

**This document is downloaded from CityU Institutional Repository,
Run Run Shaw Library, City University of Hong Kong.**

Title	DIY Segway
Author(s)	Cheung, Ka Ho (張嘉豪)
Citation	Cheung, K. H. (2014). DIY Segway (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.
Issue Date	2014
URL	http://hdl.handle.net/2031/7413
Rights	This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.



香港城市大學
City University
of Hong Kong

Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

BENG3-CE-2013/14-WHL-13

Project Title
DIY Segway

Student Name: Cheung Ka Ho

Student ID:

Supervisor: Dr LAU, Ricky W H

Assessor: Dr SIU, Timothy Y M

Bachelor of Engineering (Honours) in
Computer Engineering

Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

Project Title : DIY Segway

Student Name : Cheung Ka Ho

Student ID:

Signature gary

Date : 21/4/2014

No part of this report may be reproduced, stored in a retrieval system, or transcribed in any form or by any means – electronic, mechanical, photocopying, recording or otherwise – without the prior written permission of City University of Hong Kong.

Acknowledgements

I would like to take this opportunity to express my deep regards and profound gratitude to my supervisor Dr. Ricky W.H. Lau for his guidance, encouragement and monitoring throughout this project. His help and support enable me to complete this Segway product.

I also take this opportunity to express my gratitude to technician Mr. Peter, Cheung who teach and share lots of experience and knowledge on the mechanical designs. He also give advice on the design of the Segway unselfishly.

I would also like to acknowledge with much appreciation to Mr. Yu Wing Lung for sharing his knowledge and experience on circuit design.

Lastly, I thank almighty, my classmates, friends and my parents for their consideration and encouragement without which this project would not be successful.

Abstract

Segway was once a mysterious invention created by Dean Kamen. The Segway Human Transporter is a personal transport device that uses a built-in computer to remain upright.

The aim of this project is to mimic the design of the Segway and build a low cost DIY Segway. The first stage of this project is to design the mechanical structure of the transport device. According to different power and functional requirements, different mechanical and electronic components have been chosen for the implementation.

The rider shifting weight and a manual turning mechanism on the handlebar are used to control the speed and direction of the Segway. Gyroscope and variable resistor are used to monitor user's physical motion.

The control board used in this project is an Arduino ATMEGA2560. A control software for monitoring the sensor readings, calculating PID and also outputting motor speed values has been developed. The two motors are controlled by a Maxon power board which receive signal from a purposely built DAC board. The control board varies the Segway speed by sending required data to the DAC board.

Table of Contents

Title	Page
Chapter 1 Introduction -----	1
1.1 What is a Segway -----	1
1.2 How to use a Segway -----	2
1.3 Some common use of Segway-----	4
1.4 Regulations about Segway -----	6
Chapter 2 Background study-----	8
2.1 Hardware-----	8
2.1.1 DC geared Motor-----	8
2.1.2 Battery-----	12
2.1.3 Maxon Motor Controller -----	14
2.1.4 Arduino Mega2560 -----	15
2.1.5 Accelerometer and Gyroscope -----	16
2.1.6 Variable resistor -----	17
2.1.7 DAC -----	18
2.1.8 I2C -----	18
2.1.9 SPI -----	19
2.2 Software Study -----	20
2.2.1 Arduino -----	20
2.2.2 Kalman filter -----	21
2.2.3 Proportional-Integral-Derivative Control -----	22
2.2.4 Eagle -----	23
Chapter 3 Project description -----	25
3.1 Motivation of the project-----	25
3.2 Project description -----	26
3.3 Project objectives -----	28
3.4 Project Block diagram -----	29

Chapter 4 Implementation -----	30
4.1 Chassis design -----	30
4.2 Hardware implementation -----	38
4.2.1 Power Board -----	38
4.2.2 DAC Board -----	44
4.3 Software implementation -----	47
4.3.1 Arduino and program overview -----	47
4.3.2 SPI and DAC initialization -----	49
4.3.3 I2C -----	52
4.3.4 Gyroscope and accelerometer -----	54
4.3.5 Kalman filter -----	57
4.3.6 PID control -----	65
4.3.7 Variable resistor -----	68
4.3.8 DAC board and motor control -----	69
Chapter 5 Result and testing -----	70
5.1 System testing -----	70
5.2 How to ride the DIY Segway -----	76
5.3 Segway self-balance -----	77
5.4 Segway Balancing with user -----	78
5.5 Top speed of the Segway -----	79
Chapter 6 Discussion -----	82
6.1 SPI signalling-----	82
6.2 Sampling time of PID -----	83
Chapter 7 Further development -----	84
Chapter 8 Budget -----	85
Chapter 9 Conclusion -----	86
References -----	87

List of Figures

Figure 1: Component of Segway PT	2
Figure 2: To step on the Segway	3
Figure 3: To control Segway going forward and backward	3
Figure 4: To turn the Segway left or right	4
Figure 5: Segway Polo	4
Figure 6: Segway guided tours	5
Figure 7: Police using Segway in the airport [5]	5
Figure 8: Segway X2 Golf model [6]	6
Figure 9: DC geared motor	11
Figure 10: A 30cm diameter wheel	12
Figure 11: Detail of a lead–acid battery	12
Figure 12: A Lithium Iron Phosphate battery	13
Figure 13: Connection part of the Lithium Iron Phosphate battery	14
Figure 14: Maxon motor controller board	14
Figure 15: Arduino Mega2560 controller board	15
Figure 16: MPU6050 front look	16
Figure 17: MPU6050 back look	17
Figure 18: logic on a variable resistor	17
Figure 19: A –8 bit DAC logic	18
Figure 20: I²C connection and logic	19
Figure 21: SPI connections	19
Figure 22: Arduino IDE logo	20
Figure 23: Arduino Programming environment	21
Figure 24: The Kalman model	22
Figure 25: The PID algorithm	23
Figure 26: Eagle start up page	24
Figure 27: Project block diagram	29
Figure 28: Base part design	30

<u>Figure 29: Wheel and axle design</u>	30
<u>Figure 30: Component parts after manufacture</u>	31
<u>Figure 31: Car chassis with motor and wheel installed</u>	31
<u>Figure 32: Motor and Wheel connected with a chain mechanism</u>	32
<u>Figure 33: Handlebar design</u>	32
<u>Figure 34: Handlebar case design</u>	33
<u>Figure 35: Connection part design and the variable resistor</u>	33
<u>Figure 36: Base design for variable resistor</u>	33
<u>Figure 37: Connection part, base and the variable resistor</u>	34
<u>Figure 38: Springs used in the handlebar case</u>	34
<u>Figure 39: Handlebar and the case</u>	35
<u>Figure 40: Shell design for variable resistor</u>	35
<u>Figure 41: Shell for variable resistor</u>	36
<u>Figure 42: Shell designed for motors and battery</u>	36
<u>Figure 43: Shells for the motors and battery</u>	36
<u>Figure 44: Safety wheel design and product</u>	37
<u>Figure 45: Logic of power board</u>	38
<u>Figure 46: Schematic of power board</u>	39
<u>Figure 47: Layout of power board</u>	39
<u>Figure 48: Soldered power board</u>	40
<u>Figure 49: CRO measure of battery output</u>	41
<u>Figure 50: CRO measure of 5V output</u>	41
<u>Figure 51: CRO measure of the another 5V output</u>	42
<u>Figure 52: CRO measure of the positive 15V</u>	42
<u>Figure 53: CRO measure of the negative 15V</u>	43
<u>Figure 54: Schematic of DAC board part1</u>	44
<u>Figure 55: Schematic of DAC board part2</u>	45
<u>Figure 56: Schematic of DAC board part3</u>	45
<u>Figure 57: Upper layer layout of DAC board</u>	46
<u>Figure 58: Bottom layer layout of DAC board</u>	46

Figure 59: Soldered DAC board	46
Figure 60: Flow chart for the initialization of Arduino	47
Figure 61: Console output of initialization	48
Figure 62: Program flow after initialization	48
Figure 63: Program for SPI transmit	49
Figure 64: Send data though SPI	49
Figure 65: Output range Select register of AD5734R	50
Figure 66: Power control register of AD5734R	50
Figure 67: I²C read program flow chart	52
Figure 68: I²C write program flow chart	53
Figure 69: I²C Pins in Arduino board	53
Figure 70: Setting for gyroscope and accelerometer	54
Figure 71: Reading data from sensors	54
Figure 72: Gyroscope sensitivity	55
Figure 73: Program for calculation of gyro rate	55
Figure 74: atan2 around a circle	55
Figure 75: Program for the calculations	56
Figure 76: Variable used in Kalman filter	57
Figure 77: Init value of variables	57
Figure 78: Program code for prediction step	59
Figure 79: Program code for update the state covariance matrix	60
Figure 80: Program code for comparing the values	60
Figure 81: Program code for updating covariance	61
Figure 82: Program code for calculate Kalman gain	62
Figure 83: Program code for prediction model update	63
Figure 84: Program code for covariance update	64
Figure 85: console output for gyroscope readings	64
Figure 86: Function of PID algorithm	66
Figure 87: Flow chart for PID	66
Figure 88: The variable resistor	68

<u>Figure 89: Program for reading sensor and measuring the speed value</u>	68
<u>Figure 90: Program code for speed outputting</u>	69
<u>Figure 91: Result of speed control</u>	69
<u>Figure 92: Hardware setup of the balancing system</u>	70
<u>Figure 93: Output of the above system</u>	70
<u>Figure 94: Increase the resistance value</u>	71
<u>Figure 95: Output result of increased resistance</u>	71
<u>Figure 96: Decrease the resistance value</u>	72
<u>Figure 97: Output result of decreased resistance</u>	72
<u>Figure 98: Simulation for going forward</u>	73
<u>Figure 99: Output result of going forward</u>	73
<u>Figure 100: Simulation for going backward</u>	74
<u>Figure 101: Output result of going backward</u>	74
<u>Figure 102: Output result of going right when moving forward</u>	74
<u>Figure 103: Output result of going left when moving forward</u>	75
<u>Figure 104: Output result of going left when moving back</u>	75
<u>Figure 105: Output result of going right when moving back</u>	75
<u>Figure 106: DIY Segway</u>	76
<u>Figure 107: Self balanced Segway</u>	77
<u>Figure 108: Segway balancing in user mode</u>	78
<u>Figure 109: Console output of balance speed</u>	79
<u>Figure 110: Console output of forward speed</u>	80
<u>Figure 111: Console output of backward speed</u>	80
<u>Figure 112: Calculation for backward maximum speed</u>	81
<u>Figure 113: SPI program</u>	82
<u>Figure 114: Consecutive output during operation</u>	83

Chapter 1 Introduction

1.1 What is a Segway [1]

Dean Kamen first introduced the idea of personal transportation during the mid-1990s. Nowadays, this invention – Segway Human Transporter, is a common sight around the world.

The Segway PT is a two-wheeled, battery-powered, self-balancing electric vehicle. The two coaxial wheels are driven independently by a controller that balances the vehicle both without and with a rider. Sensors and gyroscopes in the car base give the feedback for regulation of balancing. The system is responsive to provide adequate balancing for different riders and riding styles and it is robust enough to accept riders of different weights.

A user can command the Segway to go forward or backward by shifting their weight on the platform. The Segway will detect the tilting angle and the change in its center of mass. In order to maintain balance, it will first establish and then maintain a corresponding speed to go forward or backward. A handle bar is used for the user to command the car to go left or right. The Segway can reach a maximum speed of 12.5 miles per hour.

1.2 How to use a Segway [2]

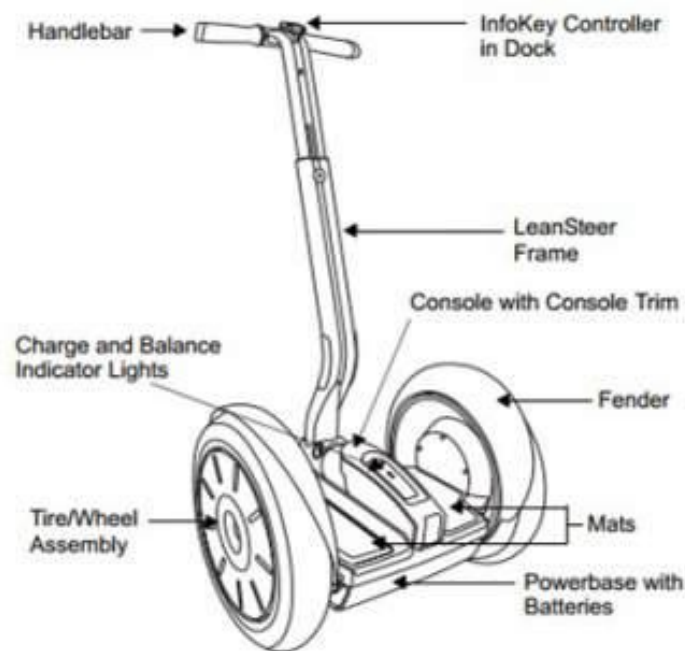


Figure 1: Component of Segway PT

The Segway PT that sold on website consist of the above components and subsystems. The follow shows some steps in using the Segway PT. This project is going to mimic some of the steps, functions and components to design the DIT Segway.

1. Choose a flat and smooth area to start up your Segway. User should put on their helmet when riding this car.
2. The handlebars on the Segway should be adjust for a comfortable and safe ride.
3. User should find the mode button, key port, display and steering grip on the handlebars. The display will provide information for a safe ride.
4. Segway need to be turn on by putting the key into the key port. User should stand on the side of the Segway and tap and hold the mode button. A green smiling face on the display represent the Segway is ready to ride.

5. The balance mode should be tested by moving the handlebar back and forth to see if the wheels respond.
6. When a rider is ready to step on the Segway, another person should stand in front of the Segway and hold the handlebar securely. After the rider step onto the Segway, the rider should stand straight and feel the Segway balance

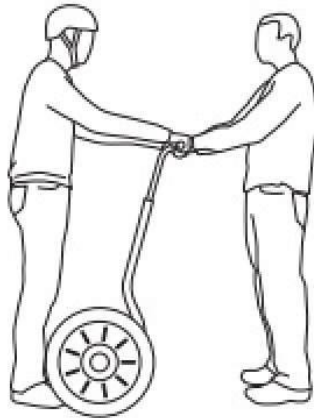


Figure 2: To step on the Segway

7. Rider should keep their legs and arms loose with knees slightly bent. Rider should grip the handlebar lightly
8. Rider could lean forward to move the Segway forward and lean backward to travel backwards on the Segway. To turn the car, rider can shift the handlebar.

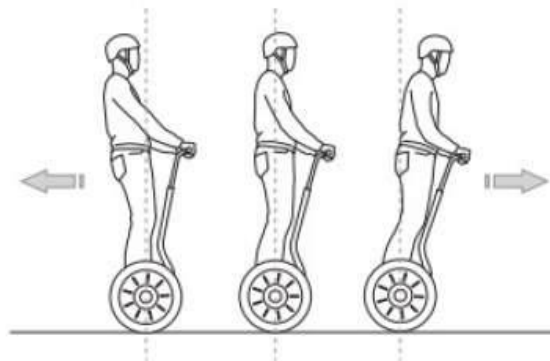


Figure 3: To control Segway going forward and backward

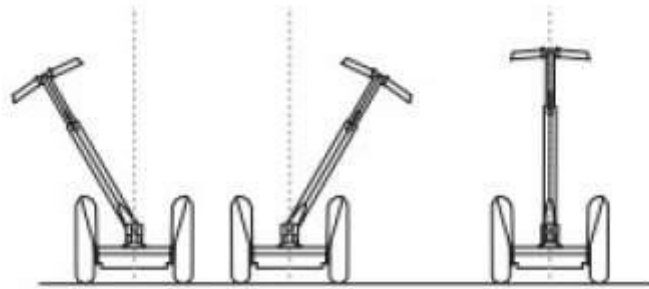


Figure 4: To turn the Segway left or right

9. When rider want to stop he should first shift his weight to the centre. Rider should step off the Segway one foot at a time. After stepping off the Segway, the power should be turned off.

1.3 Some common use of Segway

When Segway was first invented, it develop a reputation for being nerdy and useless, but in the last few years people found plenty of things that it can be used for. This personal transporter become more popular as more uses are found and the gas prices rise.



