

# Major Project Report

*by* Daniel Nilsen

---

|                |                                                                        |                 |       |
|----------------|------------------------------------------------------------------------|-----------------|-------|
| FILE           | 14288_DANIEL_NILSEN_MAJOR_PROJECT_REPORT_78227_1673776900.DOCX (6.14M) |                 |       |
| TIME SUBMITTED | 21-APR-2016 01:11PM                                                    | WORD COUNT      | 11016 |
| SUBMISSION ID  | 56194201                                                               | CHARACTER COUNT | 56240 |



**Prifysgol Cymru**  
Y Drindod Dewi Sant  
**University of Wales**  
Trinity Saint David

# **A Study of self-balancing robot using PIC Microcontroller**

by

Daniel Nilsen

Supervisor: Mr. Gareth Jones

Second Marker: Dr. John Rees

Project submitted as part of the requirements for the  
award of BSc (Hons) Computer Networks

April 2016

---

## DECLARATION

I, Daniel Nilsen declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

*Signed* : .....

*Date* : .....

---

## FORM OF CONSENT

I Daniel Nilsen hereby consent that my Project, submitted in candidature for the Computer systems & Electronics degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

**Signed :** .....

**Date :** .....

---

# Abstract

This project aims to achieve a working fully autonomous self balancing robot with the use of a PIC microcontroller. You are required to present the different design methods used to achieve this end goal. The report explains why the design choices taken was made and how the different methods aided to solve problems that came up. The project is driven by the interest of robot technology with fundamental electronic and software knowledge and have been simulated and built to prove that the initial designs and goals was achieved.

There are testing within the simulation for specific building blocks to prove that the intended prototype would function within the parameters set originally for this project.

---

# Table of Contents

|                                                            |                         |
|------------------------------------------------------------|-------------------------|
| <b>Chapter I: Introduction</b>                             | <b>Page 7</b>           |
| I.I - Introduction                                         | Page 7                  |
| <br><b>Chapter II: Research &amp; Review of Literature</b> | <br><b>Page 8 - 13</b>  |
| II.I - Similar Projects/Products                           | Page 8 - 10             |
| II.I.I - Segway                                            | Page 8 - 9              |
| II.I.II - nBot                                             | Page 9                  |
| II.II - Motion Algorithm                                   | Page 11 - 12            |
| II.III - Inverted Pendulum Theory                          | Page 12 - 13            |
| II.IV - Automatic Stability                                | Page 13                 |
| <br><b>Chapter III: Research Methodology</b>               | <br><b>Page 14 - 18</b> |
| III.I - Research Onion                                     | Page 14 - 17            |
| III.II - Software & Hardware                               | Page 18                 |
| <br><b>Chapter IV: Design</b>                              | <br><b>Page 19 - 38</b> |
| IV.I - Design Process                                      | Page 19 - 20            |
| IV.I.I - Vehicle Stability                                 | Page 19                 |
| IV.I.II - Control System                                   | Page 20                 |
| IV.I.III - System Response                                 | Page 20                 |
| IV.II - Advanced Block Diagram                             | Page 21                 |
| IV.III - Vehicle Stability                                 | Page 21 - 26            |
| IV.III.I - Accelerometer                                   | Page 21 - 23            |
| IV.II.II - Gyroscope                                       | Page 23 - 24            |
| IV.II.III - Kalman & Complementary Filter                  | Page 24 - 26            |
| IV.II.IV - PID Controller                                  | Page 26                 |
| IV.IV - Control System                                     | Page 26 - 31            |
| IV.III.I - PIC Microcontroller                             | Page 26 - 31            |
| IV.V - System Response                                     | Page 32 - 34            |
| IV.IV.I - Motors                                           | Page 32                 |
| IV.IV.II - Wheels                                          | Page 32                 |
| IV.IV.III - Motion Control                                 | Page 32 - 34            |
| IV.VI - Voltage Regulator                                  | Page 34 - 35            |
| IV.VII - LCD Display                                       | Page 35 - 38            |
| <br><b>Chapter V: Implementation</b>                       | <br><b>Page 39 - 50</b> |
| V.I - Circuit Diagram                                      | Page 39 - 44            |
| V.II - Software Implementation                             | Page 45 - 48            |
| V.III - Software Flowchart                                 | Page 49 - 50            |

---

|                                                    |                     |
|----------------------------------------------------|---------------------|
| <b>Chapter VI: Evaluation</b>                      | <b>Page 51 - 53</b> |
| VI.I - Objectives & Aims Outcome                   | Page 51             |
| VI.II - Research Review                            | Page 51 - 52        |
| VI.III - Future Improvements                       | Page 52 - 53        |
| <br>                                               |                     |
| <b>Chapter VII: Conclusion and Recommendations</b> | <b>Page 54</b>      |
| VII.I - Summary                                    | Page 54             |
| <br>                                               |                     |
| <b>Chapter VIII: Reflection</b>                    | <b>Page 55</b>      |
| VIII.I - Gained Knowledge & Experience             | Page 55             |
| <br>                                               |                     |
| <b>Chapter IX - References</b>                     | <b>Page 56 -57</b>  |
| <br>                                               |                     |
| <b>Chapter X - Bibliography</b>                    | <b>Page 58 - 59</b> |
| <br>                                               |                     |
| <b>Chapter XI – Appendices</b>                     | <b>Page 60 - 61</b> |

---

# Table of Figures

**Figure 2.1 - Segway Model**

**Figure 2.2 - nBot**

**Figure 2.3 - PID Controller Algorithm**

**Figure 2.4 - PID Block Diagram**

**Figure 2.5 - Inverted Pendulum Illustration**

**Figure 3.1 - Illustration of the different layers**

**Figure 4.1 - Block Diagram**

**Figure 4.2 - Advanced Block Diagram**

**Figure 4.3 - ADXL335 Block Diagram**

**Figure 4.4 - Kalman Filter Illustration**

**Figure 4.5 - Kalman Filter Terms**

**Figure 4.6 - External Crystal Circuit**

**Figure 4.7 - H-Bridge Circuit**

**Figure 4.8 - Voltage Regulator Circuit**

**Figure 4.9 - LCD Block Diagram**

**Figure 4.10 - LCD Pin Configuration**

**Figure 4.11 - LCD Circuit Diagram**

**Figure 5.1 - Project Circuit Diagram**

**Figure 5.2 - Project Simulation (1)**

**Figure 5.3 - Project Simulation (2)**

**Figure 5.4 - Project Simulation (3)**

**Figure 5.5 - Circuit on Breadboard (1)**

**Figure 5.6 - LCD display**

**Figure 5.7 - PIC16F887**

**Figure 5.8 - ADXL335**

**Figure 5.9 - Circuit on Breadboard (2)**

**Figure 5.10 - Project Code Sample (1)**

**Figure 5.11 - Project Code Sample (2)**

---

**Figure 5.12 - Project Code Sample (3)**

**Figure 5.13 - Project Code Sample (4)**

**Figure 5.14 - System Flowchart**

---

# Chapter I - Introduction

## Chapter I.I - Introduction

Since technology really started to progress at a rapid rate, the robot industry have been improving drastically over the past 10 years. You start to see how this technology start to implement itself into our everyday life. It have always been a thrive to develop a machine that could mimic and replicate human thought and behaviour. This is what peaked the interest in doing a major project around robot technology, and try to achieve something that would be complex enough to really delve into, but at the same time achievable by a third year electronic student.

The main objective of this project is to gather knowledge about and around a self balancing robot, and to understand and design an effective and efficient control system so that the robot could achieve self balance automatically. This will all be done making the use of previous experience with PIC microcontrollers and with fundamental electronic principles.

The following chapters will include the process around how the information that was needed for this project was gathered and the different methods used to aid the implementation of this new acquired knowledge.

---

# Chapter II - Research & Review of Literature

During this chapter there will be a reflection on the information gathered and give an overview of the decisions that was made during the project period. There will be sections focused on each part required to complete this project as well as similar products or projects with relevant information. This chapter focuses on bringing a broader understanding to this project with information gathered from online sources, thesis', reports and journals published on either the internet or in written format.

## **Chapter II.IV - Similar Projects/Products**

When researching into self balancing robots there are several balancing projects or products that would help understanding key aspects and building blocks common for all balancing project and there will be some mentioned during this sub-chapter.

### **Segway**

The Segway is a new type of human transportation, and was called "The world's first self balancing human transportation", by Dean Kamen [1] the inventor of this product. The product has only two wheels, just like the bike, scooter or wheelchair and is focused on the rider using his/her movements to control it. The rider leans backward or forward to make the Segway move accordingly, and by moving the handlebar either left or right corresponds in it moving either left or right. The Segway has two wheels, a motor, a collection of microprocessors and a series of tilt sensors in order to control the self balancing aspect of this product. You could say that the Segway is controlled by three systems, motor, sensor and control.

---

The sensor system is built up of gyroscopes which is a device that resists changes to its axis of rotation and mimic the balancing of a human being. Because of its resistance to outside forces the gyroscope wheel will leave its position in space with its frame being able to operate freely, even if you tilt it. The Segway is a complex machine and a standard regular gyroscope would not be as efficient in this type of product, so a solid-state angular rate sensor is what controls the balance, and is using the Coriolis effect on a small scale.

The Coriolis effect is states as:

*“An effect whereby a mass moving in a rotating system experiences a force (the Coriolis force) acting perpendicular to the direction of motion and to the axis of rotation” [2].*

This solid-state gyroscope is supported by a silicon frame where the silicon particles are moved using an electrostatic current applied across the plates.



*Figure 2.1 - Segway Model*

Whenever there is a change ie. the Segway moves there is an alteration in vibration in the plates, this change is monitored and sent on to the computer which can then figure out if the computer is changing in direction or rotating along particular axes. The information is then sent to the motors to either correspond to the change or operate against it to achieve the self balancing aspect.

