;Compare divisor and remainder
            subwf         rem_dr3,w
            btfss         STATUS,z
            call          test_more3    ;test if rem_dr3 is more than divisor if not


equal       movf          divisor2,w    ;High byte is equal, so test the lower byte
            subwf         rem_dr2,w
            btfss         STATUS,z
            call          test_more2
            movf          divisor1,w    ;Test the lower bytes
            subwf         rem_dr1,w
            btfss         STATUS,z
            call          test_more1
            movf          divisor0
            subwf         rem_dr0,w
            call          test_more0

;These sections are to test the result byte whether the divisor value is
;larger or smaller than the rem_dr value.
;If the rem_dr value is larger then the divisor then goto subs subroutine

test_more3  btfss         STATUS,c
            call          last
            call          subs2

test_more2  btfss         STATUS,c
            call          last
            call          subs1

test_more1  btfss         STATUS,c
            call          last
            call          subs0

test_more0  btfss         STATUS,c
            call          last
            bsf           prod_res0,0
            call          last

;These sections subs2, subs1 and subs3 are to compare
;the adjacent bytes on whether there is any carry bits.
;If there is any carry, adjust affected bytes by decresing one
;bit value
```

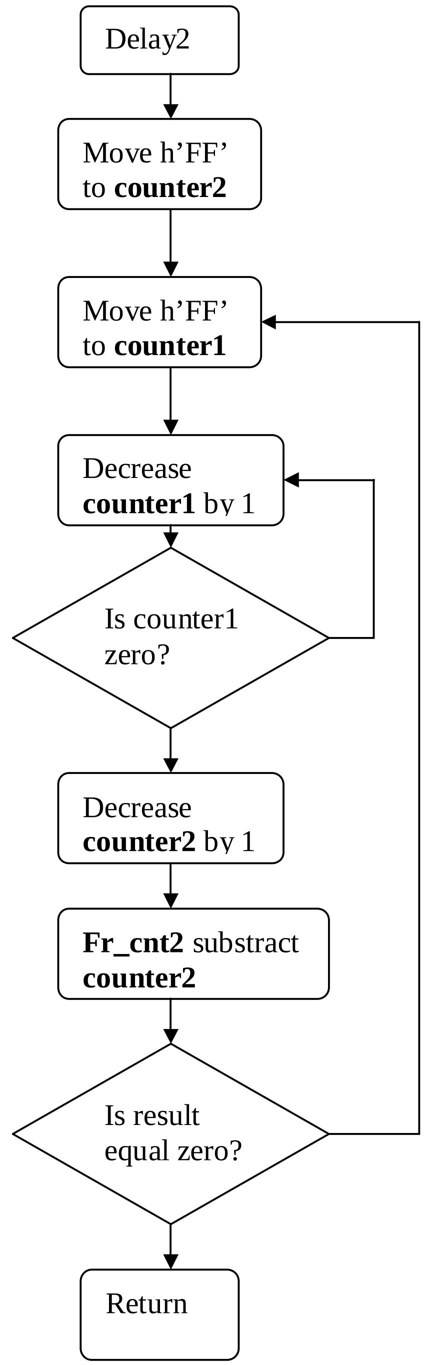
```
subs2       movf        divisor2,w
            subwf       rem_dr2,w
            btfss       STATUS,c
            decf        rem_dr3
            movf        divisor3,w
            subwf       rem_dr3,w
            movf        divisor1,w
            subwf       rem_dr1,w
            btfss       STATUS,c
            decf        rem_dr2
            movf        divisor2,w
            subwf       rem_dr2,w
            movf        divisor0,w
            subwf       rem_dr0,w
            btfss       STATUS,c
            decf        rem_dr1
            movf        divisor1,w
            subwf       rem_dr1,w
            bsf         prod_res0,0
            call        last

subs1       movf        divisor1,w
            subwf       rem_dr1,w
            btfss       STATUS,c
            decf        rem_dr2
            movf        divisor2,w
            subwf       rem_dr2,w
            movf        divisor0,w
            subwf       rem_dr0,w
            btfss       STATUS,c
            decf        rem_dr1
            movf        divisor1,w
            subwf       rem_dr1,w
            bsf         prod_res0,0
            call        last

subs0       movf        divisor0,w
            subwf       rem_dr0,w
            btfss       STATUS,c
            decf        rem_dr1
            movf        divisor1,w
            subwf       rem_dr1,w
            bsf         prod_res0,0
            call        last
```