Figure 5: Segway Polo

Segway Polo [3] is like regular polo, players ride on a Segway instead of horses. Segway PT is used on the field. Two teams of five players each hit a ball with their

mallets and try to get the ball into other team's goal. The first organized match was in 2004 and now the game is played worldwide.



Figure 6: Segway guided tours

Many historical areas and cities offer Segway guided tours [4]. Sightseeing can be hard and tiring on the feet. This easy mode of transportation allows visitors to quickly glide around town from point to point.



Figure 7: Police using Segway in the airport [5]

The maximum speed of the police use Segway can reach to 12mph. Although it could not keep up with a car, police can kept the road in sight. It was easy to catch up with the perpetrators when they were on foot. The Segway widens their field of vision which help them arrive an emergency scene more safely and quickly.



Figure 8: Segway X2 Golf model [6]

Segway have different models that suit for different usage. There is off-road models for some off road sports. There are also “Segway X2 Golf” which is a special model that equipped with special tires and golf club bag for golf players to use.

1.4 Regulations about Segway [7]

Below shows some regulations in some areas about Segway. It shows that many countries and company accept these kind of transporters

- Asia
 - China: Police have begun using Segway to patrol certain areas, such as Tiananmen Square.
 - Japan: Segway was classified as a motorcycle. There is no report of registration. Manufacturers commonly sell Segway to corporations to use in facilities
 - Singapore: Segway were used in Airport, it can be by customer service and security employees.
- Europe
 - Austria: Segway can be rented in parks.

- Italy: Segway is allowed to use within city limits wherever pedestrians or bicycles are allowed. For example some sidewalks, bicycle paths and parks.
- Germany: Segway is allowed on bicycle paths and public roads within city limited. Segway used on public roads must be equipped with front and rear lighting, reflectors, a bell, and an insurance plate. The driver must have procured a vehicle insurance and license.
- Switzerland: Segway is classified as a light motorcycle. Segway PT i2 model has been approved for use. It can be used on roads provided that it is equipped with a license plate and Swiss road kit.

In Hong Kong, according to Road Traffic Ordinance chapter 374 [8]. Any battery powered and three wheeled car should have license in order to use on public roads. In addition the car should be examined by Transport Department. The car should equipped with front and rear lighting, license plates and mechanical brakes. The design of the Segway in this project is not allowed to use in public areas in Hong Kong.

Chapter 2 Background study

Literature review was undertaken, focussing on the hardware and software part of the project. Session 2.1 discuss about the hardware informations. They included the hardware components, main boards, external sensor device and communication protocol. In Session 2.2 software that used in this project is discuss. They included the programing software and mathematical algorithms.

2.1 Hardware

Section 2.1.1, 2.2.2 and 2.2.3 discuss about the motors, battery and the motor controller. They are the most important components in this project. Calculation need to be carried out in order to avoid compatibility problem. Main board and the external sensors is discussed in section 2.1.4, 2.1.5 and 2.1.6. In Section 2.1.7, 2.1.8 and 2.1.9, DAC and communication protocol like I²C and SPI is discussed.

2.1.1 DC geared Motor [9]

Gear motor is a design that allows low horsepower motor to drive a large force on an object but with a low speed. The increase in torque is inversely proportional to the decrease in speed. With the gear motor, even a small motor can support as large torque as human weight.

To choose a motor, the following calculations need to be carried out:

- 1) Convert mph to km/h
- 2) Convert km/h to m/s
- 3) Calculate the force
- 4) Calculate the power of the motor

5) Calculate the rpm and the diameter of the wheels

Segways found in market have three different speed standard to suit for different level of users. For the beginner standard, the maximum speed is 6mph. For intermediate standard, the maximum speed is 8mph. For the advance standard, the maximum speed is 12mph. In addition, the standard Segway weighted 60kg.

1) Convert mph to km/h and km/h to m/s

a)beginner	b)intermediate	c)advance
6mph = 10km/h 10km/h = 2.7m/s	8mph = 13km/h 13km/h = 3.61m/s	12mph = 20km/h 20km/h = 5.5m/s

Table 1: Calculation of mph to m/s

2) Calculate the force (F=ma i.e. Assume 5s is used to go to top speed)

a)beginner	b)intermediate	c)advance
$F = ma$ $F = (120kg) \frac{2.7777m/s}{5s}$ $F = 66.6N$	$F = ma$ $F = (120kg) \frac{3.6111m/s}{5s}$ $F = 86.6N$	$F = ma$ $F = (120kg) \frac{5.5555m/s}{5s}$ $F = 133.3N$

Table 2: Calculation of Force

3) Calculate the power of each motor (power = FV)

a)beginner	b)intermediate	c)advance
$P = (66.66N) * (2.77m/s)$ $P = 179.9999W$	$P = (86.66N) * (3.61m/s)$ $P = 312.8666W$	$P = (133.3N) * (5.55m/s)$ $P = 733.333W$

Table 3: Calculation of power

As there is two motor for driving the car, the power of each motor is total power divided by two.

a)beginner	b)intermediate	c)advance
$P = \frac{179.9999W}{2}$ $P = 89.9999W$	$P = \frac{312.8666W}{2}$ $P = 156.433W$	$P = \frac{733.333W}{2}$ $P = 366.666W$
Power of the motor available = 250W	Power of the motor available = 250W	Power of the motor available = 350W

Table 4: Calculation of power for each motor

For battery voltage up to 24V, beginner and intermediate standard require 10A and advance standard require 16.666A.

4) Calculate the rpm and the diameter of the wheels

$$Speed = \frac{(rpm) * (2\pi) * (\frac{diameter}{2})}{60s}$$

Equation 1: Speed equation

The above equation is used to relate speed and the diameter of the wheel. In choosing the motor of 350W standard, it can have a maximum of 400rpm per second with no load. When there is load, the maximum rpm is 320.

a)beginner	b)intermediate	c)advance
$2.777 = \frac{(400) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.1326m$	$3.611 = \frac{(400) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.1724m$	$5.555 = \frac{(400) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.2652m$

$2.777 = \frac{(320) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.1658m$	$3.611 = \frac{(320) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.2155m$	$5.555 = \frac{(320) * (2\pi) * (\frac{d}{2})}{60s}$ $d = 0.3316m$
--	--	--

The wheels' diameter should be chosen with a value between the loaded rpm and without load rpm.

By the above calculations, a motor of 24V with 350W should be chosen and a wheel of diameter about 30cm should be used. In addition, a motor controller that can support 20A should be chosen.



Figure 9: DC geared motor



Figure 10: A 30cm diameter wheel

2.1.2 Battery

Form the calculation in section 2.1.1, the battery chosen in this project should have at least 24V and 17A to support the motors and the system to operate.

The lead–acid battery [10] is the oldest type of rechargeable battery. It is able to supply high surge currents and the cells have a relatively large power-to-weight ratio. The price of these kind of battery is relative low and it is popular among motor vehicles for the high current requirement.



Figure 11: Detail of a lead–acid battery

Each battery can have a voltage level of 12V, by connecting two battery in serious can achieve the target of 24V. This set of batteries was used in the early stage of the project. Considering the weight of them, another replacement was researched.



Figure 12: A Lithium Iron Phosphate battery

Lithium Iron Phosphate battery [11] is the battery that used in the finalized Segway. The voltage and current level of it is 24V and 20A. It is enough for supporting the system to run.

Lithium Iron Phosphate battery can provide stable voltage output upon fully charged. The chemical is non-toxic, green and stable. It can be worked under high temperature and the lifetime is long. Weight of this battery is about 4.5kg which is suitable for small vehicles.

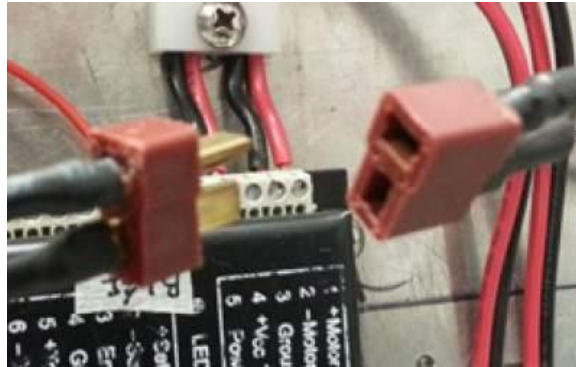


Figure 13: Connection part of the Lithium Iron Phosphate battery

The connection part of this battery is a “T-head”, which can prevent the swapping between positive and negative terminal.

2.1.3 Maxon Motor Controller



Figure 14: Maxon motor controller board

Maxon Motor controller is a powerful servo amplifier, it can be used to drive motors from 80Watts up to 500Watts. It support different mode depends on user requirement. User can choose to use a tacho signals or an encoder signals to control the speed. There is also torque or current control and IxR compensated speed control.

Maxon Motor controller is protected against excess temperature, short circuit and excess current on the motor winding. The efficiency of the motor control can reach to 95% with the FET power transistors incorporated in the servo amplifier. It support a nominal supply voltage range of 12-50VDC. The maximum output current can reach 20A. It is suitable for using in an electric vehicle with require high power.

2.1.4 Arduino Mega2560 [12]

Atmega2560 is the controller that use on Arduino Mega2560 microcontroller board. It has 15 PWM output, 16 analog inputs and 54 input and output pins. The board provide a USB connection, a power jack, a reset button, and ICSP header and a 16MHz crystal oscillator. To program the device, user can connect the board to the computer using a USB cable. Accept from USB port to supply power, the board can be also power up by connecting the power jack with a 5V supply.



Figure 15: Arduino Mega2560 controller board

Many computer have an internal protection on the USB ports. This Arduino board provide an extra protection of a resettable polyfuse that protect from overcurrent or short circuit. If there is more than 500 mA applied to port, connection will be automatically break until overload or short circuit is removed. It is useful during debugging stage as the Segway system under test is running at a high power level which may damage the computer if accidents occur.

2.1.5 Accelerometer and Gyroscope [13]

MPU-6050 combines an on board Digital Motion Processor with a 3-axis accelerometer and a 3-axis gyroscope on the same silicon die. It can be used to process complex 9-axis Motion Fusion algorithms. Communication with MPU-6050 is achieved over two pins SCL and SDA. He AD0 pin can be connected to ground of VCC, it determine the I²C address of the chip. This AD0 feature allows you to have multiple gyroscopes without mixing the I2C address together.



Figure 16: MPU6050 front look

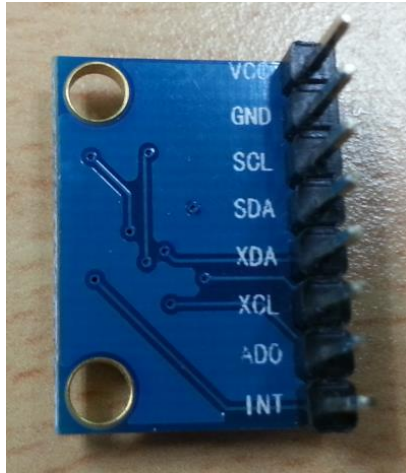


Figure 17: MPU6050 back look

2.1.6 Variable resistor [14]

A variable resistor is a three-terminal resistor with a continuously adjustable knob in which use can rotate it to adjust the resistance value. These resistors are known as the potentiometers, they act as a continuously adjustable voltage divider. One of the example usage is the volume adjust in a radio.

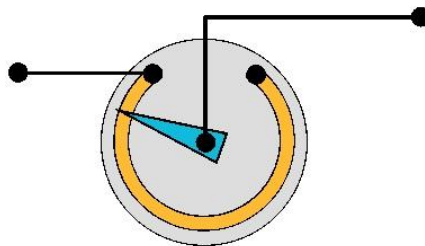


Figure 18: logic on a variable resistor

$$V = I \cdot R.$$

Equation 2: Ohm's law

When a user turn the knob the pointer will rotate to a corresponding location. The resistance value is depends on the length of the yellow circle it touch. With the above equation, the analog output of the resistor will be vary according to the resistance value.

2.1.7 DAC [15]

Digital to analog converter is the circuit that performs the digital to analog conversion. This process is to convert signals having a few defined levels or states into signals that have a theoretically infinite number of states.

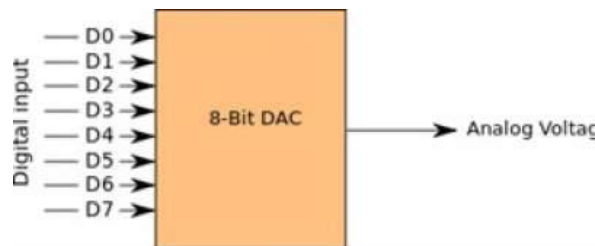


Figure 19: A –8 bit DAC logic

2.1.8 I²C [16]

I²C is a communication protocol uses 2 signal lines. They are called SDA (serial data) and SCL (serial clock). It is a multi-master protocol in which the number of slaves and masters are not limited.

Each slave will have their own unique address. When a master want to address the slave, the address is used in waking up the slave device. First, master will issue a start signal. All the device on the bus will listen to the incoming data. Then master will sends the address of the device it is going to access and also whether it is going to read or write. After that all the device will compare the address, device that are not match will wait until the bus is released by stop signal, the device that match will produce a acknowledge signal. After the

acknowledgment, master can start to transmit or receive data and slave will start sending the requested data byte by byte.

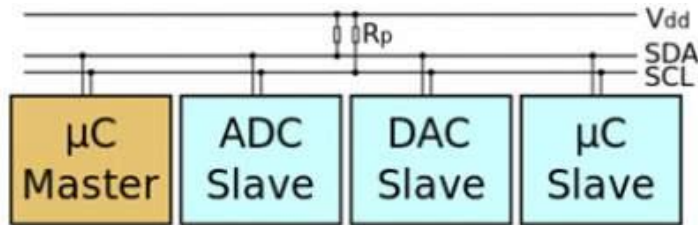


Figure 20: I²C connection and logic

2.1.9 SPI [17]

SPI is a single-master communication protocol. A central device is named master and it is responsible to initiate all the communications with the slaves. When a master want to send data to a slave or request data from slave, it need to select the slave by pulling the SS line low and activates the clock signal with a clock frequency that is usable by the master and the slave. The master will then generates information onto the MOSI line while it will samples the MISO line to receive data.

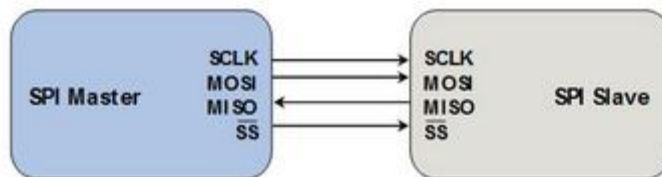


Figure 21: SPI connections

2.2 Software Study

This session provide the software and the mathematic algorithms that had been used in this project. They include Arduino as the programing software, PID algorithm, Kalman filter and eagle software for drawing board layout.

2.2.1 Arduino [18]

Arduino is an open source computing platform for Arduino microcontroller board. It provide a development environment for user to write software program and upload to the controller. The extensible software and the open source resources provide user a clear and simple programing environment. Arduino board is cheaper comparing with other microcontroller which is suitable for project development.

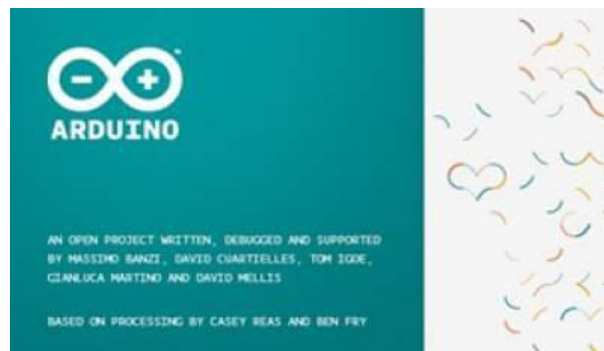


Figure 22: Arduino IDE logo

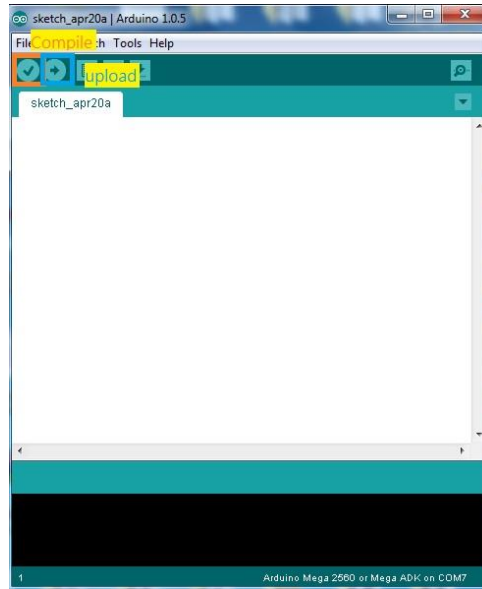


Figure 23: Arduino Programming environment

2.2.2 Kalman filter [19]

Kalman filter is known as the linear quadratic estimation. It is an algorithm that use a series of measurements observed over time to produces estimations. These data contains random variations like inaccuracies and noise. Based on the data, Kalman filter can produce the value of an unknown variable that tend to be more precise than measurement done alone.