---

The reason this is of interest in this project is that the original aim was to create this on a small scale but having a robot stand in a 90 degree angle without the need of external aid.

### **nBot**

David Anderson is the creator behind a self balancing robot called nBot. It is a two-wheeled robot that utilize a gyroscope and accelerometer with its outputs fused together by a Kalman filter. The Kalman filter is what provides this self balancing robot with the accurate input control for the stability. It has been revised and improved several times with better accuracy, stability and features.



*Figure 2.2 - nBot*

## **Chapter II.II - Motion Algorithm**

With all the information gathered from the sensors there are certain problems that needs to be addressed. There are several mentions in journals and reports about certain controllers needed to achieve the robot to be self balancing. One of the mentioned methods are called a PID controller, or proportional-derivative controller. This is a control loop feedback mechanism used in many different control systems, which calculates error values between a desired setpoint chosen by the user and the measured value coming from the sensors. The controllers job is to minimize the error value of time to achieve the desired setpoint, which in this case would be a robot to be self balancing around a 90 degree angle.

---

A PID controller is built up of three coefficients  $K_p$ ,  $K_i$ , and  $K_d$  each correspond to the three terms, proportional, integral and derivative. Given the algorithm:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Figure 2.3 - PID Controller Algorithm

This is a block diagram showing the feedback loop in a PID controller.

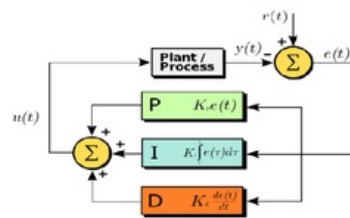


Figure 2.4 - PID Block Diagram

Where the coefficients in this model goes like this:

- “P accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
- I accounts for past values of the error. For example, if the current output is not sufficiently strong, error will accumulate over time, and the controller will respond by applying a stronger action.
- D accounts for possible future values of the error, based on its current rate of change”[3].

Some systems might not need every term in a PID controller, since every term has its own property you might just need one or two and this is done by setting the parameters you don’t need to zero, it will then be called either P, PI, PD, P or I controller in the absence of the respective controller present.

---

## Chapter II.III - Inverted Pendulum Theory

When you want to develop and design a self balancing robot it is essential to research and get an understanding of all relevant parameters within the system. One of the parameters coming up during the research was Inverted Pendulum Theory or more traditionally known as Pole and Cart theory. The theory states that forces must be properly applied to keep a system stable and is appropriate for this type of project. In simple terms you could picture yourself trying to balance a long broomstick on your finger, and to keep it in an upright position you would have to move your finger in a direction to prevent the broomstick from falling over. This is what we aim to achieve with our self balancing robot.

The inverted pendulum theory goes hand in hand with the PID control law and have been of great importance when it comes to the topic of control engineering during this project. The ability to maintain a vertical stability is one of the biggest importance of this project. The purpose of the inverted pendulum theory is to generalize the left and right wheels to work in unison to maintain vertical stability of the robot. You want to have the wheels beneath the robot to be at the center of the robot chassis's mass and if the robot starts to tilt either forward or backward you will have to compensate this effect by powering the wheels to move in opposite direction.

The following figure shows an inverted pendulum, which is the basis of the theory.

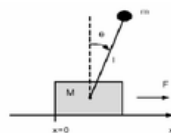


*Figure 2.5 - Inverted Pendulum Illustration*

---

## **Chapter II.IV - Automatic Stability**

For this project to achieve its original goal you will need to design an automatic stability control system in order to have the robot be self balancing without any external aid. You are required to have the robot be balanced in an upright, preferably a 90 degree angle position at all time, and you want the balancing to be autonomous. The inverted pendulum theory is what provides the equations required to maintain this stability, and it is necessary to apply this control system as effective and efficient within a small timeframe from when the sensors detect change to the motors prevent this change.

The project have the aim to use a PIC microcontroller to maintain the control systems and achieve autonomous stability.

---

# Chapter III - Research Methodology

This chapter is all about the different methods every researcher uses to answer different questions about how to succeed and successfully achieve what you first aim to do in a project. There are many methods, but there will be mentioning about what methods was used to get the results during this specific project.

## Chapter III.I - Research Onion

Whenever you want to read up on different research methods there is one common theme that comes up, and that is the research onion[4]. The research onion was developed by *Saunders et al(2007)* for the sole purpose of describe the different stages which a researcher must pass when they want to formulate a precise and effective research methodology.

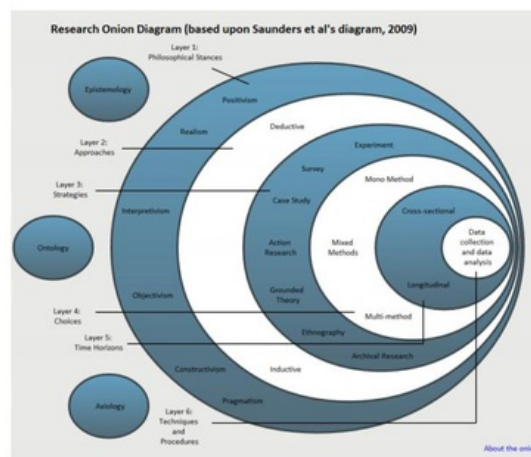


Figure 3.1 - Illustration of the different layers

- Layer 1: Philosophical Stances
- Layer 2: Approaches
- Layer 3: Strategies

- 
- Layer 4: Choices
  - Layer 5: Time Horizons
  - Layer 6: Techniques and Procedures

In each layer there are different approaches that the researcher needs to take to go further down the layers. By doing layer by layer you should achieve a well prepared and suited research methodology towards your end results.

### **Layer 1: Positivism**

In the first layer the approach taken is called Positivism, which generates a hypothesis, or a research question or questions that can be tested which would allow a explanation measured against the accepted and common knowledge in the field which the project is developed. This specific method creates a body of research that can be useful to other researchers who would have a similar research topic or question so they could generate a similar or the same result. This method gives the researcher quantifiable results that can be statistically analysed.

During this project this method was used on the background of the researcher generating his own hypothesis and answering the research questions given at the beginning. The different hypothesis and questions would be answered by testing different procedures and generate a analysis of these results.

### **Layer 2: Deductive**

The second layer methods is called Deductive research, which means that when you start a project you have a statement or question that you would want to answer during the project period. You should aim to at the end of the project either have a yes or no to your stated question/statement. Questions can be a statement or pure speculation around the researched topic that you would believe can be answered through a methodical research process.

The process behind deduction is that you move from a theory, to the statement/question you set/ask yourself, before moving on to the data collection where at the end you would have the finding to either reject or

---

confirm the initial statement/question. At this point your conclusion would either let you revise the theory to improve, or even make you start your process all over again. This process is extremely relevant for this project as it let's you move from a simple question/statement towards results that would either answer or disprove your initial question/statement.

### **Layer 3: Ground Theory & Experiment**

The third layer lets you decide on a strategy to take to get towards your end result, which for this project was the ground theory and experiment. Ground theory is when you use inductive methods to either predict or/and explain a theory you might have before starting the project. This usually involves the researcher gathering data from several observations, so that the ground theory can be generated from that data and then tested to either reject or confirm that ground theory. This is used during this project as a method to improve or reinvent several aspects of the project and is backed up by previous theory and literature around the topic at hand. An example is the data gathered from the accelerometer to let the robot be self balanced.

The other strategy that was used from this layer is the experiment method. This is based around a more rigid and scientific structure so that the researcher will be able to replicate the designs for testing. You test the design to analyse and improve the project so that the end results does support your initial statement/aim.

### **Layer 4: Mixed Methods**

This type of research method is when you use both quantitative and qualitative research methods during the process of your study, data collection and analysis. This was the method that would fit the best over the two other different research methods, as you would both have a large number of qualitative research as well as some quantitative research.

### **Layer 5: Cross-Sectional**

This layer just state the two form of time horizons a project would endure, with them being either long term, or short term. This project uses a short term method, as you would gather the data in real time for the project to fulfill your stated aims and objectives.

---

**Layer 6: Data Collection & Data Analysis**

The final layer of the research onion includes the decisions on your sampled groups, questionnaire content, and questions that would be asked in interviews to aid the researcher in achieving the end results. For this specific project there are no sampled groups, questionnaires or interviews that will be done to achieve the results set. As there are just a product and whether or not it achieve the balancing aspect that would determine if its a success or not. What is an accelerometer and what should you be using at certain times?

## **Chapter III.II - Software & Hardware**

For many project there is certain requirements when trying to implement the different design/research methods. And to aid you with this is practical hardware and software that would aid you as a designer on the implementation of your research to achieve your aims and objectives. For this particular project there will be used these types of hardware and software:

### **Software**

- Proteus Simulation 8
- MPLAB
- CCS C
- PicKit
- Internet Browser
- Online Library Resource Software

### **Hardware**

- Breadboard
- Soldering Iron
- Ammeter/Voltmeter

---

# Chapter IV - Design

Within this chapter there will be a discussion around the key aspects of completing this project, along with the obstacles during the design and how to overcome them. The chapter will be divided into a series of sub-chapter where each discuss and talk about the appropriate areas that was focused during the design.

## Chapter IV.I - Design Process

The initial idea to this project was to design a self balancing robot with a PIC microcontroller using an accelerometer to gather information needed to keep the algorithms and control systems fed with relevant information in real time. This would allow the robot to achieve a self balancing stability using an autonomous system. The microcontroller is what would provide the computational power so that the robot could balance itself, with the given information gathered from the sensors. Given the complexity of this project regarding the balancing aspect the PIC microcontroller and the accelerometer is required to handle data at an extreme speed and accuracy. These are the three biggest building blocks that needs to work together in order to achieve the aims set out for this project with a brief explanation to why this is needed.