```
last    decfsz  bitcnt          ;Test whether 32 rotations has been
        Goto    loop            ;executed
        Movf    prod_res0,w
        Movwf   Quo_0           ;Result is stored in Quo variables
        Movf    prod_res1,w             ;The result is 32 bits values
        Movwf   Quo_1
        Movf    prod_res2,w
        Movwf   Quo_2
        Movf    prod_res3,w
        Movwf   Quo_3

        End
```

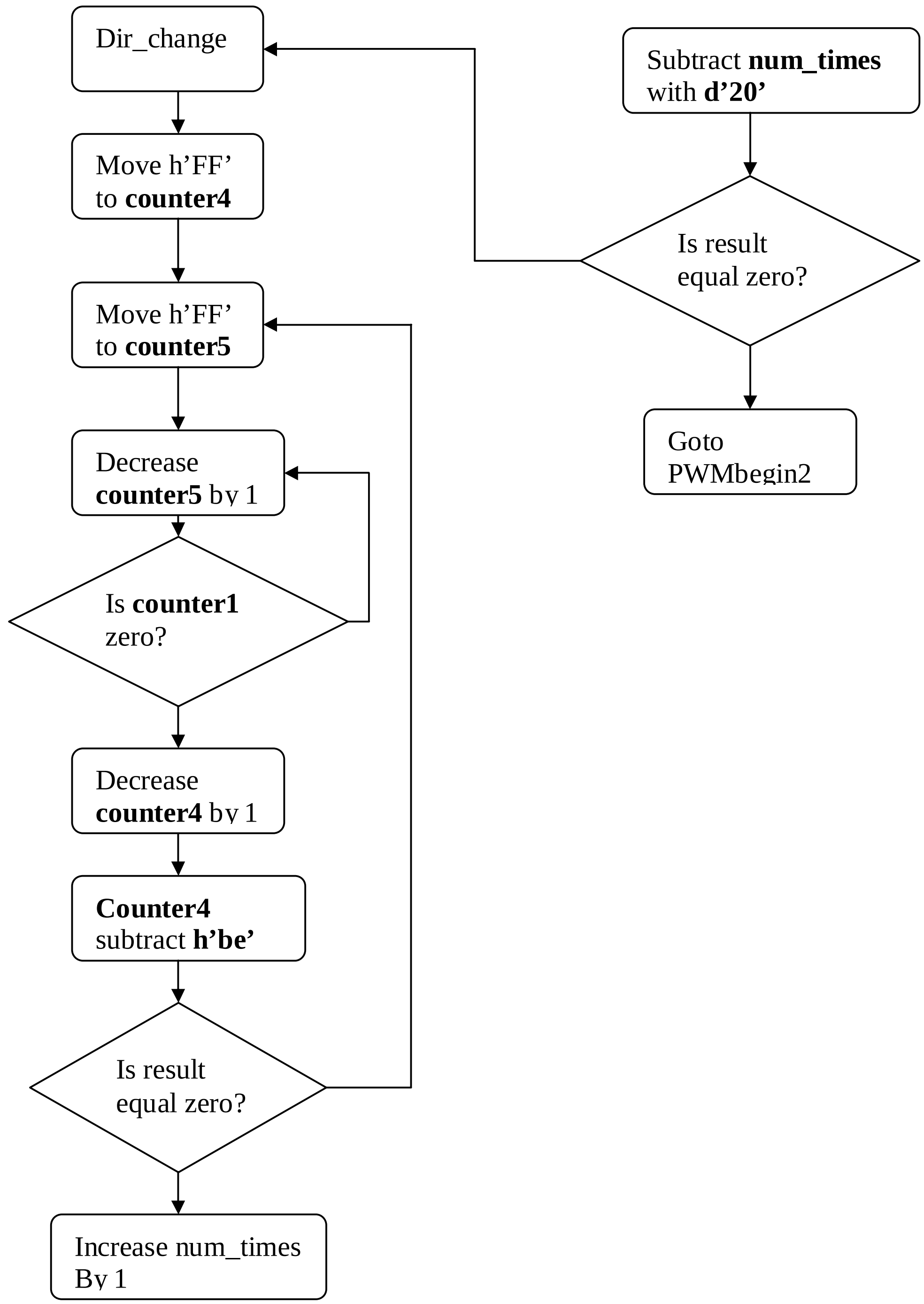
Programming flow-chart (On h-bridge (bi-direction motor turn))



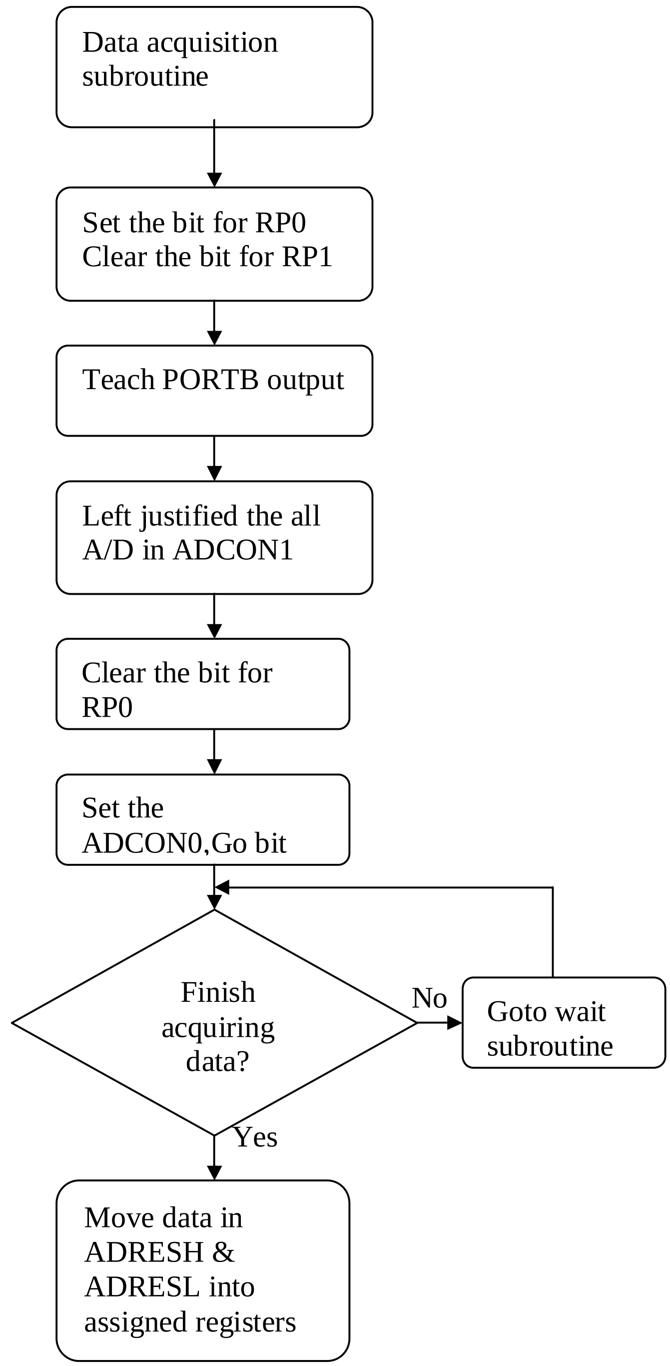

Delay1
Move h'FF' to **counter2**
Move h'FF' to **counter1**
Decrease **counter1** by 1
Is **counter1** zero?
Decrease **counter2** by 1
Fr_cnt substract **counter2**
Is result equal zero?
Return


Delay2
Move h'FF' to **counter2**
Move h'FF' to **counter1**
Decrease **counter1** by 1
Is counter1 zero?
Decrease **counter2** by 1
Fr_cnt2 substract **counter2**
Is result equal zero?
Return

Direction change subroutine



Data Acqusition flow chart



Gyro-Accelerometer

```
;If one of the portc,2 exhibit a 1 signal then use the accelerometer data
;If portc,2 exhibit a 0 signal then use the gyro data
;Both the accelerometer and gyro run simulaneously
;Refer to the CD for more details


Data_chg        btfsc       PORTC,2
                call        acce_data
                call        gyro_data

acce_data       movf        u_term_lo_acce,w
                movwf       u_term_lo
                movf        u_term_hi_acce,w
                movwf       u_term_hi
                return

gyro_data       movf        u_term_lo_gyro,w
                movwf       u_term_lo
                movf        u_term_hi_gyro,w
                movwf       u_term_hi
                return

                end
```