Numerous applications in technology can apply Kalman filter. A common application is for navigation, control and guidance of aircraft, spacecraft and vehicles. It can also be applied in time series analysis used in fields such as econometrics and signal processing.

The Kalman filter algorithm works in a recursive manner. In the prediction step, Kalman filter produces estimation of the current state variables along with the uncertainties. When the next measurement is read, together with the random noise and error, the estimation model is

undated using a weighted average. After a few run, with more weight being given to estimates with higher certainty.

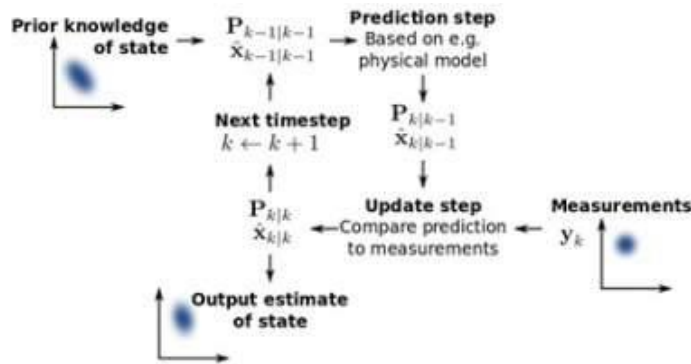


Figure 24: The Kalman model

2.2.3 Proportional-Integral-Derivative Control [20]

P, I, D stands for proportional, integral and derivative. A PID controller is a control loop feedback mechanism used in some industrial control system for example temperature control. PID control calculates and error value as the difference between the desired set point and a measured process variable. The controller attempts to minimize the error by adjusting the process control inputs.

The three value P, I, D is constant parameters that used in the control algorithm. Proportional depends on the present error, integral is the accumulation of the past errors and derivative is a prediction to the future error based on the current rate of change.

P, I and D can do different combination depends on the system needs. It can be changed into a PD, PI, I or P controller.

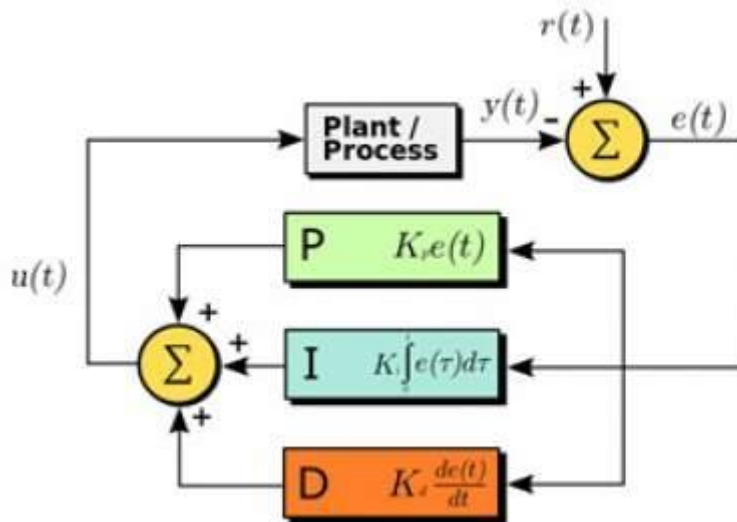


Figure 25: The PID algorithm

2.2.4 Eagle [21]

Eagle is a PCB design software, it stands for Easily Applicable Graphical Layout Editor. Eagle is a powerful, user friendly and affordable for users to implement their PCB design. It included variety of functions for example library drawing for electronic components, schematic drawing and board layout design.

Eagle is a popular PCB design software that consist of lots of online or build in libraries to support user's design. The build in standard libraries consist of some normal use components. If user cannot find the required components in the build in library, a lot of libraries can be found on web or even users can design their own components.

After the design, user need to fabricate the PCB. Eagle software provide a Gerber file output for user's PCB layout. User can choose the required file or data from the setting and these files can be used in fabrication.



Figure 26: Eagle start up page

Chapter 3 Project description

3.1 Motivation of the project

The Segway is a product that can be used in many different kind of area. It can be used in some parks like Ocean Park or Disney land for visitors to navigate. It can also use to replace police car for policemen to make an arrest.

There is no lack of examples for the using of this product. Many of the tourist sight in Singapore have Segway Eco Adventures. User can ride on a Segway and pay visit to the beach sides and some off road areas. In China, Segway was used in some airports and Tiananmen Square for police to patrol. It help to reduce the time for patrols and complete more patrols of the premises in the expected time.

The problem of this product is the prize is quite high and also user concern about its' safety problems. The advertisement is also not enough, as a result it is not very popular among citizens.

In Hong Kong, it is not popular and even people have no idea what is a Segway. This project aims to promote the benefits and the use of Segway. As a result, a DIY Segway was build up with some necessary components and mechanisms, so as to demonstrate the ideas of this personal transporter.

3.2 Project description

Component list:

1. Arduino ATMEGA2560
2. 24V Lithium Iron Phosphate battery
3. DC geared motor x2
4. Maxon motor controller x2
5. DAC board
6. Power board
7. Accelerometer (analog signal)
8. Gyroscope (analog signal)
9. Variable resistor (analog signal)

Implementation:

Gyroscope is used to monitor the angle of the car chassis. The gyroscope output gives the angle but it drifts a lot. By combining the result with the accelerometer values and pass through the Kalman filter, the output angle will be more precise. If the car is leaning, micro-controller will notice the angle changed and output a speed signal to the dac board after calculation. The speed of the wheels depends on the leaning angle. Platform balancing is done by controlling the motor speed and direction, and PID is the algorithm in calculating the speed value. The speed value include the direction and magnitude.

Variable resistor is used for detecting the turning motion of the handle bar. The value of variable resistor will be originally set to middle. Micro-controller will keep monitoring the analog value of the resistor. If the resistance increased or decreased, which means the handlebar is turned. Micro-controller will give out a signal to speed up one wheel which gives a result of turning left or right.

Over speed problem will be solved depending on the actual model and user, as weight, centre of gravity will also need to be considered. One of the solution to this problem is before the speed of the wheel cannot balance the user as the rider lean too down, micro-controller will increase the wheel even more to push back the rider.

3.3 Project objectives

- 1) To design a Car Chassis which can/have
 - Support 90kg person
 - Fit through door frames
 - Non-slip surface on platform for feet
 - Ergonomic and aesthetically pleasing
 - Handle bar for turning
 - Safety Wheels
- 2) To choose components
 - Motors that are
 - ✧ Bidirectional
 - ✧ Enough speed
 - ✧ Small
 - Battery that
 - ✧ Is Rechargeable
 - ✧ Provide power to all components
 - ✧ Provide adequate current to motors
 - Motor Controller
 - Wheels
 - Controller
 - ✧ Enough processing speed
 - ✧ Enough I/Os
 - ✧ Low power consumption
- 3) To write a program that perform
 - Monitor balancing
 - Monitor turning
 - Speed calculation
 - Motor controlling

3.4 Project Block diagram

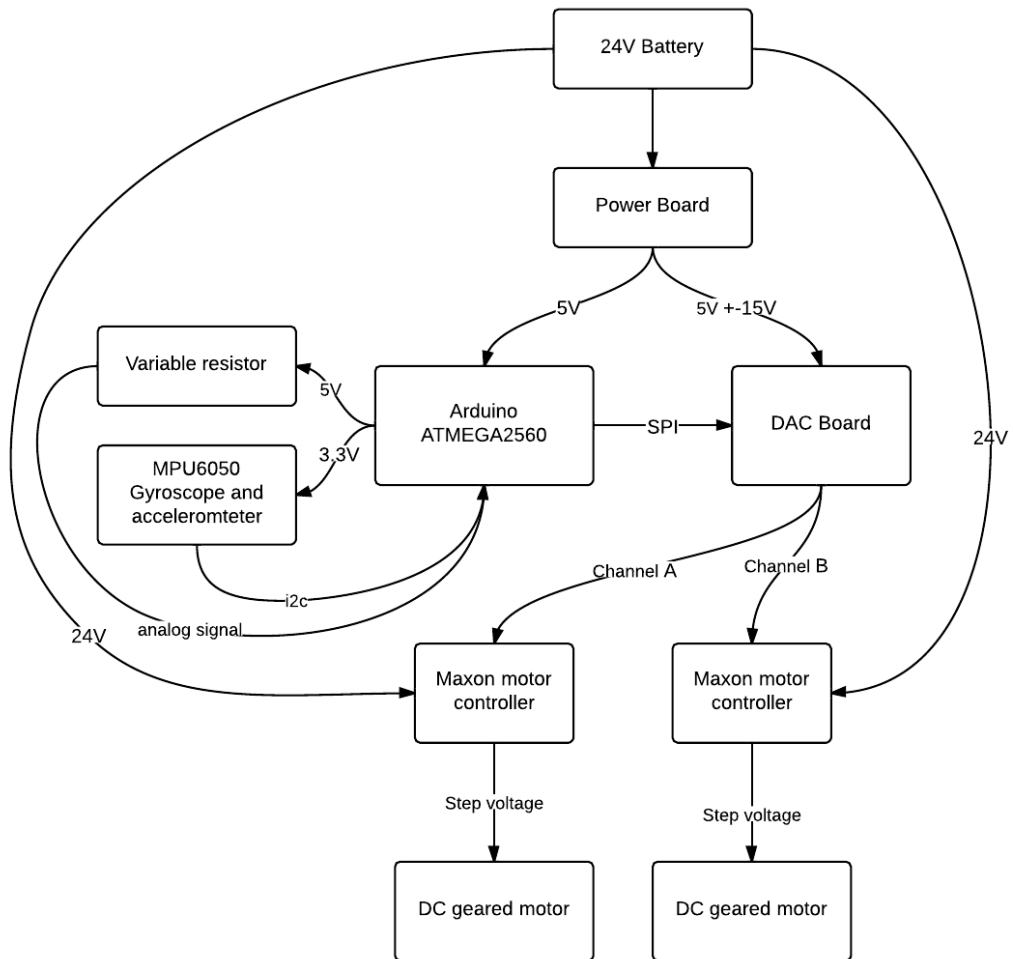


Figure 27: Project block diagram

Chapter 4 Implementation

4.1 Chassis design

Below shows the design of the chassis and their description.

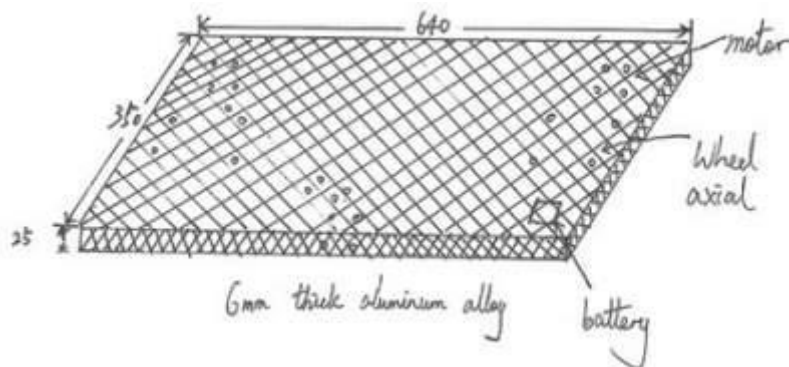


Figure 28: Base part design

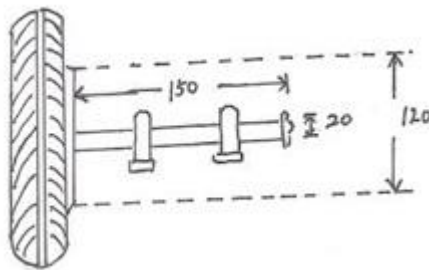


Figure 29: Wheel and axle design



Figure 30: Component parts after manufacture

The two sets of motors are installed in the opposite side to balance their weight

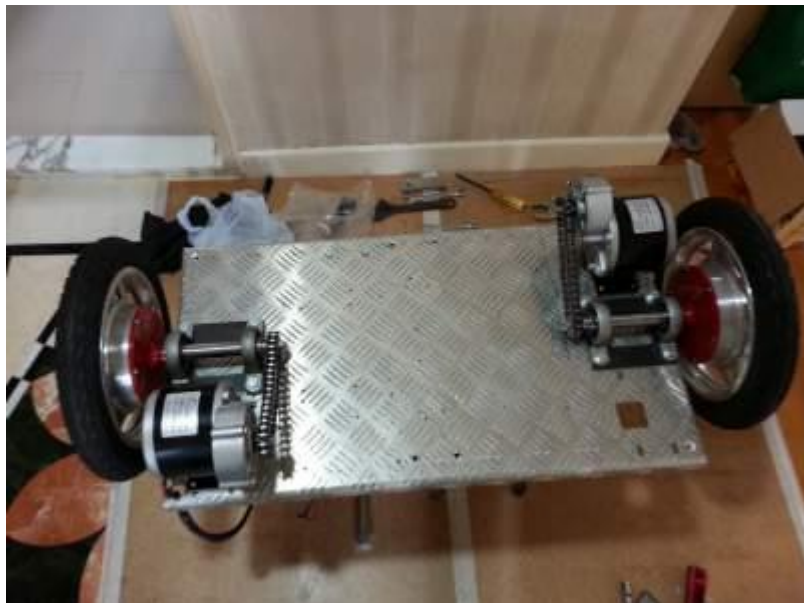


Figure 31: Car chassis with motor and wheel installed

The axle of the motor has a short diameter, it is not enough to support large torque as human weight. A thicker axle is designed and used to support human weight and it is connected to the motor through a chain mechanism.