### Vehicle Stability - Accelerometer & PID Controller

- This is what provides the information of where the robot is in space, the accelerometer is providing the inputs to the control systems
- It measures where the robot is on a x-,y-, and z-axis
- Can be either digital or analog
- Uses the information from accelerometers to create a feedback loop to prevent tilt and reducing error
- Gyroscope can also be implemented to increase accuracy of balance and minimize error
- Filter required if gyroscope and accelerometer is used to incorporate both signals

---

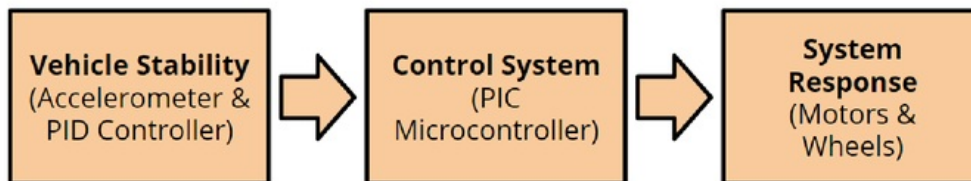
### Control System - PIC microcontroller

- This is the brain of the project, where the information gathered by the above sensors gets used to control the stability of the robot
- It needs to be quick as there are a limited timeframe of which the errors needs to be corrected before the stability of the robots is critical(e.g, falling over)
- Needs to interface with the relevant sensors and also able to provide outputs to control external sources

### System Response - Motors & Wheels

- Motors capable of driving the robots chassis and maintain speed given from the outputs of the control system
- Key aspect of generating the opposite force needed to prevent tilt errors being too large

Below in Figure 4.1 there is shown a standard block diagram to illustrate the three main areas that was first



thought of in this project:

*Figure 4.1 - Block Diagram*

---

## Chapter IV.II - Advanced Block Diagram

Below is a more advanced block diagram that created from the initial block diagram design. This shows in more detail what the aims and objectives with this project is trying to achieve, with both achievable and future features in mind.

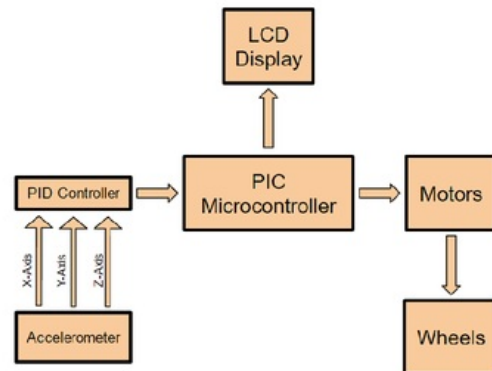


*Figure 4.2 - Advanced Block Diagram*

## Chapter IV.III - Vehicle Stability

### Accelerometer

When designing a self balancing robot you are required to gather certain variables in order to achieve that aspect. You are required to access variables from the accelerometer in order for the control system to act accordingly. The accelerometer will give you information of where it is along three different axis, x-,y-, and z-axis. You will get an estimate of the velocity and displacement of the accelerometer, and if placed in a suitable place on the chassis correspond to the placement of the robot on these three axis. This is a critical part of the project since it is the information given from this accelerometer that would make the robot able to counteract and balance itself on the basis of this information.

Accelerometers works in many different ways, but the most common type among accelerometers is

---

with the use of the piezoelectric effect, where the chip contains microscopic crystal structures that gets stressed by accelerative forces, which again causes some sort of voltage to be generated. Depending on what type of accelerometer you are using, with the two most common being analog or digital you can use this voltage value to represent where the chip is located on the three different axis. How the voltage corresponds to where in space the chip is can be found on the datasheet of your specific chip.

When deciding on an accelerometer you first have to determine whether or not you would like to go analog or digital and understand the difference between them two. There are advantages and disadvantages to both types. When deciding on an analog or digital is mostly up to how you are planning to interfacing the accelerometer and if the connections between them two parts require the analog or digital outputs. The main difference between an analog and a digital accelerometer is how the output signals are generated. With an analog accelerometer the outputs are a continuous generated voltage that is proportional to acceleration, e.g. 2.5V for 0g, 2.6V for 0.5g, or 2.7V for 1g. A digital accelerometer usually use pulse width modulation (PWM) to generate a square wave where the amount of time the voltage is high corresponds to the amount of acceleration.

For this project an analog accelerometer called ADXL335[5] was chosen due to the fact that we are operating with analog inputs on the control system. The interfacing with the accelerometer and the control system would therefore be easier than having to deal with a digital outputs coming from a digital accelerometer. The reason behind choosing this accelerometer was its availability, price and several references in similar projects. The ADXL335 is a small, low-powered analog 3-axis accelerometer with output voltage to measure acceleration with a minimum full-scale range from -3g to +3g. It can measure the static acceleration of gravity and is heavily used in tilt-sensing applications such as self balancing projects. You as a designer can select the bandwidth the accelerometer uses by changing the capacitors on the Cx, Cy, and Cz outputs. The range of capacitors you can choose is described in its datasheet, and states that on the X- and Y-axis pins a range of 0.5 Hz to 1600 Hz is most acceptable, where for the Z-axis a range of 0.5 Hz to 550 Hz is acceptable. The accelerometer have a low voltage range, with a single-supply operation from 1.8 V to 3.6V, which means that a 3.3V supply would suffice to power this accelerometer.

---

This is a functional block diagram taken from the datasheet of the ADXL335:

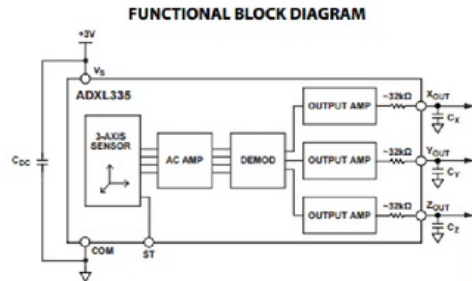


Figure 4.3 - ADXL335 Block Diagram

## Gyroscope

“Hypothesis: The earth, having once received a rotational movement around an axis, which agrees with its axis on the figure or only differs from it slightly, will always conserve this uniform movement, and its axis of rotation will always remain the same and will be directed toward the same points of the sky, unless the earth should be subjected to external forces which might cause some change either in the speed of rotational movement or in the position of the axis of rotation.” [6] This is a quote from Leonhard Euler and is a perfect explanation of the thesis behind gyroscopes. A gyroscope is a device used to measure the orientation of an object in real time. There are several types of gyroscopes on the market, ranging from different build materials to different functions and features, but every gyroscope is built on the same principles. The principles of every gyroscope is called the principle of “Rigidity in Space” and explains the rotating mass inside the gyroscope. A gyroscope while spinning tends to remain at its original position and resist being moved, the greater the spinning mass and the closer this mass is located on the spinning axis the better a gyroscope operates. The reason for having a gyroscope for this project is for more accuracy in regards to the angular rate of motion, but there is a common problem using gyroscopes in a project like this because of the vibration the output can drift over time. To reduce the errors coming from this drift you

combine both the accelerometer and the gyroscope together to make the readings more accurate and reliable, this will make the robot be better at balancing when implemented both signals with the PID controller. There have been several gyroscopes considered, but due to the lack of progress on other blocks of the project the gyroscope have been put on hold all together and can be considered as future work.

### Kalman & Complementary Filter

When building a self balancing robot with both accelerometer and gyroscope you are required a math filter in order to merge the signals coming from both sensors. As mentioned above the gyroscope after some time can have a drift in values, so using both sensors this is minimized.

There are two common math filters used for this type of project, which is the Kalman and the Complementary filters, where the biggest difference is the complexity in algorithms and the implementation of these filters into software code. The most efficient filter type is the Kalman filter, and will minimize the error values and make it as close to its original value as possible. However, even though this is the best math filter to apply for this project, it is also the most complicated in terms of mathematical knowledge as well as implementation to software code.

In figure 4.4 the algorithms to achieve the kalman filter is represented:

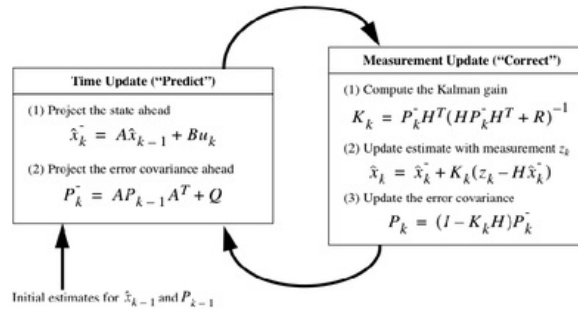


Figure 4.4 - Kalman Filter Illustration

In figure 4.5 you can see how the mathematical terms of the kalman filter is applied to reducing the errors of any measured stage from the sensors:

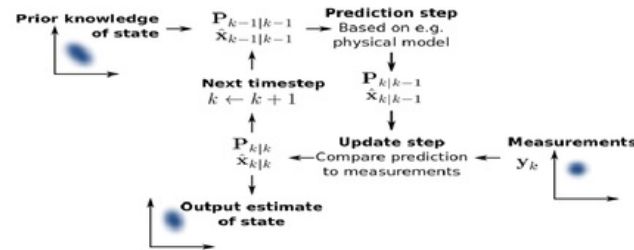


Figure 4.5 - Kalman Filter Terms

The implementation of the kalman filter have not been fully functional.

## PID Controller

When talking about balance there are several set of terms and variables that needs to be kept within a certain range in order to achieve this balance. There are many methods of keeping this range in your desired area, with a numerous set of systems and controllers. And by mentioning a few of the most common linear systems, the LQR, lead-lag, PID, state-feedback and pole placement the decision for this project was to try and design a PID controller to achieve the balancing robot. The PID controller is a common feedback loop system generally used in control systems and applications and fits in this sort of project. I brief terms the PID controller finds the difference between a user specific set point or in this case a predetermined position of 90 degrees, and the results that comes from the sensors, in this case the accelerometer. As mentioned in the literature review you have the option of up to three different systems ranging from P, PI, I, ID, PD and PID.