Figure 32: Motor and Wheel connected with a chain mechanism

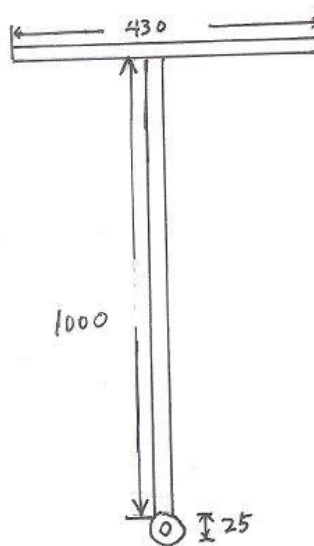


Figure 33: Handlebar design

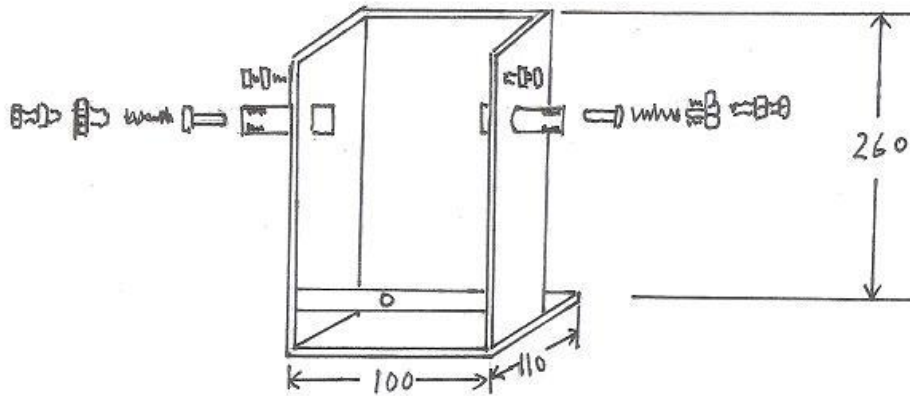


Figure 34: Handbar case design

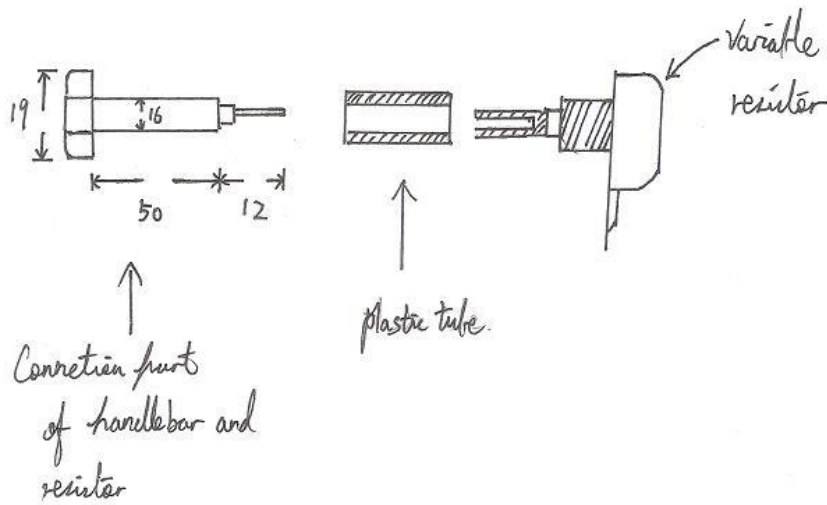


Figure 35: Connection part design and the variable resistor

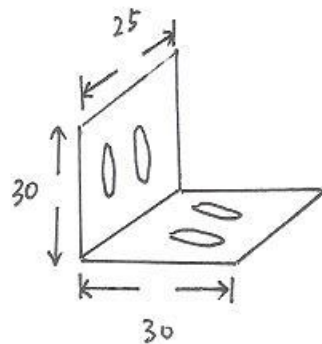


Figure 36: Base design for variable resistor



Figure 37: Connection part, base and the variable resistor



Figure 38: Springs used in the handlebar case



Figure 39: Handlebar and the case

The handlebar part is connected to the chassis through a case. The case uses a spring to maintain the handlebar in the center part of the Segway. When the handlebar is turned, the knob of the variable resistor will rotate.

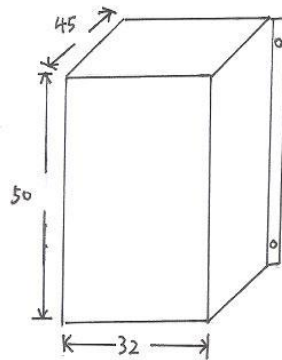


Figure 40: Shell design for variable resistor



Figure 41: Shell for variable resistor

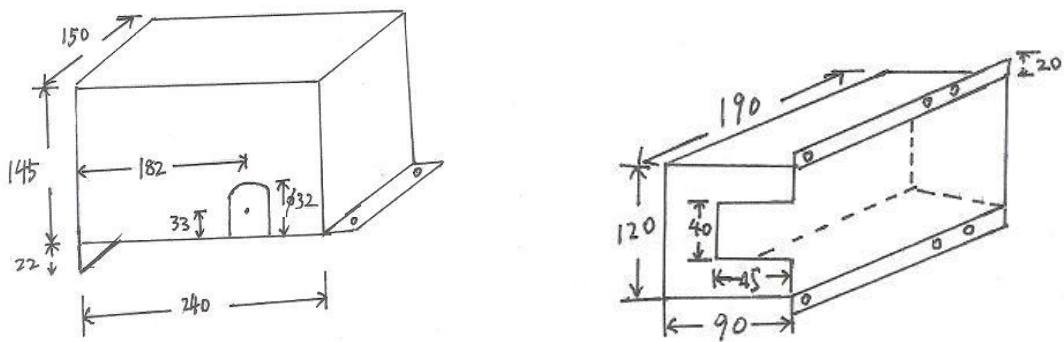


Figure 42: Shell designed for motors and battery

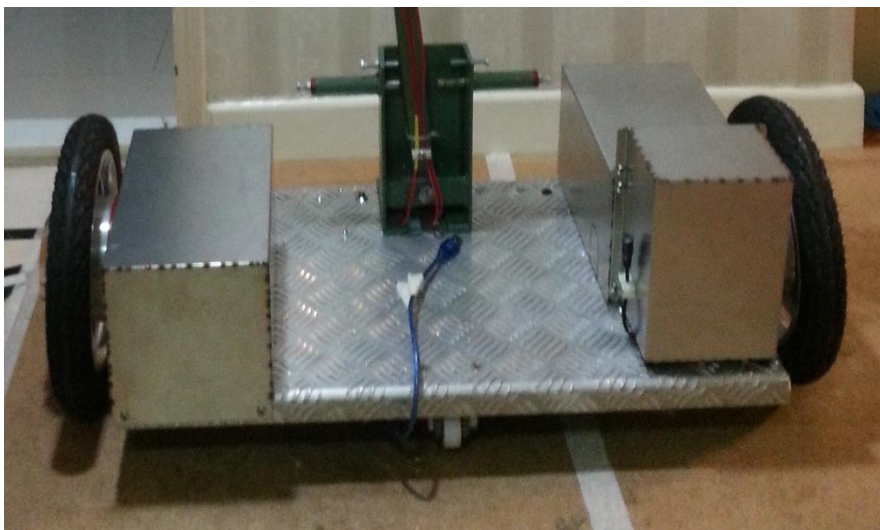


Figure 43: Shells for the motors and battery

Some shells are designed to protect the components and user from getting hurt.

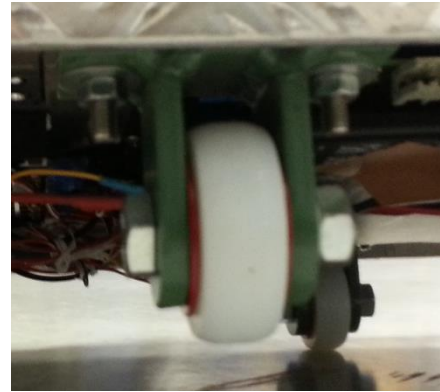
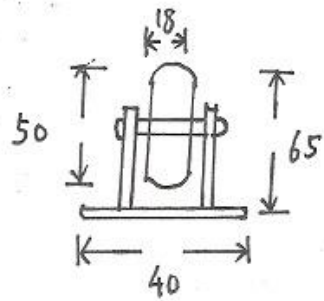


Figure 44: Safety wheel design and product

A safety wheel is designed to protect the Segway from turnover. The safety wheel limited the angle of the Segway can tilt, so it can also prevent over speed.

4.2 Hardware implementation

In this chapter the implementation of the hardware design is discussed. It include the design of semantic, board layout, finalized appearance and their output. The software eagle is used in the board designing.

4.2.1 Power Board [22]

Lithium Iron Phosphate battery provide a voltage level of 24V. Arduino board and the DAC board need a supply of 5V, a positive and negative 15V to operate. In order to make the system with a single power source, a power board is used to regulate the voltage into different level to suit for different components. The logic flow to the power board design is shown in the below figure.

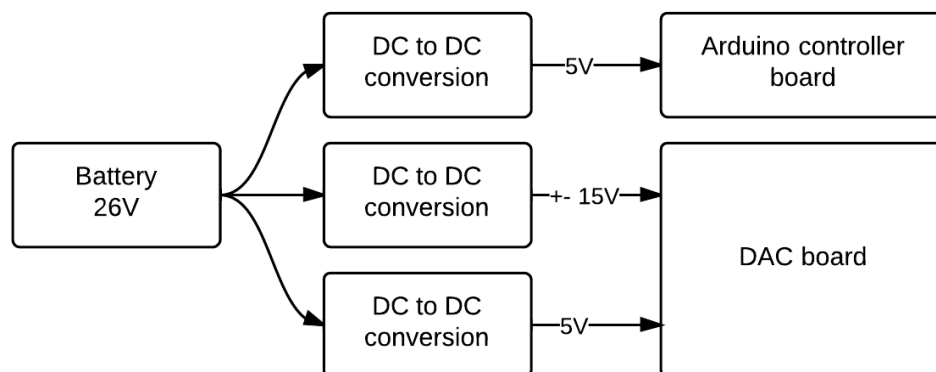


Figure 45: Logic of power board

The implementation of schematic and board layout are shown in the below figures.

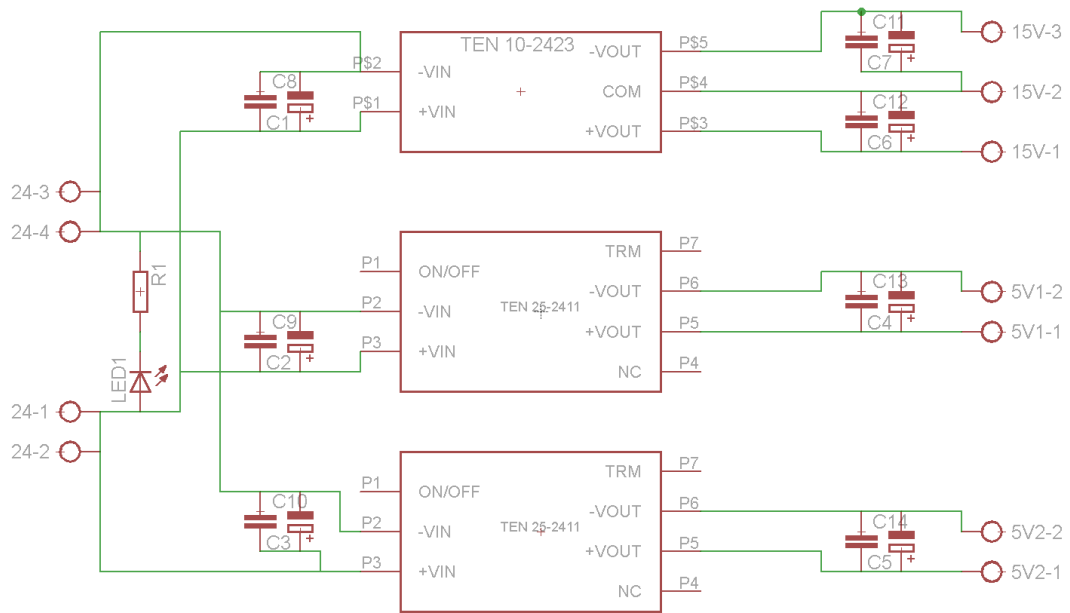


Figure 46: Schematic of power board

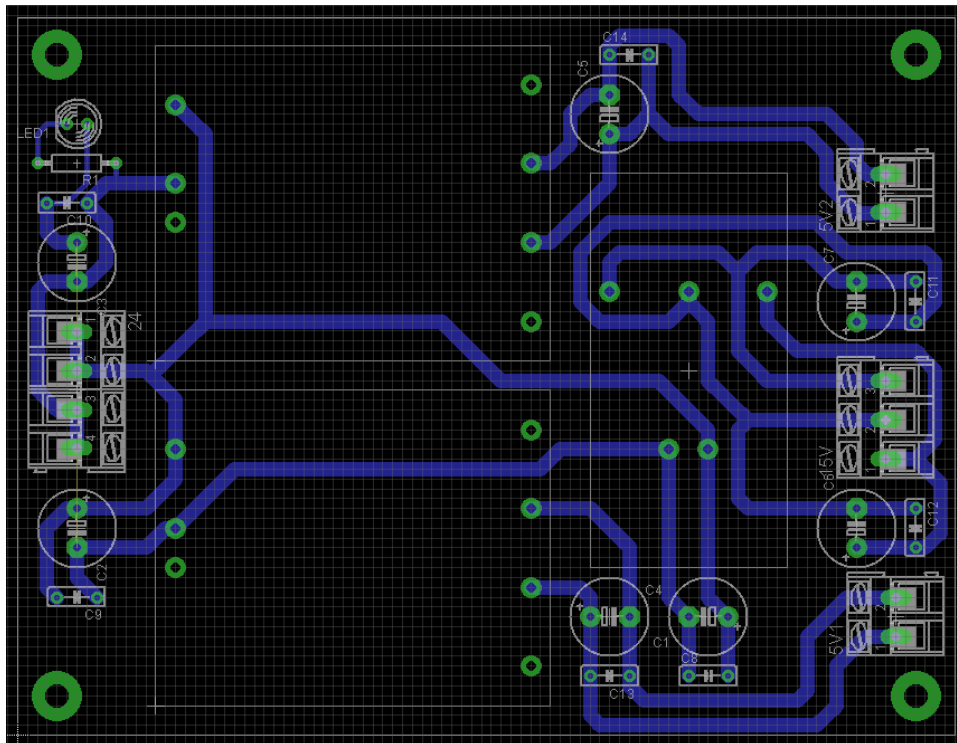


Figure 47: Layout of power board



Figure 48: Soldered power board

Two standard of “Traco block” is used in the power board. TEN25-2411 is used to convert 26V into 5VDC. TEN10-2423 is used to convert 26V into positive and negative 15V. The Power board is tested with CRO after soldering. The CRO result is shown below.