---

## Chapter IV.IV - Control System

### PIC Microcontroller

With all the information gathered from the accelerometer, translated through the PID controller you need “central hub” for this information to be controlled, systematically used and sent further down the block diagram. You are required to process the information gathered from the sensors efficiently which require you to have a fast “central hub”. This can be achieved by using a PIC microcontroller, and during the course of this degree a heavy focus around PIC microcontrollers made this choice rather easy. There are several other methods of controlling the information such as Raspberry PI, Arduino, or ARM chips to mention a few.

There are certain steps you as a designer have to follow in order to choose the correct PIC microcontroller as there are an enormous amount currently on the market. After deciding on several other steps in your design you acquire knowledge and requirements the microcontroller needs to fill, such as number of input- and outputs required, RAM/ROM, clock speed etc. as well as choosing the correct pic family. The major families being 8,16 and 32 bit microcontrollers, depending on the length of bits or number you are operating within corresponds to which family that would be able run them. With the complex algorithms needed for this project to achieve a self balancing robot you are required to maybe pick a microcontroller in one of the highest bit families in order for it to be able to run the calculations quick enough.

When you decide on a family there are hundred, if not thousands of microcontrollers within that family, so you will have to specify other terms in order to limit the range down to a manageable number. First up would be to decide on the memory size needed, with one goal in mind, the most ram from the cheapest price. In a microcontroller there are three basic memories within one chip, you have the program memory where the normal range goes from 1 to 32 K Bytes. This is where you are storing your program code, the bigger the program code, the bigger the program memory needs to be.

The second type of memory is the RAM, this have an enormous range going from as low as 1K byte to upward of numbers in the MB. Here is where you store your variables and parameter values, this type of memory is reset each time the pic turns off.

---

The third and last type of memory is the EEPROM memory and is the “Hard Disk Storage” of your PIC. In this type of memory there is a fixed written program with its variables stored and kept even if the PIC is turned off, the normal size of this type of memory is up to 4K is the most expensive part of the microcontroller in terms of memory.

Another aspect that needs to be decided is the number of pins required, as both input and outputs counts towards the total pin count. The most common pin counts is the 16 I/O with 18 pins and the 38 I/O with 40 pins.

If needed you can also get pic microcontrollers with different communications, with both analog and digital peripherals. Here is a list of some of the most typical:

#### **Analog Peripherals**

- Analog comparator
- Analog to digital converter
- Digital to analog convert

#### **Digital Communication**

- UART
- USART
- AUSART
- SPI
- I2C

#### **Connectivity**

- CAN BUS
- USB

---

### Other Peripherals

- Capture/Compare PWM features
- LCD Driver
- Quadrate Encoder Interface

For this project there is a requirement of three input pins coming from the accelerometer and at least two output pins going to the motors. Since the decision on going for an analog accelerometer the pic needs to have at least 3 analog to digital pins to gather the data from the accelerometer and translate the data into digital so that the pic can read and deal with the data appropriately. There are not many other features needed other than a fast clock speed so that the calculations can be done to minimize the error in the stability feedback loops. The decision was first to go for the PIC24HJ256GP210A[7] for its superior capability of these tasks. This microcontroller benefits from using 16-bit addressing which has a range of 0 to 65535 as opposed to the 0 to 256 range on the 8-bit family. Features this microcontroller have include the following:

- Low power
- Reprogrammable
- Onboard ADC
- PWM input/outputs
- Required I/O pins
- Timers
- Voltage comparator

This microcontroller was needed in order to achieve the aims and objectives of this project, but due to limitation in both time and resources another route had to be taken. When first choosing the PIC microcontroller it was mostly because of its speed and efficiency, but when the resources to handle this type of chip was not available it could no longer be used. There was no compilers available for the programming of this PIC so continuing forward would be difficult and time consuming. With the disregard to this PIC microcontroller another one had to be picked, but now with a different aim to be achieved.

Proof of concept is now the new aim for this project, as the self balancing part would be of too large

---

of a complexity to achieve. With the new aim in mind an analog accelerometer is now being connected to a microcontroller with its outputs going to an LCD display instead of motors. This is to prove that we could print out values coming from the accelerometer and implement those values with the PID controller to achieve the primary goal of building a self balancing robot.

For the new task it's required to have three analog to digital pins on the microcontroller, as well as at least 7 output pins for the LCD display. Due to its availability, cost and efficiency the choice of PIC microcontroller landed in the PIC16F887, the reason behind this was that this type of circuit is relative simple, so a standard microcontroller that could perform basic to moderate programs are more than capable. The PIC16F887 is one of the 8-bit family microcontrollers with 40 pins in total, with 35 being I/O pins there are more than enough to cover what this project aims to do, as well as having available pins for future improvements, features if needed. Some of the PIC16F887's main characteristics are stated in the table below, taken from the datasheet of this PIC:

#### **PIC16F887**

- Precision internal oscillator
- Wide Operating Voltage Range (2.0V - 5.5V)
- A/D Converter with 10-bit resolution
- Easy to learn with a 35 instructions
- Programmable Prescaler
- PWM output steering control
- Software Tuneable

With this microcontroller you have the option to either use the internal oscillator clock or have an external clock circuit connected. For this project there is an external clock of 8MHz connected to the PIC Microcontroller. Consulting the datasheet for this microcontroller it's stated the different types of configurations you could use for your external crystal.

Here is the design that was used in the circuit:

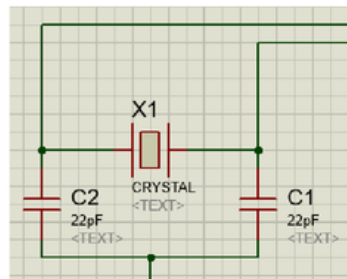


*Figure 4.6 - External Crystal Circuit*

It is said in the datasheet that you are required to connect a combination of two capacitors on both left and right of the crystal.

Here are some other PIC microcontrollers that was considered for this project:

#### **PIC16F877**

- Low instruction set to learn
- 8K x 14 words of FLASH Program Memory
- Wide operating voltage range 2.0V to 5V
- Timer0 with 8 bit timer/counter, 8 bit prescaler
- 10 bit multichannel Analog to Digital Converter
- Synchronous Serial Port, with SPI and I2C

---

## Chapter IV.V - System Response

### Motors

Motors is an essential component when you want to create mobility of a static object. The motors are the force that drives this static object in the direction the designer chooses. There are several types of motors that could be used for this type of project, ranging from Alternating Current (AC) or Direct Current (DC) types. Inside these two main types there are variances with special features and characteristics, which include brush, brushless, servo and permanent magnet type motors. One of the key features you want to look at when choosing motors is common between every type is Revolutions Per Minute (RPM).

The project requires two motors of the same type and size, as they are required to work together to provide the drive to counteract the force of gravity. Even though the decision in the beginning was to use DC motors, they have not been used in the end due to the fact that several key aspects of the final self balancing robot will not be present, e.g. the PID controller.

### Wheels

The wheels was required in order to provide traction for the base of the robot, as well as being utilised as a support structure for the chassis. How big the wheels needs to be is decided alongside the motors. It was stated in the original goal for this project that it was going to be a two-wheeled self balancing robot, so two wheels would have been used. This is to ensure that the aspect of balance needs to be achieved. The diameter of the wheels are chosen to best reflect the torque needed. Since we are using sensitive components in form of sensors etc. the wheels can impact the functionality of the sensors due to vibration which can be reduced by using wheels that would have a smoother traction to the surface e.g. rubber wheels.

### Motor Control

Motor controllers are systems that control the motors by varying the voltage with its output signals and is most commonly used to interface with DC motors. The most used control systems are direct, relay, transistor

and H-Bridge controllers and for this project an H-bridge motor controller will be used, mainly because it's been used before in previous projects(year 2, group project) so some basic knowledge is already gained from that. Another reason for using an H-Bridge is that it is an easy system for the implementation of a pulse width modulation (PWM) signal later on if needed. The PWM signal can be then used to varying the motor speed.

You can design your custom H-Bridge circuit by incorporating protection diodes and dual H-Bridges. Even though you could design your own circuit there is a more common choice available which is the L298 Motor Driver [8]. This driver is a more common choice mainly because of its efficiency, availability and cost compared to designed an h-bridge circuit from scratch. The L298 motor driver requires an input and enable voltage around 5-7V which can be provided by our VDD or a 9V battery. The motor bridge also have an onboard 5V regulator, so it is possible to outsource the voltage connected to this bridge to power the DC motors, or any other source that would require 5V input voltage. The motors is controlled by sending either HIGH or LOW signals to the drive for each motor.