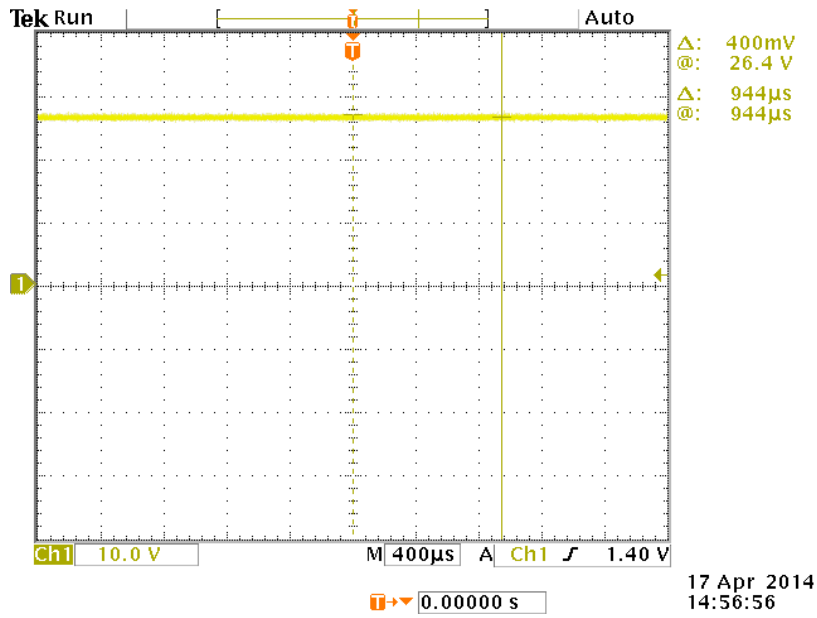


Figure 49: CRO measure of battery output

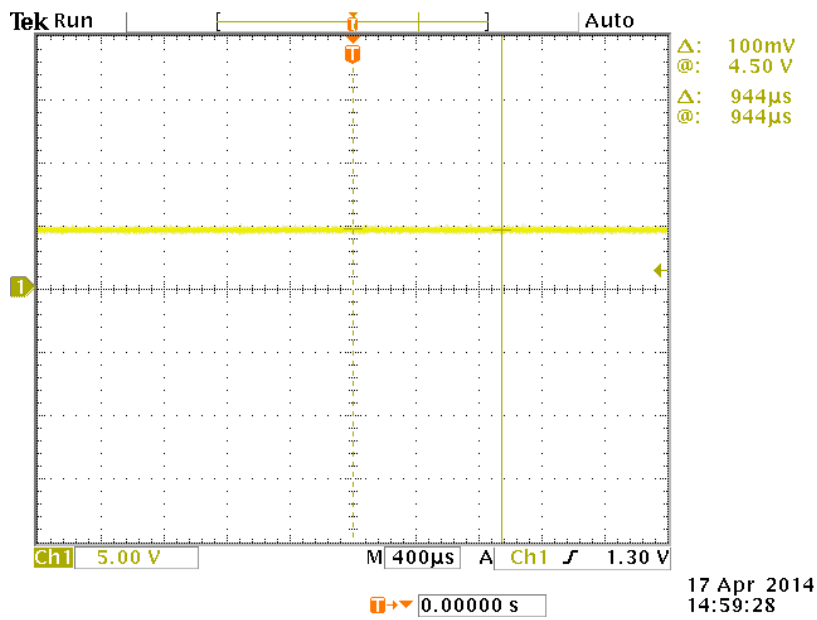


Figure 50: CRO measure of 5V output

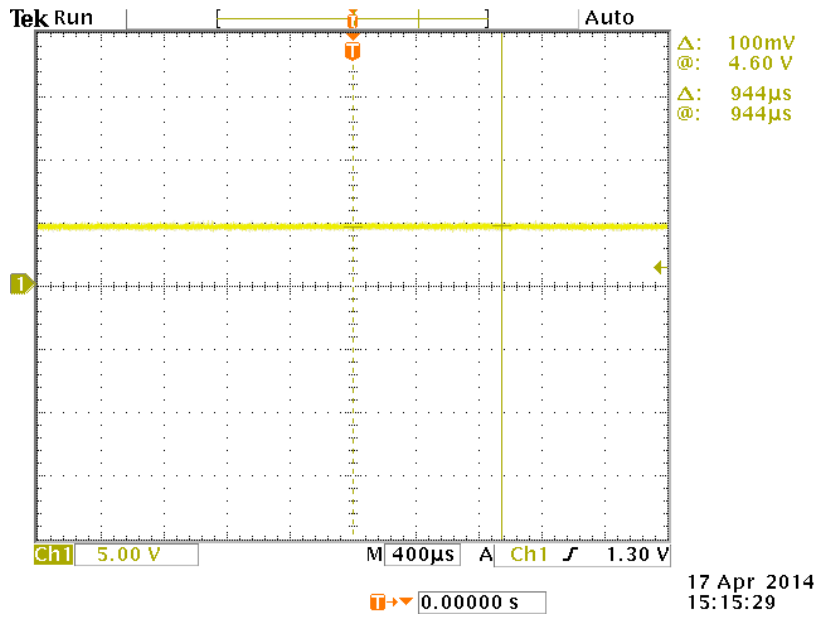


Figure 51: CRO measure of the another 5V output

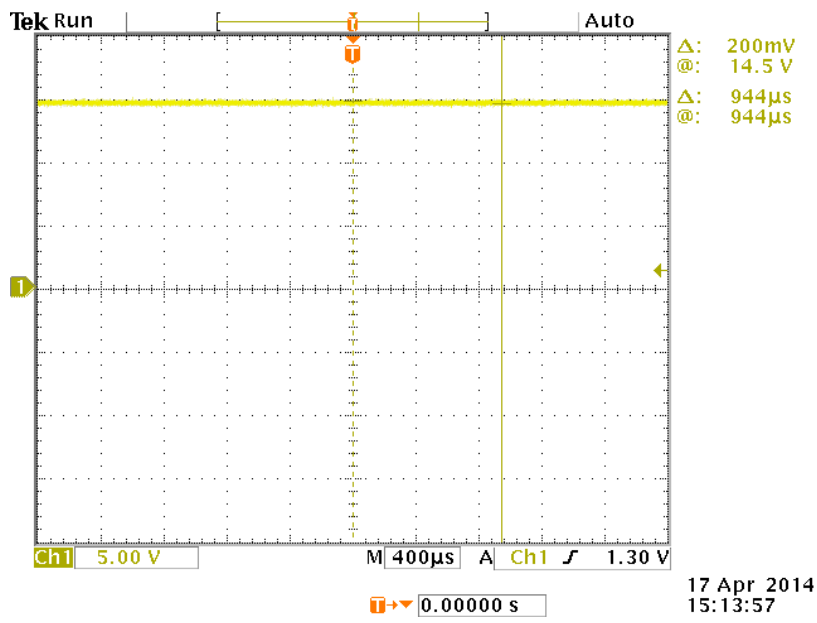


Figure 52: CRO measure of the positive 15V

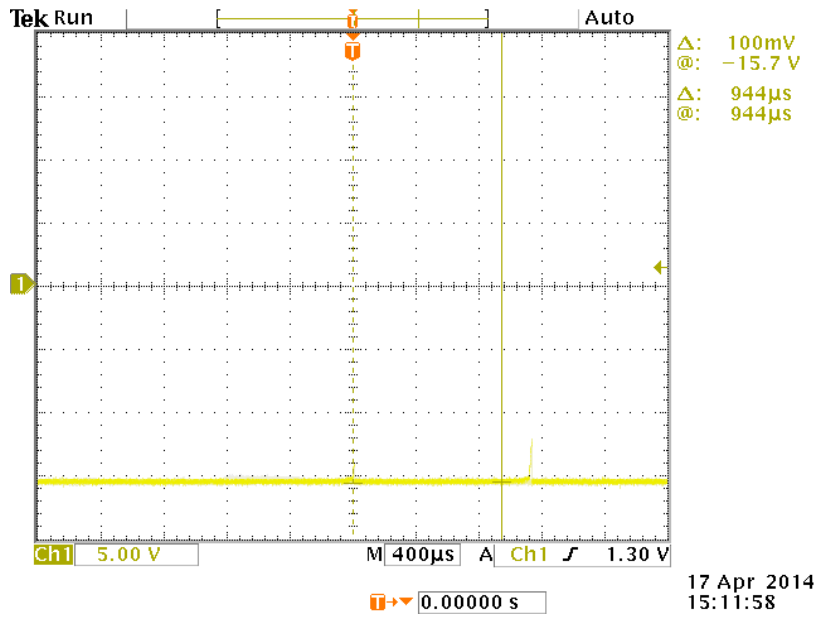


Figure 53: CRO measure of the negative 15V

4.2.2 DAC Board [23]

DAC Board is used as a bridge between the Arduino and the Maxon motor controller. For the ease of installation of the hardware devices to the chassis, DAC board is purposely design as an Arduino shield. In addition, the DAC board also provide a connection of MPU6050 to the Arduino controller board.

The Schematic design is divide into three parts. The first part is for the connection between Gyroscope and the Arduino controller board.

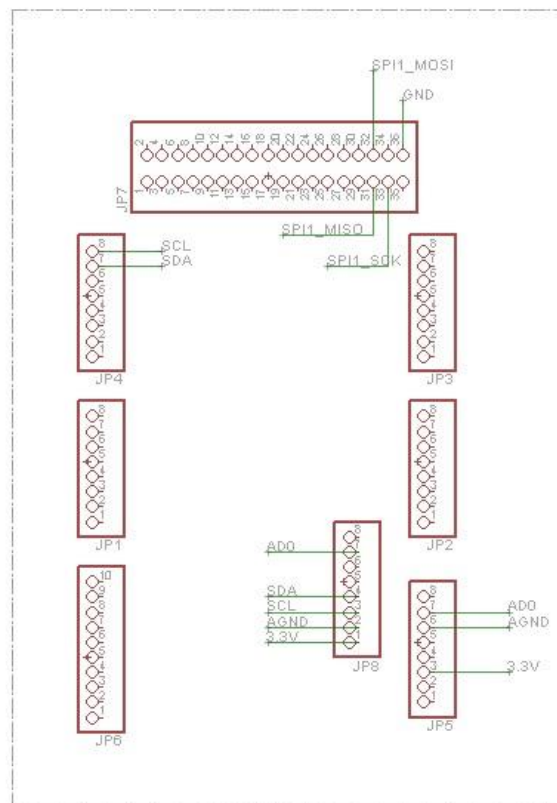


Figure 54: Schematic of DAC board part1

The second part is for the input power source.

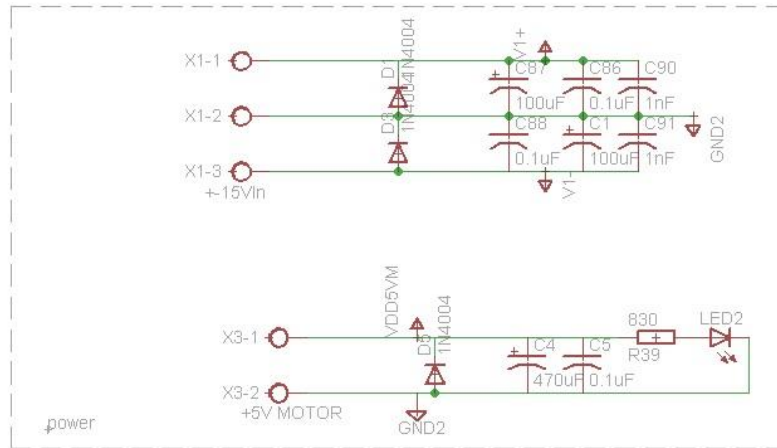


Figure 55: Schematic of DAC board part2

The third part is for the isolator ADUM1400 and the DAC AD5734R.

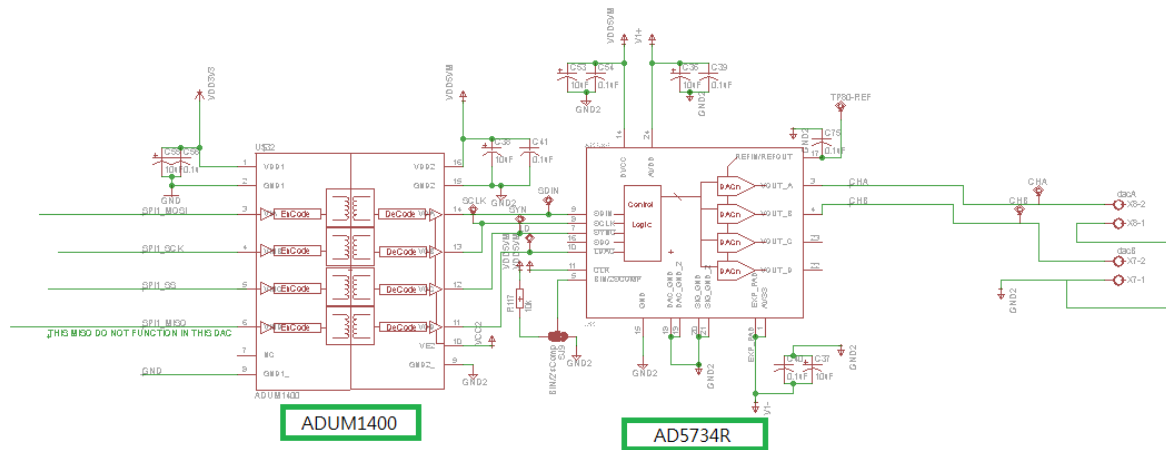


Figure 56: Schematic of DAC board part3

The layout of the board is than generated from the schematic. Below shows the final appearance of the layout with all components placed.

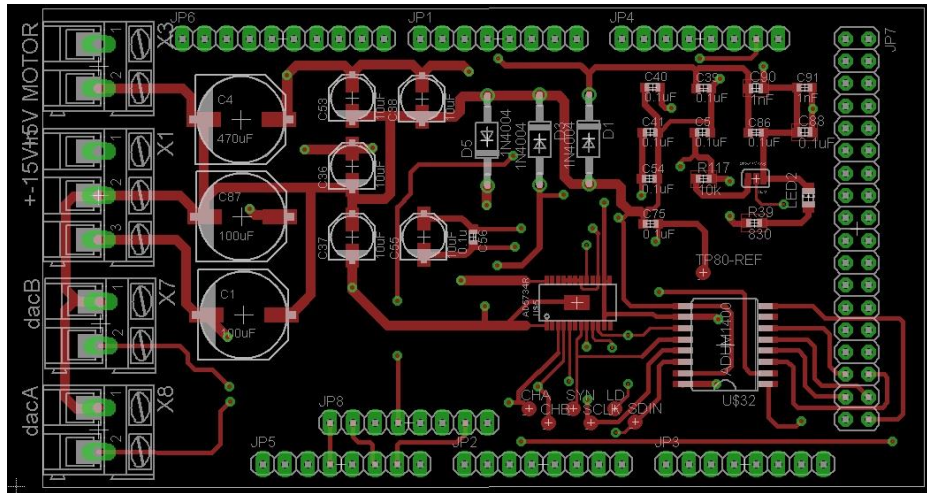


Figure 57: Upper layer layout of DAC board

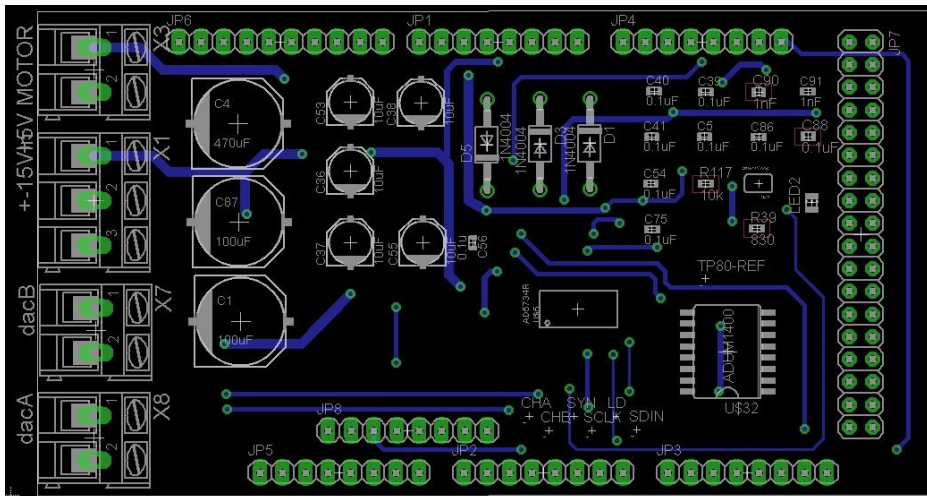


Figure 58: Bottom layer layout of DAC board

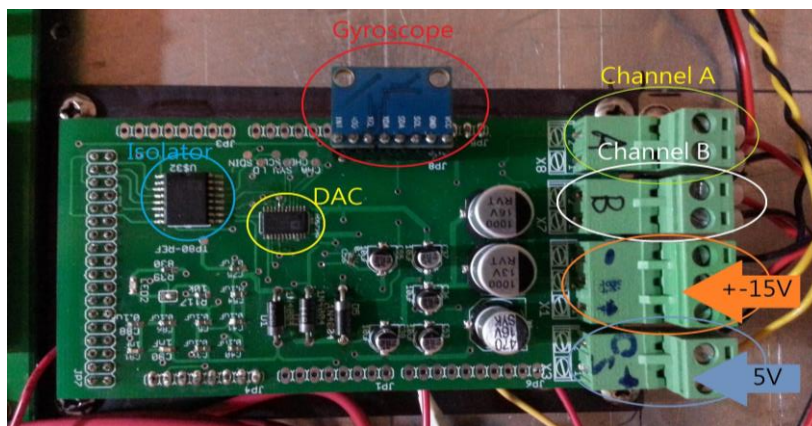


Figure 59: Soldered DAC board

4.3 Software implementation

In this chapter the implementation of the software for controlling the Segway is discussed. Arduino programming environment and the overview is discussed in Section 4.3.1. In Section 4.3.2 and section 4.3.3, the communication program SPI and i2c is examined. In Section 4.3.4, section 4.3.5 and section 4.3.6, gyroscope, Kalman filter and PID control is discussed. Section 4.3.7 discuss about variable resistor. The final software implementation on motor control is discussed in section 4.3.8.

4.3.1 Arduino and program overview

When Arduino is first start up, it need to initialize the components, the communication protocol and the mathematic algorithms.