In figure 4.7 you can see an example of how you would interface the motors with the L298 H-Bridge driver in proteus (LM = left motor)(RM = right motor):

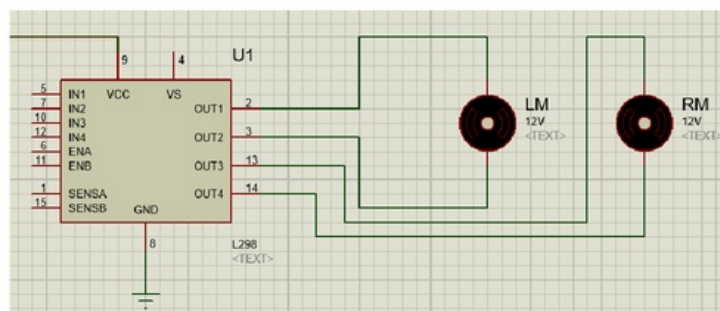


Figure 4.7 - H-Bridge Circuit

---

## Chapter IV.V - Voltage Regulator

When you design a circuit around the chosen components the datasheet is much help provide crucial information, especially regarding the voltage range of the components. Whether or not the design is focused on portability or not, there are always some advantages of using a voltage regulator to provide the circuit with a steady and stable supply of a voltage value. If you want to run your circuit from a 9V battery, you might have components that won't accept voltage that high, so this is where the voltage regulator comes in. After reading the datasheets of the components that is used in the circuit design you have a voltage range that the circuit needs to be operating in. Here is a table summarizing the different voltage ranges the components needs:

- PIC16F877 - 2.0V to 5V
- ADXL335 - 1.8V to 3.6V

A common voltage regulator is the LM117[9], and is a low dropout voltage regulator. This voltage regulator can provide a large fixed output voltage range from 1.8V, 2.5V, 2.85V, 3.3V and 5V, which is within the range of every component that is used in the circuit. This voltage regulator got 3 pins with each having a characteristic one input pin for input voltage, one output voltage adjustment pin, where you connect a resistor to adjust the desired output voltage and one output voltage pin, where your desired output voltage comes from.

Below is a generic voltage regulator circuit, where you have capacitors on each side to generate the stable voltage output:

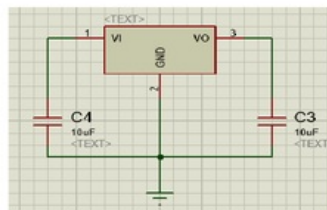


*Figure 4.8 - Voltage Regulator Circuit*

You have many choices for what values on your capacitors should be, and referring to the datasheet of your specific voltage regulator there are examples of what values would achieve what output. By using larger output capacitors you help smooth out the output, but too large and it will be unstable.

## Chapter IV.VI - LCD Display

When the decision on going a different path and trying to prove the concept of the self balancing aspect there was an easy choice of going with a LCD display to represent the accelerometer working, and how the numbers change in response to location in space. There are a number of different types of LCD displays where the most common being a 16x2 Liquid Crystal Display. When it comes to choosing the right display for the project you take into consideration these characteristics, price, number of columns, and number of rows. In this project a generic 16x2 LCD display is being used as the space on the display is more than enough to display the information required for the aim of this project. Another reason behind this type of display is that it was available through the university store and that it was non-complex, but not trivial interfacing it with the PIC microcontroller. The main reason behind choosing a LCD for this project was that it would make it easier to demonstrate the proof of concept with numbers changing on the screen, as well as demonstrating knowledge around interfacing an LCD with a PIC microcontroller.

Below is a block diagram showing how the interface of a generic LCD display works:

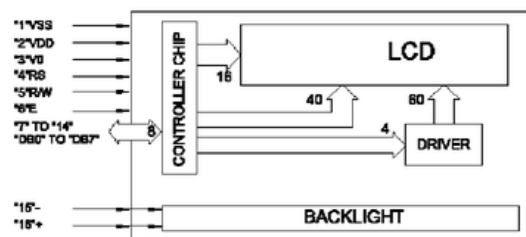


Figure 4.9 - LCD Block Diagram

---

Here is a diagram showing how the pin configuration works on a 16x2 LCD display:

| Pin No. | Symbol          | Level      | Description                                |
|---------|-----------------|------------|--------------------------------------------|
| 1       | V <sub>SS</sub> | 0V         | Ground                                     |
| 2       | V <sub>DD</sub> | 5.0V       | Supply Voltage for logic                   |
| 3       | V <sub>O</sub>  | (Variable) | Operating voltage for LCD                  |
| 4       | R/S             | H/L        | H: DATA, L: Instruction code               |
| 5       | R/W             | H/L        | H: Read(MPU →Module) L: Write(MPU →Module) |
| 6       | E               | H,H→L      | Chip enable signal                         |
| 7       | DB0             | H/L        | Data bus line                              |
| 8       | DB1             | H/L        | Data bus line                              |
| 9       | DB2             | H/L        | Data bus line                              |
| 10      | DB3             | H/L        | Data bus line                              |
| 11      | DB4             | H/L        | Data bus line                              |
| 12      | DB5             | H/L        | Data bus line                              |
| 13      | DB6             | H/L        | Data bus line                              |
| 14      | DB7             | H/L        | Data bus line                              |
| 15      | A               | —          | Supply power for LED +                     |
| 16      | R               | —          | Supply power for Red -                     |
| 17      | G               | —          | Supply power for Green -                   |
| 18      | B               | —          | Supply power for Blue -                    |

*Figure 4.10 - LCD Pin Configuration*

With every LCD display you have the option to have it in either 8 bit mode or 4 bit mode. The difference is that you are using 4 I/O pins in 4 bit and 8 I/O pins, where the advantage for 8 bit is faster data transfer with 8 bit transfer rate, while 4 bit mode can only handle 4 bits of data each transfer. The reason for using an LCD display in 4 bit mode is the required pins needed is reduced to 4 instead of 8, and if you do not necessarily need the faster 8 bit data transfer going for this option is very useful, as you have 4 spare pins for future features etc.

Here is how the PIC microcontroller is connected to the generic LCD display in proteus, this is to simulate the program code for future testing in the final circuit.

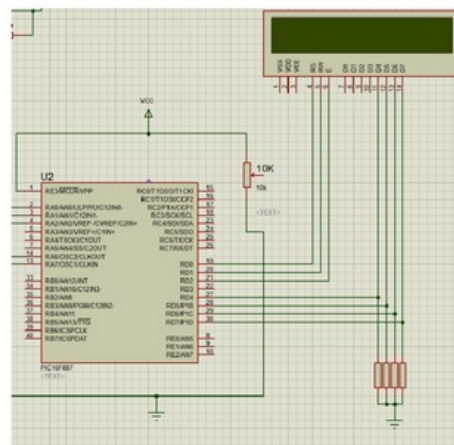


Figure 4.11 - LCD Circuit Diagram

It is not specifically required to have the four pull down resistors, but for this testing purpose it was put in to reduce the risk of simulation errors when trying to get the output pins to go high and send data to the LCD.

As mentioned before you had the choice of going either 8 bit or 4 bit mode on a LCD and by only connecting 4 pins to the microcontroller and the other 4 data pins to ground you have set the LCD into a 4 bit mode.

When interfacing a LCD display with a PIC microcontroller you have several paths to go in regards of software language. There are several software languages ranging from complex non trivial to more simplified and understandable languages. Which one that is more suitable for this project depends on previous relations and knowledge, as you have the option to do most of the common languages on the PIC16F887. For this project the CCS C programming language have been used, this is because it is similar to the objective-c language previously taught and might be easier to interface. Through the university there is access to the CCS for C compiler, so this was the language the software code interfacing the LCD to the PIC microcontroller was written in. A further explanation of the software code will be explained during the implementation chapter.

---

# Chapter V - Implementation

This chapter is focused on using the designs from the previous chapter and implement it to create the project.

It contains diagrams that further the knowledge gained through the decisions made in the design and will summarize the project. There will be figures showing the circuitry and how it was implemented in either proteus or on breadboards.

## Chapter V.I - Circuit Diagram

When the decisions on the design is final the next step is to implement this design into real time or simulated circuitry. You should first try and simulate using software to check for either errors or faults without the problem of either faulty parts or the possibility to damage your parts, as this costs money to replace in real time. The software used, as mentioned in the Literature Review is Proteus Simulation 8. Here is a circuit diagram created in proteus:

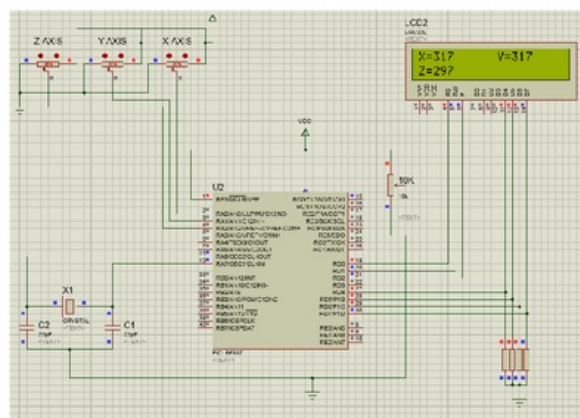


Figure 5.1 - Project Circuit Diagram

You can see the external crystal connected to a PIC microcontroller with three active potentiometers connected to three analog input pins. These three potentiometers are there to represent the three different axes gathered from the accelerometer as there are no other way to simulate the properties of a real accelerometer in proteus. They are used in order to construct the software code and to have values represented and displayed on the connected LCD display. There are no voltage regulator presented in this diagram as that will be addressed on another time. The main reason for the voltage regulator not being in this circuit design is that it was not required for the interfacing of the LCD display as they just run of a generic VDD supply line that can be set to any value. In figure 5.2, 5.3 and 5.4 you can see a screenshot of the circuit simulated with the values of from the active potentiometers being displayed on the LCD display, and is presented with best intention to demonstrate a representation of an accelerometer.

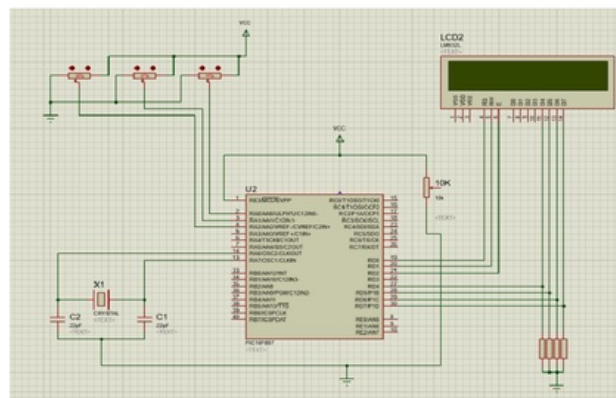


Figure 5.2 - Project Simulation (1)

On the next page in figure 5.3 the three axis represented with potentiometers is set with a different percentage and is clearly written on the LCD display.