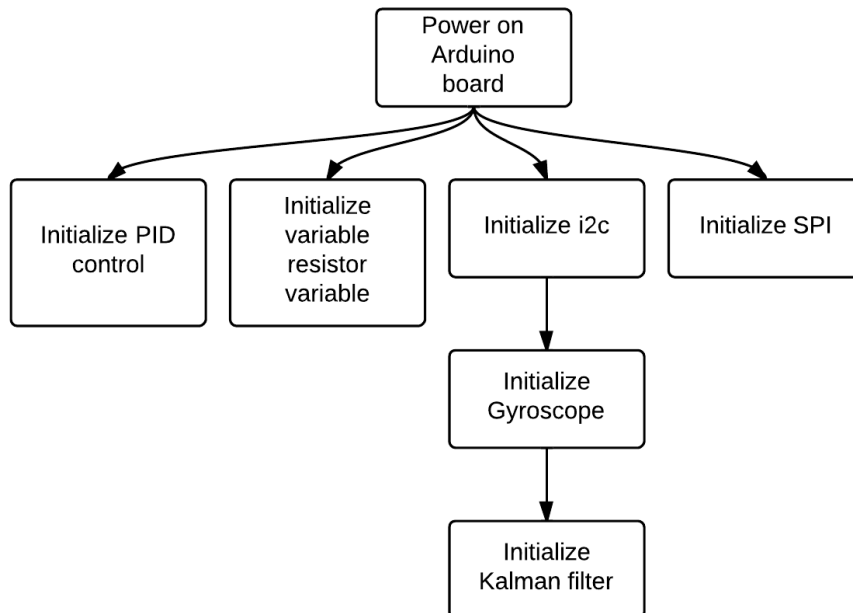


Figure 60: Flow chart for the initialization of Arduino

```

SPI_init
SPI_finish_init
Gyro_scrope_init
Gyro_scrope_end_init
dac_start_init
output range init
output range init
power control init
Dac_end_init
PID_init
PID_end_init
START time:12305264

```

Figure 61: Console output of initialization

After the initialization, the flow of program is shown below:

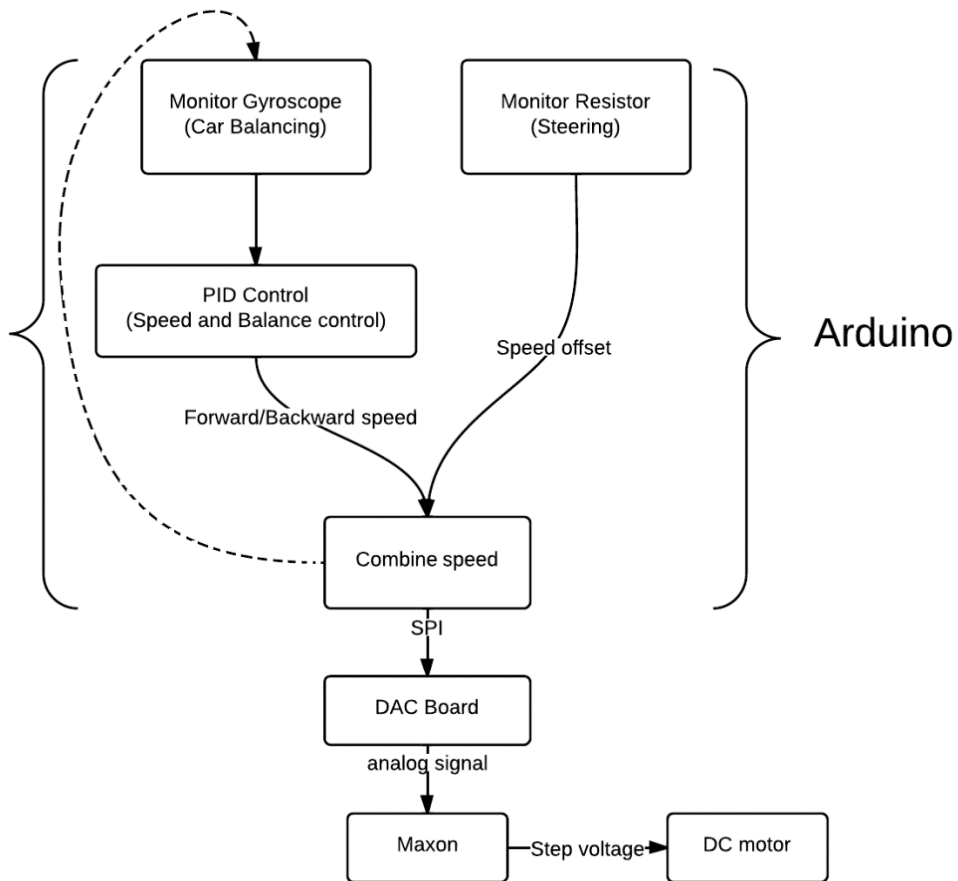


Figure 62: Program flow after initialization

4.3.2 SPI and DAC initialization

SPI communication is used in controlling the analog output of the DAC board.

Arduino board	MOSI	MISO	SCK	SS
Mega 2560	51	50	52	53

Table 5: SPI pins for Arduino controller board

Arduino Mega 2560 have the predefined pins for SPI protocol. The program need to first define the pins and set their nature in the setup. In the SPI initialization, the start-up signal level and SPI speed is also defined.

The SPI transmission can be achieve by:

```
byte SPI_Transmit(byte cData)
{
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while(!(SPSR & (1<<SPIF)))
    ;
    return SPDR;
}
```

Figure 63: Program for SPI transmit

The DAC board can then be reached though the SPI_Transmit function. To use this function, the SS pin must be pull low first. After the transmission of the data, the SS pin need to pull high.

```
delay(40);
digitalWrite(SS, LOW);
SPI_Transmit(0x0C);
SPI_Transmit(0x00);
SPI_Transmit(0x04);
delay(40);
digitalWrite(SS, HIGH);
```

Figure 64: Send data though SPI

DAC [24] is initialization with first sending a dummy packet follow by an output range select packet and a power control packet. These data are send though SPI, which is already initialized in the setup part.

OUTPUT RANGE SELECT REGISTER

The output range select register is addressed by setting the three REG bits to 001. The DAC address bits select the DAC channel, while the range bits (R2, R1, R0) select the required output range (see Table 22 and Table 23).

Table 22. Programming the Required Output Range

MSB											LSB
R/W	Zero	REG2	REG1	REG0	A2	A1	A0	DB15 to DB3	DB2	DB1	DB0
1/0	0	0	0	1	DAC address			Don't care	R2	R1	R0

Table 23. Output Range Options

R2	R1	R0	Output Range (V)
0	0	0	+5
0	0	1	+10
0	1	0	+10.8
0	1	1	±5
1	0	0	±10
1	0	1	±10.8

Figure 65: Output range Select register of AD5734R

In the project, output range of positive to negative 10V is used in controlling both forward and backward rotation of the motor. The data to send in output range select register is 0x0C0004.

POWER CONTROL REGISTER

The power control register is addressed by setting the three REG bits to 010. This register allows the user to control and determine the power and thermal status of the AD5724R/AD5734R/AD5754R. The power control register options are shown in Table 27 and Table 28.

Table 27. Programming the Power Control Register

MSB											LSB									
R/W	Zero	REG2	REG1	REG0	A2	A1	A0	DB15 to DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	0	1	0	0	0	0	X	OC _b	OC _c	OC _a	OC _a	0	TSD	PU _{usr}	PU _b	PU _c	PU _a	PU _a	

Figure 66: Power control register of AD5734R

This register is used in controlling the thermal status and the power of the AD5734R. According to the description of different data bits in the data sheet, the data that send in the power control packet is 0x10001F.

With the success in the initialization of AD5734R, a reference voltage of 2.5V should be achieved and this reference voltage is used to test whether the chip is working.

After the success in the initialization of the DAC board, analog output can then be controlled by sending the corresponding speed value to the corresponding channel through SPI.

4.3.3 I²C

I²C is used in communication between controller board and MPU6050. The I²C communication protocol can be waked up by calling the build in wire library. By using the wire functions, program of I²C read and I²C write are developed.

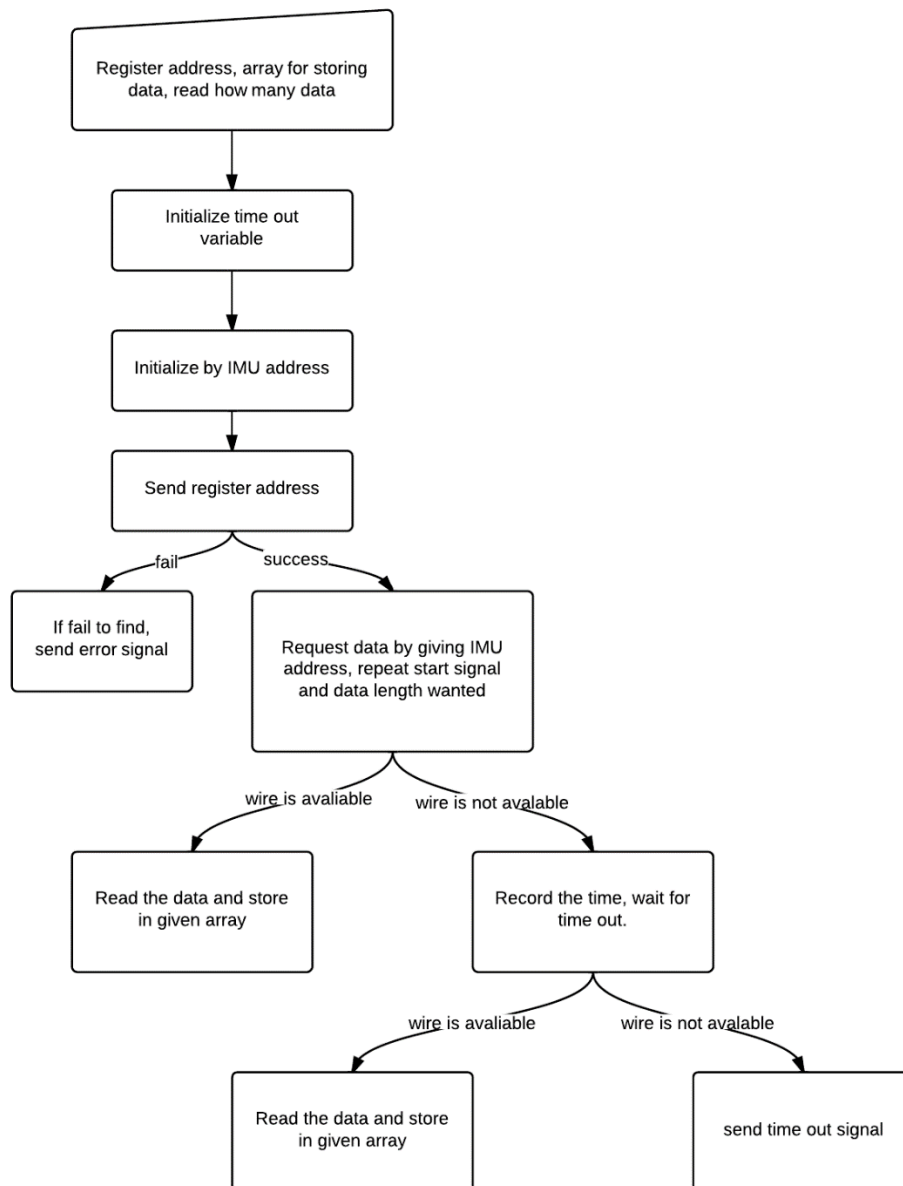


Figure 67: I²C read program flow chart

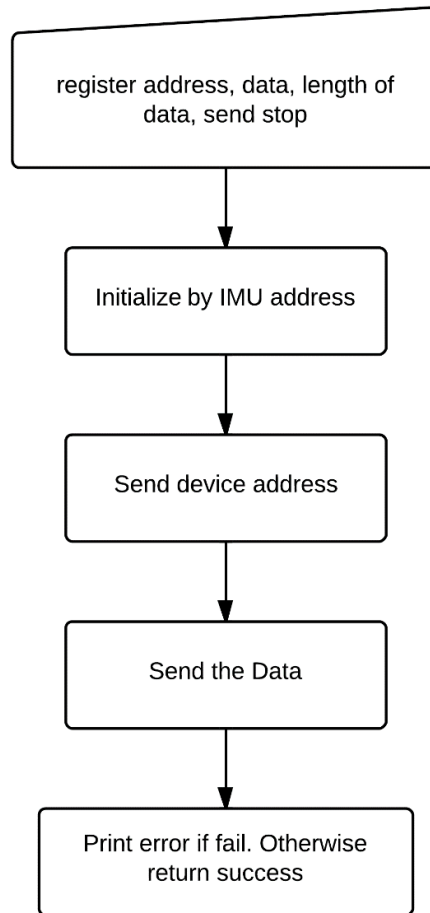


Figure 68: I²C write program flow chart

Arduino provide the default pins for SDA and SCL which will be waked up when calling the wire library. Programmer do not need to define the pins and pin modes for using I²C protocol.



Figure 69: I²C Pins in Arduino board

4.3.4 Gyroscope and accelerometer [25]

Gyroscope and accelerometer are the device that use to monitor the tilting angle of the Segway. They are the key issue in balancing control. The value read from the sensors are filtered by Kalman filter which is discussed in section 4.3.5. The filtered angle is then used to calculate the balancing force by PID algorithm which is discussed in section 4.3.6.

```
i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz/(7+1) = 1000Hz
i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc filtering, 256 Hz Gyro filtering, 8 KHz sampling
i2cData[2] = 0x00; // Set Gyro Full Scale Range to ±250deg/s
i2cData[3] = 0x00; // Set Accelerometer Full Scale Range to ±2g
```

Figure 70: Setting for gyroscope and accelerometer

According to the MPU6050datasheet, different parameters are initialized according to the project requirements.

```
while(i2cRead(0x3B,i2cData,14));
accX = ((i2cData[0] << 8) | i2cData[1]);
accY = ((i2cData[2] << 8) | i2cData[3]);
accZ = ((i2cData[4] << 8) | i2cData[5]);
tempRaw = ((i2cData[6] << 8) | i2cData[7]);
gyroX = ((i2cData[8] << 8) | i2cData[9]);
gyroY = ((i2cData[10] << 8) | i2cData[11]);
gyroZ = ((i2cData[12] << 8) | i2cData[13]);
```

Figure 71: Reading data from sensors

The data from both sensors can then be read consecutively and store in variables.

The Gyroscope:

The value read in from gyroscope is a number. It need to be translated in to something useful by the equation: $gyroRate = (gyroAdc - gyroZero) / sensitivity$.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	

Figure 72: Gyroscope sensitivity

```
double gyroXrate = (double)gyroX/131.0;
double gyroYrate = -((double)gyroY/131.0);
```

Figure 73: Program for calculation of gyro rate

The resulted value is the degree per second. The current angle can be calculated by adding the change in angle with the angle value in the last cycle. The change in angle can be calculated by multiplying the change of time with the gyroRate.

The gyroscope value drifts over time and it cannot be trusted for a long period, but it is precise for a short time.

The accelerometer:

It measures the acceleration in g of the three dimensions. The values need to translate into the angle value by using atan2 [26].

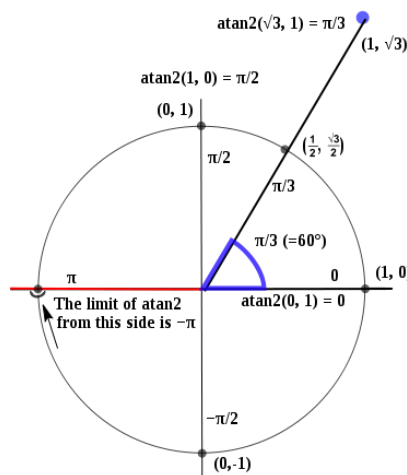


Figure 74: atan2 around a circle

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

Equation 3: arctan equations

The angle can be calculated by

```
accXangle = (atan2(accY,accZ)+PI)*RAD_TO_DEG;
accYangle = (atan2(accX,accZ)+PI)*RAD_TO_DEG;
```

Figure 75: Program for the calculations

RAD_TO_DEG is used in transforming the radians to degrees and it is a build in variable.

4.3.5 Kalman filter [27]

Tilting angle sensing is the key issue in this project. The angle read should be very precise otherwise accident may happen. Gyroscope value is very precise but tend to drift. The accelerometer is unstable but does not drift. By applying Kalman filter, the precise angle can be calculated. The estimation model consist of different equation and they will be implemented using C++ code.

0. Initialize the Kalman filter

To use a Kalman filter, a list of variables are need to be initialized.

```
/* Kalman filter variables */
double Q_angle; // Process noise variance for the accelerometer
double Q_bias; // Process noise variance for the gyro bias
double R_measure; // Measurement noise variance

double angle; // The angle calculated by the Kalman filter - part of the 2x1 state matrix
double bias; // The gyro bias calculated by the Kalman filter - part of the 2x1 state matrix
double rate; // Unbiased rate calculated from the rate and the calculated bias

double P[2][2]; // Error covariance matrix
double K[2]; // Kalman gain
double y; // Angle difference
double S; // Estimate error
```

Figure 76: Variable used in Kalman filter

```
Q_angle = 0.001;
Q_bias = 0.003;
R_measure = 0.03;

bias = 0; // Reset bias
P[0][0] = 0;
P[0][1] = 0;
P[1][0] = 0;
P[1][1] = 0;
```

Figure 77: Init value of variables

1. State prediction based on dynamic model.

The model is used in predicting the next state value based on the current state and the signals. The model can reduce the effect of noise and error.

$$\mathbf{x}_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k$$

Equation 4: State matrix

\mathbf{x}_k represent the state matrix with bias $\dot{\theta}_b$ and angle θ , they are the measurements from the gyroscope and accelerometer and bias is the amount the gyro has drifted.

To predict the next state, the equation is given by:

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\dot{\theta}_k \\ \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} &= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k \\ &= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k \\ &= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t + \dot{\theta} \Delta t \\ \dot{\theta}_b \end{bmatrix} \\ &= \begin{bmatrix} \theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}_b \end{bmatrix} \end{aligned}$$

Equation 5: Equation for prediction

F is a state transition model matrix is applied to the previous state and B is the control input model matrix.

The C code can be implemented as:

```

/* Step 1 */
rate = newRate - bias;
angle += dt * rate;

```

Figure 78: Program code for prediction step

2. To update the state covariance matrix

Matrix P measures how much we trust the current value. Small P value implies the filter is close to the real value. After model predictions have been done in step 1, P matrix has to be updated according to the uncertainties induced by the model noise w with the covariance Q. This is done with the equation

$$\mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$$

Equation 6: To update the state covariance matrix

T means that the matrix is transposed.

$$\begin{aligned}
\mathbf{P}_{k|k-1} &= \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}_k \\
\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} &= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t \\
&= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t \\
&= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t \\
&= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta \Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix} \\
&= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}
\end{aligned}$$

Equation 7: To update the state covariance matrix

After the transformation of the equation it can be written into C code as:

```

// Update estimation error covariance - Project the error covariance ahead
/* Step 2 */
P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);
P[0][1] -= dt * P[1][1];
P[1][0] -= dt * P[1][1];
P[1][1] += Q_bias * dt;

```

Figure 79: Program code for update the state covariance matrix

3. Compare reality and prediction

A new measurements arrive and the difference between it and the states. The equation is given by

$$\begin{aligned}
 \tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \\
 &= \mathbf{z}_k - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} \\
 &= \mathbf{z}_k - \theta_{k|k-1}
 \end{aligned}$$

Equation 8: Compare reality and prediction

With the gyroscope measure the angle based on the gravity vector and translate it using atan^2 (b, a). It will results in a measurement vector and the code can be implemented by

```

// Calculate angle and bias - Update estimate with measurement zk (newAngle)
/* Step 3 */
y = newAngle - angle;

```

Figure 80: Program code for comparing the values

4. Covariance update

S is representing the covariance matrix. It is calculated by following equation:

$$\begin{aligned} S_k &= \mathbf{H} \mathbf{P}_{k|k-1} \mathbf{H}^T + \mathbf{R} \\ &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \mathbf{R} \\ &= P_{00k|k-1} + \mathbf{R} \\ &= P_{00k|k-1} + \text{var}(v) \end{aligned}$$

Equation 9: Covariance update

The covariance matrix S is depending on the covariance predictions of the previous model. If value of S increased, it implies a low confidence in the value. On the other side if S decreased, it implies a high confidence and the data is more accurate.

```
/* Step 4 */  
S = P[0][0] + R_measure;
```

Figure 81: Program code for updating covariance

5. Kalman gain calculation

This step is going to merge the knowledge from previous calculated model and the measurements. It is achieved by a matrix named Kalman gain K. It weight the measurements and the model together.

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}^T \mathbf{S}_k^{-1} \\
\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k &= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{S}_k^{-1} \\
&= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} \mathbf{S}_k^{-1} \\
&= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{\mathbf{S}_k}
\end{aligned}$$

Equation 10: Calculate the Kalman gain

H matrix is used to map the state onto the measurements.

A large value of K implies there is low confidence in the model and a high confidence in measurement. A small value of K implies there is a high confidence in the model and a low confidence in the measurements.

```

// Calculate Kalman gain - Compute the Kalman gain
/* Step 5 */
K[0] = P[0][0] / S;
K[1] = P[1][0] / S;

```

Figure 82: Program code for calculate Kalman gain

6. Improvement of the model prediction

As the Kalman filter is a recursive loop algorithm, each time it need to update the prediction to achieve a better predict in the next loop. The difference between the prediction in step one and the real measurements is scaled with Kalman gain and added to the prediction.

$$\begin{aligned}\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} &= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{\mathbf{y}}_k \\ &= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \tilde{\mathbf{y}} \\ K_1 \tilde{\mathbf{y}} \end{bmatrix}_k\end{aligned}$$

Equation 11: to update the prediction model

The improvement rate is related to the Kalman gain value calculated in the previous step. In the previous step, if there is a small K matrix, it implies that there is a large confidence in the current model. As a result the change to the prediction model will be small from the equation.

```
/* Step 6 */
angle += K[0] * y;
bias += K[1] * y;
```

Figure 83: Program code for prediction model update

7. Using the new data to update the covariance matrix

The covariance matrix is updated after every sample has been taken and it is done by:

$$\begin{aligned}\mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_{k|k-1} \\ \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \\ &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \\ &= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}\end{aligned}$$

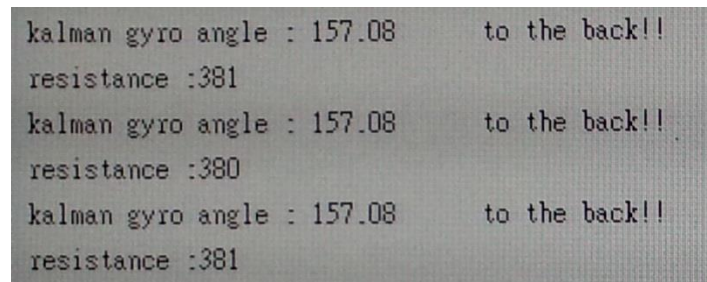
Equation 12: Update covariance matrix

P is scaled down with increased certainty of the state values after adding measurement knowledge.

```
// Calculate estimation error covariance - Update the error covariance
/* Step 7 */
P[0][0] -= K[0] * P[0][0];
P[0][1] -= K[0] * P[0][1];
P[1][0] -= K[1] * P[0][0];
P[1][1] -= K[1] * P[0][1];
```

Figure 84: Program code for covariance update

The Kalman filter algorithm will run from step one again with new sample is coming in.



```
kalman gyro angle : 157.08 to the back!!
resistance :381
kalman gyro angle : 157.08 to the back!!
resistance :380
kalman gyro angle : 157.08 to the back!!
resistance :381
```

Figure 85: console output for gyroscope readings

The gyroscope angle measurements tend to be stable with the use of Kalman filter.

4.3.6 PID control [28]

The PID equation is given below:

$$\text{Output} = K_p e(t) + K_i \int e(t) dt + K_d \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

Equation 13: PID equation

The value that proportional to the current error value can be produce by the term K_p . If the value of K_p is large, there will be a large change in the output for the change in error. If the value of K_p is small, the system will be less sensitive to errors.

The integral term K_i is the term proportional to the magnitude of error and its duration. It can be implemented simply by adding up the error over time, it gives the offset that should be corrected.

The derivative term represent the rate of change. It is used to predict the behaviour of the system by observation the rate.

Set point is representing the desired position of the Segway which is 180 degree. The input is the angle value filtered by Kalman filter. In the process error between the tilting angle and the upright position is calculated and it is used to drive the PID algorithm to modify the motor's speed. The tilting angle will then be changed to a closer position to the desired up right position.

A sampling time is set in the algorithm to let the PID algorithm functions runs in a regular interval. As direction is also an important issue in PID algorithm, in the implementation there is an input for programmer to set the direction of the model. It simply flip the value of the constant if the direction is reversed.

```
PID(double* Input, double* Output, double* Setpoint,
    double Kp, double Ki, double Kd, int ControllerDirection)
```

Figure 86: Function of PID algorithm

The process of PID calculation is shown in the following flow chart.

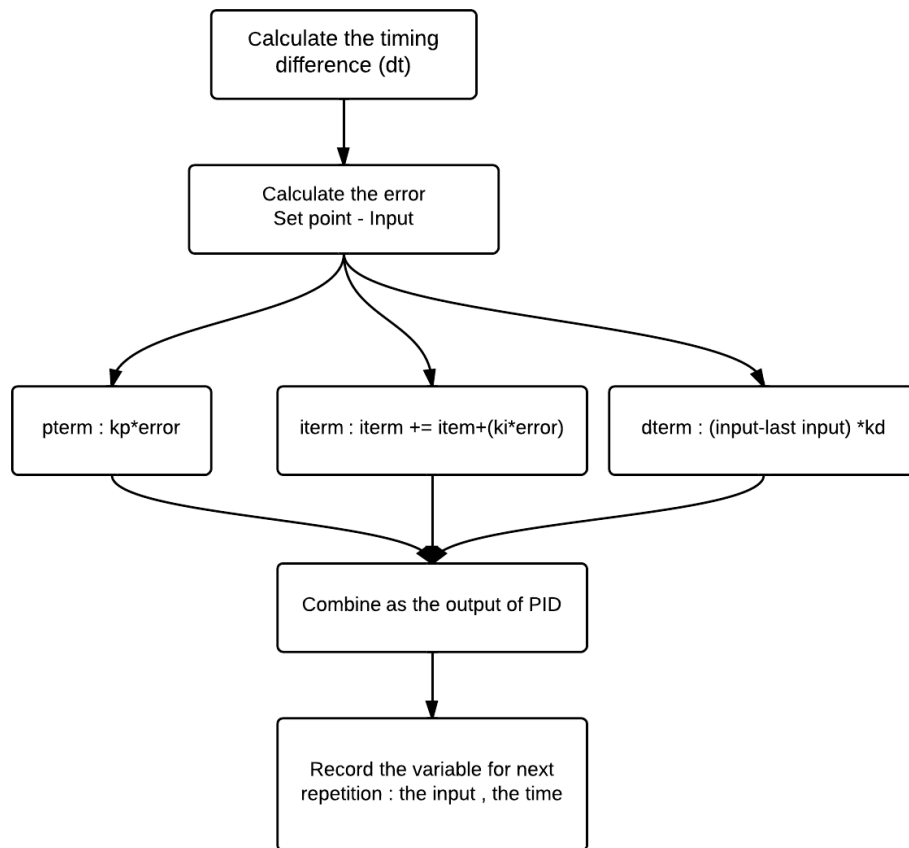


Figure 87: Flow chart for PID

The time dt is first calculated as the terms in the algorithm is related to time. The error between the input and the last input is then calculated. Each value is calculated and they are checked whether they are larger than the upper or the lower bound. If such thing happen, they are set to the upper bound or lower bound value. The three value are combined together and produce the output which representing the magnitude of difference between a measured

process and the desired position. The magnitude is further mapped to a speed value to get the forward or backward speed.

The three parameter are tuned based on the following table until the Segway can balance.

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
P	Decrease	Increase	Small change	Decrease	Degrade
I	Decrease	Increase	Increase	Eliminate	Degrade
D	Minor change	Decrease	Decrease	No effect	Improve

Table 6: P,I and D constant tuning

4.3.7 Variable resistor

The variable resistor is used to control the turning of the Segway. To read this sensor, there is an analog output pin in the resistor.



Figure 88: The variable resistor

```
resistance = analogRead(potpin); // Read the resistance
Serial.println(resistance);

//calculate magnitude of turning
if(resistance+4<stableresistance){
    right=(stableresistance-resistance)*0x25;
    left=-((stableresistance-resistance)*0x25);
}
else if(resistance-4>stableresistance){
    left=(resistance-stableresistance)*0x25;
    right=-((resistance-stableresistance)*0x25);
}
else{
    left=0;
    right=0;
}
```

Figure 89: Program for reading sensor and measuring the speed value

By reading the analog value of the resistor, the current resistance can be read. The magnitude of turning can be calculated by comparing with the stable resistance with a threshold. The magnitude is then scaled up with a factor so that it is significant to alter the speed to perform turning. The speed value is output together with the forward or backward speed.


```
// Motor control , k = forward or backward, left and right for turning
DAC_write(k+left,1);
DAC_write(k+right,2);
```

Figure 90: Program code for speed outputting

4.3.8 DAC board and motor control

The balancing speed calculated from the PID algorithm and the additional speed values calculated according to the variable resistor is combine together to form the final speed values. These values are send to the DAC board though SPI to achieve motor control

```
going to the front
it has been sent to right wheel: 11136
it has been sent to left wheel: 9360
```

Figure 91: Result of speed control

Chapter 5 Result and testing

5.1 System testing

In the first stage of the project two LED is used to simulate the two wheels for testing of the program. The brightness value original is at 60, the minimum value is 0 and the maximum value is 255. The brightness of the light bulb will represent the speed of the wheel, decrease in brightness implies moving backward and increase in the brightness represent moving forward.

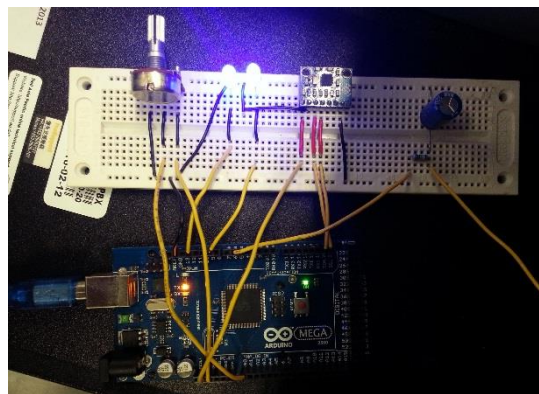


Figure 92: Hardware setup of the balancing system

```
kalman gyro angle : 180.88      stable!!  
resistance :939  
kalman gyro angle : 180.88      stable!!  
resistance :939  
kalman gyro angle : 180.88      stable!!  
resistance :939
```

Figure 93: Output of the above system

The system is at stable state when 180 degree. By applying the Kalman filter, the angle output is very stable and keep away from drifting. In this example, the center position of the resistor is set to be 939. The two LED have the same brightness level of 60 in the figure.

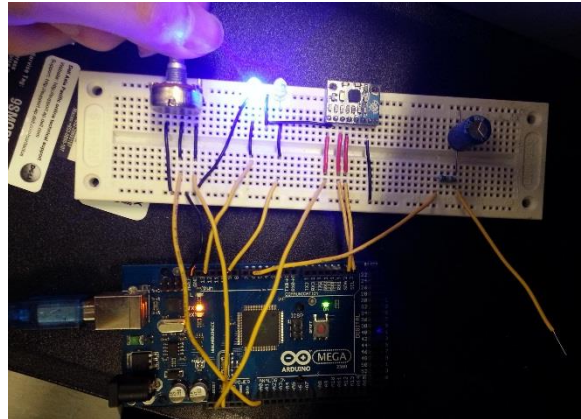


Figure 94: Increase the resistance value

```
kalman gyro angle : 180.89    stable!!  
resistance :1023  
to the right!!  
  
kalman gyro angle : 180.88    stable!!  
resistance :1023  
to the right!!  
  
kalman gyro angle : 180.88    stable!!  
resistance :1023  
to the right!!
```

Figure 95: Output result of increased resistance

A motion of turning the handle bar to the right is simulate by rotate the resistor clockwise. The resistance is increased and the lightness of the right LED is lower than the left one. It simulate the increase speed of left wheel and decrease speed on the right wheel, which will result in turning right

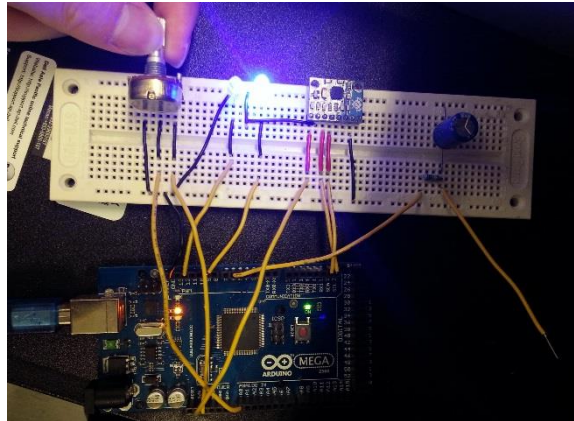


Figure 96: Decrease the resistance value

```

kalman gyro angle : 180.92      stable!!
resistance :729
to the left!!
kalman gyro angle : 180.92      stable!!
resistance :728
to the left!!
kalman gyro angle : 180.92      stable!!
resistance :729
to the left!!