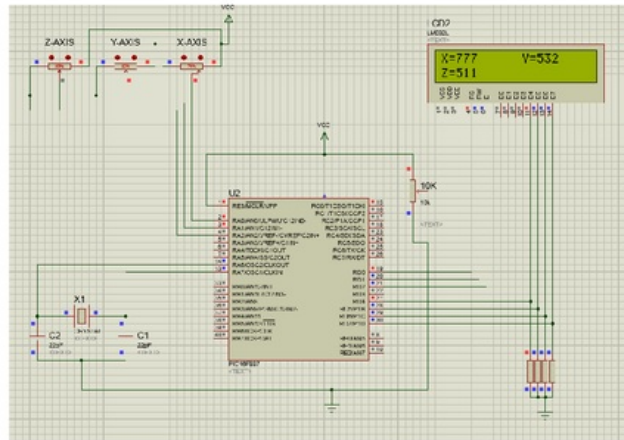


Figure 5.3 - Project Simulation (2)

Here you see the values of the three axis been changed as well as the figure below with them being changed again, this is just to demonstrate as to the best of my ability that the program code is in fact working as intended and that it can read the values from three of the analog input on the PIC microcontroller.

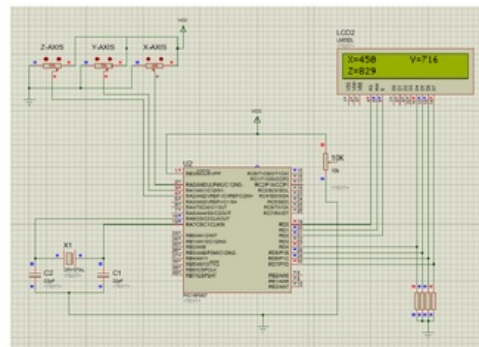


Figure 5.4 - Project Simulation (3)

---

Here is a picture of the circuit built on a breadboard:

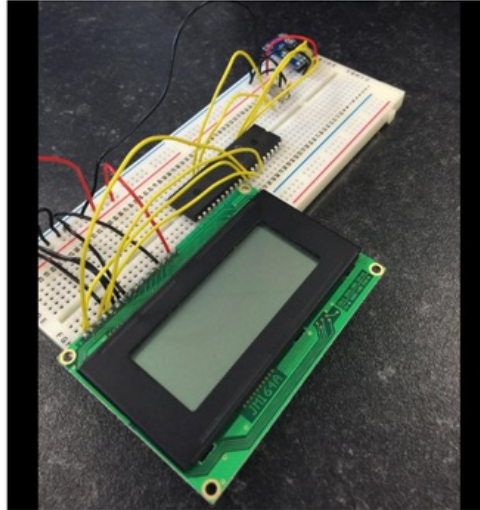


*Figure 5.5 - Circuit on Breadboard (1)*

Here is the LCD chosen for this project:

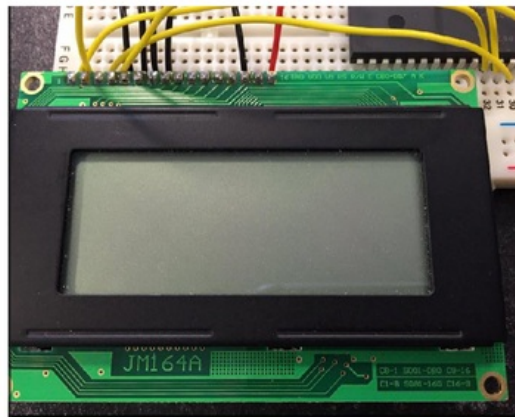


*Figure 5.6 - LCD display*

---

Here is the PIC fitted on the breadboard:

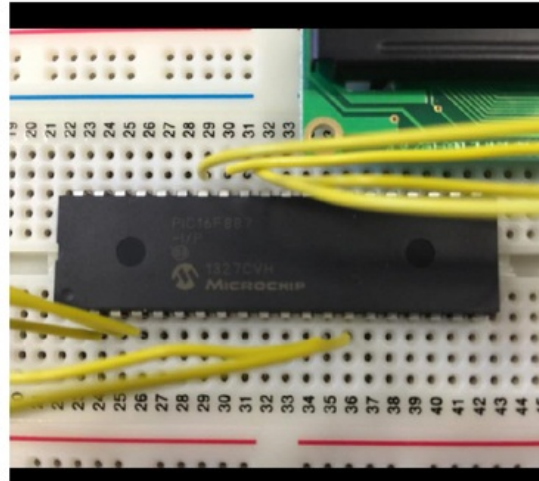


*Figure 5.7 - PIC16F887*

Here is the accelerometer connected on the breadboard:

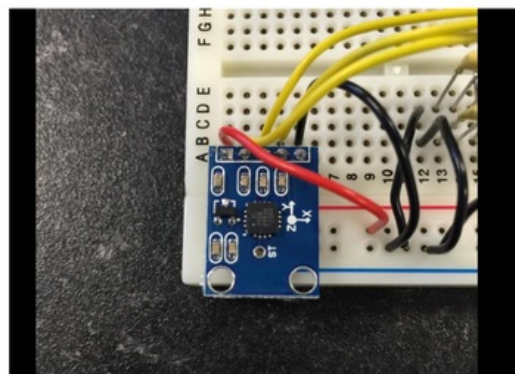


*Figure 5.8 - ADXL335*

---

Here is another overview of the circuit on the breadboard:

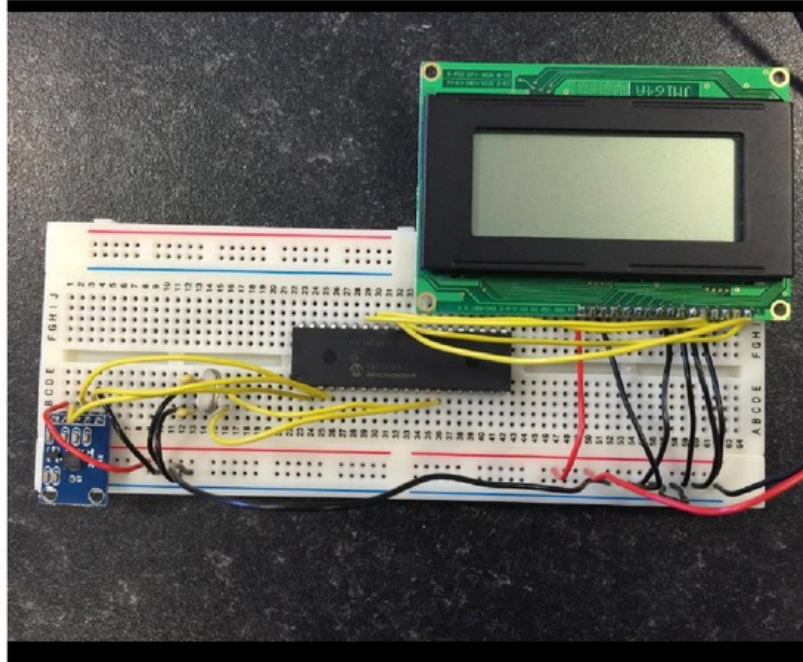


*Figure 5.9 - Circuit on Breadboard (2)*

---

## Chapter V.II - Software Implementation

The software implementation and code was done using CCS C program language, mainly because it is similar to objective-c and is fully functional with the microcontroller. Other languages that was considered was the assembly language, which is one of the most common languages used when programming a microcontroller. The software was first designed through a simple flowchart to get a brief concept of how the structure was going to be, there will be two different flowcharts present in this report, one being the original flowchart and one modified to represent the code that was actually implemented for this project.

The code was created using the project wizard from CCS as this is a major tool regarding programming PIC microcontrollers. It helps organize and structure your code so it is easier to read and build. In Figure 5.10 you can see the first few lines of the program, with the first two lines declaring which files that is included in the program.

```
1  #include <main.h>
2  #include <lcd.c>
3
4
5  void main()
6  {
7
8      unsigned long int value;
9
10
11      setup_adc_ports(sAN0|sAN1|sAN2);
12      setup_adc(ADC_CLOCK_INTERNAL);
13
14
15      lcd_init();
16  }
```

*Figure 5.10 - Project Code Sample (1)*

In Figure 5.10 above is also where you declare certain parameters in void main() that you would like to initiate during your program. In this case there is only one variable used which is value. On line 11 and 12 is where you declare which pins you would want to put as either analog or digital. Referring to the datasheet of

this microcontroller, the PIC16F887 there is a diagram showing which pins can be used as either analog or digital and which pins that have a A/D converter. For this pic the AN0-AN2 is used to gather the input signals from the accelerometer and convert them to a digital signal so that the microcontroller can use interpret and make use of them. The ADC\_CLOCK\_INTERNAL is used so that an internal clock from the microcontroller is being used to gather the data even when the PIC is sleeping. The last line of code on figure 5.10, lcd\_init() is just used to initialize the LCD display so that it can be used.

Them both being the main.h file, which have all the declarations that needs to be in a header file as seen in figure 5.11.

```
#include <16F887.h>
#device ADC=10

#FUSES PUT           //Power Up Timer
#FUSES NOMCLR        //Master Clear pin used for I/O
#FUSES NOPROTECT     //Code not protected from reading
#FUSES NOCPD         //No EE protection
#FUSES BROWNOUT      //Reset when brownout detected
#FUSES IESO          //Internal External Switch Over mode enabled
#FUSES FCEN          //Fail-safe clock monitor enabled
#FUSES NOLVP         //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES BORV40        //Brownout reset at 4.0V
#FUSES NOWRT         //Program memory not write protected

//LCD Module Connections
#define LCD_RS_PIN    PIN_D0
#define LCD_RW_PIN    PIN_D1
#define LCD_ENABLE_PIN PIN_D2
#define LCD_DATA4     PIN_D4
#define LCD_DATA5     PIN_D5
#define LCD_DATA6     PIN_D6
#define LCD_DATA7     PIN_D7
//End LCD Module Connections
#use delay(crystal=8000000)

#define LED_PIN_C0
#define DELAY 500
```

Figure 5.11 - Project Code Sample (2)

The header file is needed to declare certain characteristics on the PIC microcontroller and the LCD display. It is declared which pins on the LCD display is connected to which pin on the microcontroller. This is needed for the microcontroller to understand where to send the data and for the LCD to understand where the data is coming from, and how to display it.

```
17 while(true)
18 {
19 //Example blinking LED program
20     output_low(LED);
21     delay_ms(DELAY);
22     output_high(LED);
23     delay_ms(DELAY);
24
25
26     set_adc_channel(0);
27     delay_us(20);
28     value = read_adc(ADC_START_AND_READ);
29     lcd_gotoxy(1,1);
30     printf(lcd_putc, "X=%Lu", value);
31
32     set_adc_channel(1);
33     delay_us(20);
34     value = read_adc(ADC_START_AND_READ);
35     lcd_gotoxy(12,1);
36     printf(lcd_putc, "Y=%Lu", value);
37
38     set_adc_channel(2);
39     delay_us(20);
40     value = read_adc(ADC_START_AND_READ);
41     lcd_gotoxy(1,2);
42     printf(lcd_putc, "Z=%Lu", value);
43 }
```

Figure 5.12 - Project Code Sample (3)

Above in figure 5.12 there is one feature that was added, but is in no way related to the project itself, and that is the blinking LED program, this was added with the sole purpose of making sure that the microcontroller was configured correctly and functioning while doing the simulation. All the functions on figure 5.12 is put in a while(true) loop so that it can run all the time and not just once. The program will stay in this loop unless something happens, like the power is turned off or there is a glitch of any sort. There are three close to identical parts on figure 5.12 as the program's job is to read the analog output coming from the accelerometer(potentiometers) and display that value to the LCD Display.

On figure 5.13 you see the main part which is duplicated to replicate the three axis, X-Y, and Z-axis.

---

```
set_adc_channel(0);  
delay_us(20);  
value = read_adc(ADC_START_AND_READ);  
lcd_gotoxy(1,1);  
printf(lcd_putc, "X=%Lu", value);
```

*Figure 5.13 - Project Code Sample (4)*

On the first line in figure 5.13 you declare which pin on the microcontroller that will be set as a ADC channel. And for the first input the pin on the microcontroller is the AN0 pin, therefore you declare the input by calling the function `set_adc_channel(0)` and (0) is where you declare which pin you are initializing at any point. You call a delay function before you read the value gathered from that input pin with the function `read_adc(ADC_START_AND_READ)`. The next line just declares where on the LCD display the text is going to be printed by called the `lcd_gotoxy(1,1)`. The last line of this main portion is just the function of printing out the value taken from the input pin A0 with a simple `printf` statement, this is one of the main reasons that an analog accelerometer was chosen so that a simple `printf` statement for reading value could be used, making the code much simpler to handle.

As seen in figure 5.12 this main portion is duplicated with regards to all the three input channels AN0-AN2 by changing the channel name from (0) to either (1), or (2).

---

## Chapter V.III - Software Flowchart

Here is the original flowchart (figure 5.12) intended for this project, with a brief explanation on each step:

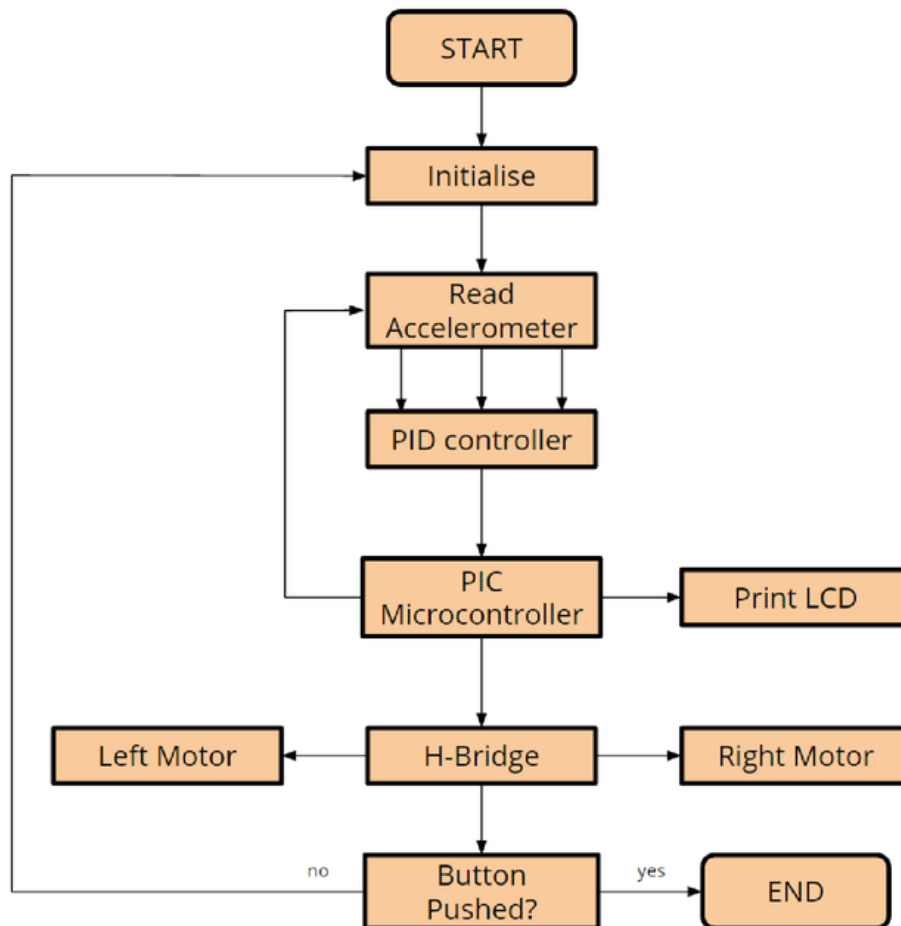


Figure 5.14 - System Flowchart

The first step is to initialise the program code and the other components that needs an initialization e.g. the LCD display. Then you jump down one step and read the values coming from the accelerometer. There are three arrows in the flowchart representing the three different values coming from x,y, and z-axis on the

---

accelerometer, this would have been sent through the PID controller as the controller feedback loop to minimize errors and achieve the balancing aspect. The value coming from the PID controller is then sent to the pic microcontroller so it could be used as a parameter for the motors. The code will continue to loop back to the accelerometer to gather the information and continue doing so until the program is shut down completely. This is needed to keep the information of where the robot is in space as updated as possible so that the balancing can be done.

The values will be printed to an external LCD display as seen on the flowchart before travelling further down the code to the motor driver (H-Bridge). The H-bridge motor driver is what controls the motors and would ideally be used for both backward/forward control as well as right/left turns if implemented in future work. The last step is to check if a button is pushed so that the user can have some sort of control of when the robot should be turned off or on. If the button is not pushed the code would then loop back to the accelerometer value to update all the steps in real time. The loop is needed to update the motors of what could have happened in terms of tilt errors and direction that the accelerometer would pick up.

---

# Chapter VI - Evaluation

This chapter will explain how the project turned out compared to the objectives and aims set out in the beginning. Are they met, did something change, and what could have been done differently.

## Chapter VI.I - Objectives & Aims Outcome

Here are some of the outcomes that can be drawn from the initial objectives and aims of this project.

- Knowledge from researching relevant topics and areas was gained which aided several of the design decisions taken on this project.
- An appropriate research methodology path was selected which aided in the completion of this report.
- Both a circuit design and software code was designed and built to complete certain aspects of this project within the time given, but some key aspects and features was not completed due to lack of progress in the time given and complexity of the project.
- A control filter was researched but not implemented

## Chapter VI.II - Research Review

Several key aspects and areas that was related to the topic of this project was researched during the literature review, and some have been implemented with success, where other areas was not completed, which would make the project not 100% completed. With some parts not being finished there have been lack of progress with the project all together, but in regards of the literature review the report have tried to best explain how the design process was meant to be implemented and how the research review helped.

## Chapter VI.III - Future Improvements

The project in itself have a lot of potential both in regards to the hardware but also the software. There are many features that could have been implemented in this type of design, such as audio and video streaming

---

capabilities, additional sensors to improve functionality, and even adding grippers to the body to make an arm capable of lifting objects etc. these type of features and improvement would greatly increase the complexity of this project and will therefore just be thought of additions and not tried in any shape or form.

A feature that would be beneficial and mentioned previously was the audio and video streaming, as this could expand into a territory of self navigation design and provide information to the robot to avoid obstacles etc. You see in the current market a huge interest in flying drones, with self navigation with the aid of multiple sensors to follow the user or even fly obstacle courses without any aid of external control systems. This would have been a feature for this project if time had allowed it, to design and implement a visual navigation system so that the robot not only be able to self balance itself, but also navigate around a set course without any external help.

Another great use of this project would have been a self made Segway of sort, that would benefit from the self balancing system already designed to allow users to drive the robot by using it as a balancing board with wheels, and implement control systems that would interact with how the driver tilted the board, to either driver it backward or forward.

A aspect that also would be of benefit for this project would have been a different approach to stability control. Maybe by using another form of math filters, sensors or motor driver you could improve the stability, reduce the tilt error, or even improve the energy consumption of the robot. Also improving the control system to give the robot the ability to rotate 180 degrees horizontal around its vertical axis could improve the features and abilities the robot could have.

---

# Chapter VII - Conclusion and Recommendations

## Chapter VII.I - Summary

After writing the report there are certain elements that after further reflection was the main aspect of not being able to finish the designs set out in the objective and aims, which is time management. Time management is a big key to reach goals that is set in the beginning of a project, and due to bad time management all goals set for this project have not been met. Even though the goals was not fully achieved, the research and literature review have been done to a certain extend so that the goals would be met if there had been more time.

Mentioned in the report there are aids and steps that have contributed to reach some goals during the project, a suitable research method along with relevant literature review have been key aspects to ensure that work progressed at a somewhat steady pace. The research method chosen in the beginning turned out to be the best suited for this type of project.

The initial aims and objectives for this project have not been fully achieved, but if time had been properly managed the goals set is not far off of being reached.

---

# Chapter VIII - Reflection

## Chapter VIII.I - Gained Knowledge & Experience

As a Computer Systems & Electronic student it was important to implement both hardware and software into this major project. And as a personal note starting a project that would on a subject both related to the course but also with a personal interest is just a bonus.

These are some of the key reflections gained by doing this project:

- A better understanding of theorems around self-balancing robots and how some of the industrial products are made was achieved during the research phase of this project.
- An increase in knowledge of electrical hardware, circuitry and basic to complex principles of electronics was achieved in order to solve problems and struggles that came up along the way.
- Both use of simulation and software implementation to understand and design a prototype is of great importance to an engineer, and by doing this during the report a more realistic view of how an engineer works in industry have been achieved.

---

## Chapter IX - References

[1] - Tom Harris. How Segway Works. Available from: <http://science.howstuffworks.com/transport/engines-equipment/ginger.htm>.

[Accessed: 20/4/2016]

[2] - Farlex, Free Dictionary. Coriolis Effect. Available from: <http://www.thefreedictionary.com/Coriolis+effect>

[Accessed: 20/4/2016]

[3] - Wikipedia. PID controller. Available from: [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller)

[Accessed: 20/4/2016]

[4] - Saunders, et al. (2009). Research Onion. Available from: <https://onion.derby.ac.uk/>

[Accessed: 20/4/2016]

[5] - Analog Devices. ADXL 335 Datasheet. Available from: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf>

[Accessed: 20/4/2016]

[6] - Leonhard Euler. Three Dimensional Rotation and Gyroscopes. Available from: <http://web.mit.edu/8.01t/www/materials/modules/chapter22.pdf>

[Accessed: 20/4/2016]

[7] - Microchip. PIC24HJ256GP210A. Available from: <http://ww1.microchip.com/downloads/en/DeviceDoc/70592d.pdf>

[Accessed: 20/4/2016]

[8] - Sparkfun. L298 Datasheet. Available from: [https://www.sparkfun.com/datasheets/Robotics/L298\\_H\\_Bridge.pdf](https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf)

[Accessed: 20/4/2016]

[9] - Texas Instruments. LM117 Datasheet. Available from: <http://www.ti.com/lit/ds/symlink/lm117.pdf>

---

[Accessed: 20/4/2016]

---

## Chapter X - Bibliography

[1] - Rideable Segway Clone. Available from: <http://www.instructables.com/id/Rideable-Segway-Clone-Low-Cost-and-Easy-Build/>

[Accessed: 20/4/2016]

[2] - Karl Johan Åström. (2002). PID Control. Available from: <http://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf>

[Accessed: 20/4/2016]

[3] - Building a Self balancing robot. Available from: <http://www.bajdi.com/building-a-self-balancing-bot/>

[Accessed: 20/4/2016]

[4] - Darci. PIC16F867A - Self Balancing Robot. Available from: <http://letsmakerobots.com/node/33243?page=1>

[Accessed: 20/4/2016]

[5] - Wikipedia. PID controller. Available from: [https://en.wikipedia.org/wiki/PID\\_controller](https://en.wikipedia.org/wiki/PID_controller)

[Accessed: 20/4/2016]

[6] - Dimension Engineering. A Beginner's Guide To Accelerometers. Available from: <http://www.dimensionengineering.com/info/accelerometers>

[Accessed: 20/4/2016]

[7] - Stephen Wong. Autonomous Stable Flight with PID Controller. Available from: [http://www.cs.cornell.edu/Courses/cs4758/2010sp/final%20project/stephen\\_wong.pdf](http://www.cs.cornell.edu/Courses/cs4758/2010sp/final%20project/stephen_wong.pdf)

[Accessed: 20/4/2016]

[8] - Robottini. Kalman filters vs Complementary Filter. Available from: <http://robottini.altervista.org/kalman-filter-vs-complementary-filter>

[Accessed: 20/4/2016]

---

[9] - Autonomous Dual Wheel Self Balancing Robot based on Microcontroller. Available from: [https://www.researchgate.net/profile/Khurram\\_Alimgeer/publication/236030928\\_Autonomous\\_Dual\\_Wheel\\_Self\\_Balancing\\_Robot\\_Based\\_on\\_Microcontroller/links/5402f92a0cf23d9765a56316.pdf](https://www.researchgate.net/profile/Khurram_Alimgeer/publication/236030928_Autonomous_Dual_Wheel_Self_Balancing_Robot_Based_on_Microcontroller/links/5402f92a0cf23d9765a56316.pdf)

[Accessed: 20/4/2016]

[10] - Miller. Building a two wheeled self balancing robot. Available from: [https://eprints.usq.edu.au/6168/1/Miller\\_2008.pdf](https://eprints.usq.edu.au/6168/1/Miller_2008.pdf)

[Accessed: 20/4/2016]

[11] - Analog Devices. ADXL335 Datasheet. Available from: <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf>

[Accessed: 20/4/2016]

[12] - Jaseem. LCD interface with Microcontroller PIC. Available from: <http://www.circuitgallery.com/2013/09/lcd-interface-with-microcontroller-pic.html>

[Accessed: 20/4/2016]

[13] - 3 Axis Accelerometer with Regulator. Available from: <https://www.rhydolabz.com/wiki/?p=1417>

[Accessed: 20/4/2016]

[14] - Sensing Orientation with ADXL335. Available from: <http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/>

[Accessed: 20/4/2016]

[15] - Voltage Dividers. Available from: <https://learn.sparkfun.com/tutorials/voltage-dividers/all?print=1>

[Accessed: 20/4/2016]

[16] - Microchip. PIC16F88X Family. Available from: <http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf>

[Accessed: 20/4/2016]

[17] - Ligo George. Interfacing LCD with PIC microcontroller. Available from: <https://electrosome.com/lcd-pic-microcontroller-ccs-c/>

[Accessed: 20/4/2016]

---

[18] - Marian Longa. Interfacing LCD with PIC. Available from: <http://marianlonga.com/interfacing-lcd-display-with-pic/>

[Accessed: 20/4/2016]

[19] - FHART. Operating Characteristics of the Segway. Available from: <https://www.fhwa.dot.gov/publications/research/safety/10025/01.cfm>

[Accessed: 20/4/2016]

[20] - Mark Hackman. The Technology behind the segway. Available from: <http://www.extremetech.com/extreme/72040-the-technology-behind-the-segway>

[Accessed: 20/4/2016]

[21] - Tom Harris. How Segway Works. Available from: <http://science.howstuffworks.com/transport/engines-equipment/ginger.htm>

[Accessed: 20/4/2016]

[22] - Coriolis Effect. Available from: <http://www.thefreedictionary.com/Coriolis+effect>

[Accessed: 20/4/2016]

[23] - David P. Anderson. nBot Balancing Robot. Available from: <http://www.geology.smu.edu/~dpa-www/robo/nbot/>

[Accessed: 20/4/2016]

[24] - Inverted Pendulum. Available from: <http://www.engr.usask.ca/classes/EE/480/Inverted%20Pendulum.pdf>

[Accessed: 20/4/2016]

[25] - Razvan v. Florian. Correct Equations for the dynamics of the cart-pole system. Available from: [http://www.coneural.org/florian/papers/05\\_cart\\_pole.pdf](http://www.coneural.org/florian/papers/05_cart_pole.pdf)

[Accessed: 20/4/2016]

[26] - PCB Heaven. Choosing the right PIC. Available from: [http://www.pcbheaven.com/picpages/Choosing\\_the\\_right\\_PIC/](http://www.pcbheaven.com/picpages/Choosing_the_right_PIC/)

[Accessed: 20/4/2016]

---

[27] - Huan Lin. Using Voltage Regulators. Available from: <http://41j.com/blog/wp-content/uploads/2014/08/HuanLin480ApplicationNote.pdf>

[Accessed: 20/4/2016]

---

# Chapter XI – Appendices

## Software Code:

```
//LCD Module Connections

#define LCD_RS_PIN    PIN_D1
#define LCD_RW_PIN    PIN_D2
#define LCD_ENABLE_PIN PIN_D3
#define LCD_DATA4     PIN_D4
#define LCD_DATA5     PIN_D5
#define LCD_DATA6     PIN_D6
#define LCD_DATA7     PIN_D7

//End LCD Module Connections


#include <main.h>
#include <lcd.c>


#use delay (clock=8000000)


void main()
{
    //setup_adc_ports(AN0);

    int left;

    lcd_init();
    Delay_ms(100);
    //left = read_adc();
```

---

```
while(TRUE)
{
    lcd_putc("\f");          //Clear Display
    lcd_putc("X-Axis: ");
    lcd_putc("Y-Axis: ");
    lcd_gotoxy(1,2);
    lcd_putc("Z-Axis: ");
    Delay_ms(2000);
    Delay_ms(2000);
}
```

```
/*while(TRUE)
{
    lcd_putc("\f");
    lcd_putc("X-Axis: ");
    Delay_ms(2000);
}*/
}
```