```

Figure 97: Output result of decreased resistance

A motion of turning the handle bar to the left is simulate by rotate the resistor anti-clockwise. The resistance is decreased and the lightness of the left LED is lower than the right one. It simulate the decrease speed of left wheel and increase speed on the right wheel, which will result in turning left

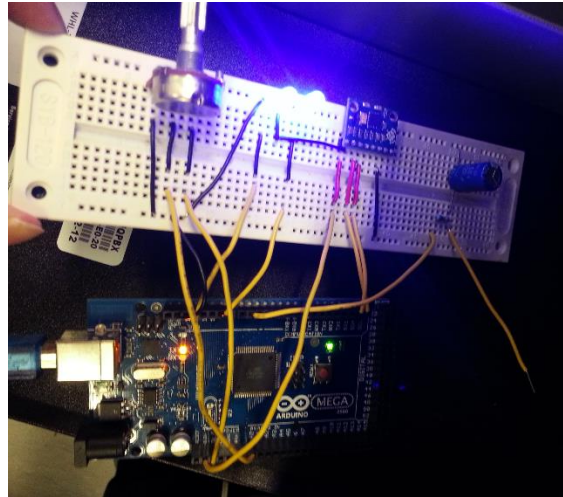


Figure 98: Simulation for going forward

```
kalman gyro angle : 198.50      to the front!!  
resistance :380  
kalman gyro angle : 198.49      to the front!!  
resistance :381  
kalman gyro angle : 198.48      to the front!!  
resistance :383  
kalman gyro angle : 198.45      to the front!!  
resistance :382
```

Figure 99: Output result of going forward

The change in tilting angle of car chassis is simulate by adjusting the tilting angle of the breadboard. The increase in the angle value simulate a user leaning forward, which will results in the Segway moving forward. Both LEDs brightness level will be increased.

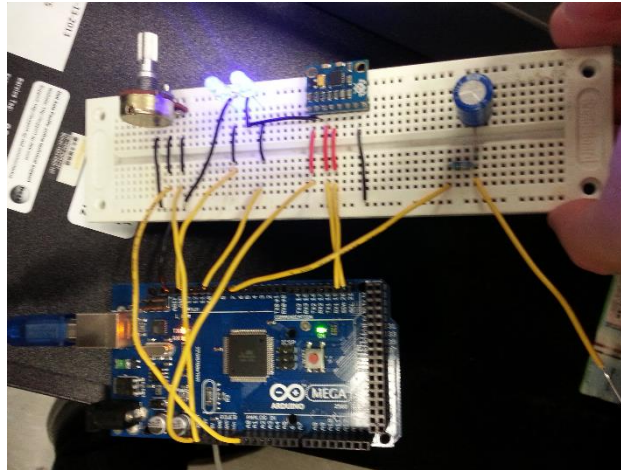


Figure 100: Simulation for going backward

```

kalman gyro angle : 157.08      to the back!!
resistance :381
kalman gyro angle : 157.08      to the back!!
resistance :380
kalman gyro angle : 157.08      to the back!!
resistance :381

```

Figure 101: Output result of going backward

In the opposite, decrease in the angle value simulate the user leaning backward. It will result in the Segway moving backward to balance the user and both LEDs brightness is decreased.

The below figures shows the combinations of turning left and right with going forward and backward.

```

kalman gyro angle : 195.64      to the front!!
resistance :1023
to the right!!

kalman gyro angle : 195.63      to the front!!
resistance :1023
to the right!!

kalman gyro angle : 195.60      to the front!!
resistance :1022
to the right!!

```

Figure 102: Output result of going right when moving forward


```
kalman gyro angle : 209.44      to the front!!  
resistance :163  
to the left!!  
kalman gyro angle : 209.43      to the front!!  
resistance :159  
to the left!!  
kalman gyro angle : 209.43      to the front!!  
resistance :162  
to the left!!
```

Figure 103: Output result of going left when moving forward

```
kalman gyro angle : 162.51      to the back!!  
resistance :428  
to the left!!  
kalman gyro angle : 162.51      to the back!!  
resistance :429  
to the left!!  
kalman gyro angle : 162.52      to the back!!  
resistance :426  
to the left!!
```

Figure 104: Output result of going left when moving back

```
kalman gyro angle : 164.48      to the back!!  
resistance :1023  
to the right!!  
kalman gyro angle : 164.47      to the back!!  
resistance :1023  
to the right!!  
kalman gyro angle : 164.47      to the back!!  
resistance :1023  
to the right!!
```

Figure 105: Output result of going right when moving back

5.2 How to ride the DIY Segway

The steps of using the DIY Segway is shown below:

1. Choose a flat and smooth area to start up your Segway. User should put on their helmet when riding this car.
2. Rider should first turn on the switch one for the power up of the system. User should wait until the LED lights up which indicate the success on initialization.
3. Rider should turn Switch three to car self-balance mode. After turning switch three, Switch two can be turn on for powering up of the motors.
4. The Segway is balanced and rider should test the Segway by moving the handlebars back and forth to see if the wheels respond.
5. Rider should step onto Segway with one foot and turn to user balance mode and step another foot onto the Segway.
6. Rider could lean forward to more the Segway forward and lean backward to travel backwards on the Segway. To turn the car, rider can shift the handlebar.
7. To step off the Segway, rider should first shift his weight to the centre to stop. Next, rider should step one foot off the Segway and turn switch three to car-self balance mode. Finally, rider can step off the car and close switch one and switch two consecutively.

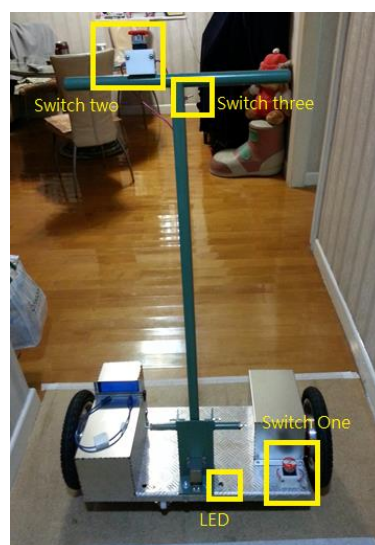


Figure 106: DIY Segway

5.3 Segway self-balance



Figure 107: Self balanced Segway

With the success in controlling the motors, the hardware system and the software system were integrated. The motors and computer in the base maintain the Segway balance during operation. In order to test the stability of the system, external force were given to the Segway to disrupt its stable state. It could rebalance itself under normal force and the settling time is very short. It was not able to rebalance in a few tests if a very large force was given to it.

5.4 Segway Balancing with user



Figure 108: Segway balancing in user mode

By turning the switch to user mode, a user can step on the car. User can command the Segway by shifting their weight forward or backward, the breaking distance is very short. User can either preform rotating or drifting by turning the handle bar to left or right. Rotation can be perform if the user turn the handlebar when the car is in stable state and drift can be perform if a user turn the handle bar while the car is moving forward or backward.

With the safety wheel, user feel more relax in using the Segway. Tests was carried out by letting different users to try to ride on the Segway. Some users were able to control the Segway after five trials.

5.5 Top speed of the Segway

The speed of the motor can be calculated by using the following equation:

$$Speed = \frac{(rpm) * (2\pi) * (\frac{diameter}{2})}{60s}$$

Equation 14: Speed equation

The rpm for the DC geared motor is 320 revolution per minute. The diameter of the wheel is 30cm. By the equation the maximum forward and backward speed it can reach is about 5m/s. To reach this speed, the DAC board channel A and B should output positive 10V for going forward or negative 10V for going backward. Below shows the corresponding speed value output form the control board:

Movement	Forward Maximum	Not moving	Backward Maximum
Voltage output	+10V	0V	-10V
Speed value	16383	8191	0

Table 7: DAC speed value

```

START time:137701092
Angle:180.03          PID:-0.24
Resistance:536
going to the front
it has been sent to right wheel: 8282
it has been sent to left wheel: 8282
END time:137729104
  
```

Figure 109: Console output of balance speed

The angle value for balancing is 180 degree. In this state, the motor should stop moving and the analog output signal measure form DAC board using CRO is about 0V.

```

START time:343979120
Angle:193.37          PID:-106.94
Resistance:538
going to the front
it has been sent to right wheel: 10194
it has been sent to left wheel: 10194
END time:344007472

```

Figure 110: Console output of forward speed

In this state, it is the maximum tilting angle and the maximum speed value used for going forward. The analog output signal measured from CRO is about 2.3V, by estimation the speed it can reach is 1.15 m/s.

$$\frac{2.3V}{10V} * 5m/s = 1.15m/s$$

$$\frac{10194 - 8191}{8191} * 5m/s = 1.22m/s$$

Equation 15: Calculation for forward maximum speed

In testing, it reached about 1m/s with a rider on the Segway. Considering the user weight and friction, the speed is reasonable.

```

START time:17642996
Angle:171.41          PID:68.74
Resistance:537
going to the back
it has been sent to right wheel: 6674
it has been sent to left wheel: 6674
END time:17670876

```

Figure 111: Console output of backward speed

In this state, it is the minimum tilting angle and the maximum speed value used for going backward. The analog output signal measured from CRO is about -2V, by estimation the speed it can reach is 1m/s.

$$\frac{-2V}{10V} * 5m/s = -1m/s$$
$$\frac{6674 - 8191}{8191} * 5m/s = -0.93m/s$$

Figure 112: Calculation for backward maximum speed

In testing, it reached about 0.8m/s for going backward with a rider on the Segway. It is also a reasonable speed.

Chapter 6 Discussion

In this project only one mainboard is used to perform all the calculations, signal input and signal output.

6.1 SPI signalling

SPI is used in speed value signalling, the delay in SPI causes the two motor not synchronized.

```
delay(5);  
digitalWrite(SS, LOW);  
SPI.transfer(v1);  
SPI.transfer(v2);  
SPI.transfer(v3);  
delay(5);  
digitalWrite(SS, HIGH);
```

Figure 113: SPI program

This part of program will run consecutively in order to drive the two motor. One motor is start early than another for 15ms, this results in a slightly turning in a long run. Although the effect is not significant, it can be improved by using multiple controllers. Each controller will response for one set of motor so that the two motor can be commanded in a synchronised mode.

6.2 Sampling time of PID

The processing power of Arduino is very good, the reading of sensors and calculation of PID can perform very fast. However due to the delay in SPI signalling, the process is forced to stop.

```
START time:137701092
Angle:180.03          PID:-0.24
Resistance:536
going to the front
it has been sent to right wheel: 8282
it has been sent to left wheel: 8282
END time:137729104

START time:137731144
Angle:180.05          PID:-0.36
Resistance:536
going to the front
it has been sent to right wheel: 8282
it has been sent to left wheel: 8282
END time:137759160

START time:137761200
Angle:180.06          PID:-0.50
Resistance:537
going to the front
it has been sent to right wheel: 8282
it has been sent to left wheel: 8282
END time:137789224

START time:137791264
Angle:180.08          PID:-0.61
Resistance:537
going to the front
it has been sent to right wheel: 8282
it has been sent to left wheel: 8282
END time:137819276
```

Figure 114: Consecutive output during operation

The time frame is shown in the above figure. The process of calculation and motor control require 28ms to complete. The process can perform 35 times in one second which is not the best sampling rate. The Segway could be more responsive to the change in tilting angle if the sampling frequency is higher.

Chapter 7 Further development

There are several changes that could be made to increase the performance and functionality of the Segway. As this was the first version of Segway, it was considered infeasible to address these issues in the allowable time frame.

- Dynamic variation of the control system based on tilt angle and current battery capacity would provide a safer and more consistent ride.
- Implementing a wireless communication for the main board, so as to monitor the operation during operation for more precise tunings.
- Reducing the cost of Segway could be achieved by using a cheaper motor and motor controller
- The delay of the two wheels can be eliminated by using multiple micro-controller. Two main board could be used and each main board is responsible for controlling one set of motor.
- Reduce of the cost of main board could be achieved by self-designing a PCB that integrated the main controller, DAC and MPU6050. As the I/O is quite adequate, more sensors and components could be added on the PCB to increase the functionality.
- Encoder could be added to enhance the feedback control.
- A panel could be implemented to provide user with more information. For example the battery capacity, the current speed and the mode of the Segway.
- Warning signals can be added for increasing of safety
 - Distance sensors can be used for helping the user to maintain a safe distance when riding the Segway
 - Over speed warning should be given to the user, encoder can be used to calculating the speed of the car.
 - User should avoid obstacles when using the Segway, sensors can be used to detect obstacles and notice the rider to prevent collision.

Chapter 8 Budget

Electronic

Arduino mega 2560	\$295
10K variable resistor	\$0
GY521 MPU6050 3axis gyroscope and accelerometer	\$0
Maxon motor driver(x2)	\$0
DC Motor	¥590
Bearing house	¥100
Metal chain	¥35
Sprocket wheel	¥74
PCB fabrication	\$196
24V 350W Battery	\$671
Electronic Total	\$2161

Raw Material

Steel plate for base	\$0
Spring(x2)	\$0
Wheels(x2)	\$0
Steel column for hand-bar	\$0
Misc tooling	\$0
Raw Material Total	\$0

Material and Service

DC plug for Arduino mega2560	\$5
Transportation cost and shipping service	\$283
LEDs	\$21
9V battery	\$20
Raw Material Total	\$329
Grand Total	\$2490

Chapter 9 Conclusion

The aim of this project was to design and build a rideable, self-balancing and coaxial scooter, which it is called “The Segway”. This aim has been achieved and a Segway with control and on-board power has been made and successfully tested.

A comprehensive literature review was conducted, including software and technical information relevant to the project.

Mathematical models and algorithms like Kalman filter and PID control is used to compute the calculation. In one way, it can make the result more precise so as to achieve a better balancing. It can also increase the safety, to prevent the errors in calculations which will affect the speed of motors.

A formulated design approach was used to create the most efficient and robust configuration to satisfy all the project goals. The structural design was considered concurrently with component selection, aesthetics, and ergonomics to minimise mechanical, electrical and rider integration problems. The use of safety wheels and control switch provide a high safety and ease of control to the Segway.

The outcome of this project has been a mechanically sound, aesthetically pleasing and easy to ride self-balancing Segway.

References

- [1] The Telegraph - What is a Segway?
<http://www.telegraph.co.uk/motoring/news/8028753/What-is-a-Segway.html>
- [2] Reference Manuel of Segway PT
<http://www.segway.com/downloads/pdfs/ReferenceManual.pdf>
- [3] Wikipedia – Segway polo
http://en.wikipedia.org/wiki/Segway_polo
- [4] City Segway Tours
<http://citysegwaytours.com/>
- [5] Case study- Enhancing the Efficiency at the Airport Border Control
<http://www.segway.com/downloads/pdfs/BCIAairportcasestudy.pdf>
- [6] Segway X2 golf
<http://www.segwayforsale.net/segway-x2-golf-for-sale/>
- [7] Wikipedia – Segway PT
http://en.wikipedia.org/wiki/Segway_PT
- [8] Regulation chapter 374
http://www.td.gov.hk/filemanager/tc/content_1178/motorcycle.pdf
- [9] eHow – How do Gear Motors Work
http://www.ehow.com/how-does_4969768_gear-motors-work.html
- [10] Wikipedia – Lead-acid battery
http://en.wikipedia.org/wiki/Lead%E2%80%93acid_battery
- [11] 24V Lithium Iron Phosphate battery
<http://item.taobao.com/item.htm?spm=a230r.1.14.5.EyXtra&id=38023721809>
- [12] Arduino Mega2560
<http://arduino.cc/en/Main/arduinoBoardMega2560>
- [13] InvenSense - MPU6050 MEMS motion tracking
<http://www.invensense.com/mems/gyro/mpu6050.html>
- [14] Wikipedia – Resistor
<http://en.wikipedia.org/wiki/Resistor>

- [15] Wikipedia – Digital to analog converter
http://en.wikipedia.org/wiki/Digital-to-analog_converter
- [16] I2C background
<http://www.totalphase.com/support/articles/200349156-I2C-Background>
- [17] Introduction to i2c and SPI protocols
<http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/>
- [18] Arduino Software
<http://arduino.cc/en/main/software>
- [19] Wikipedia – Kalman filter
http://en.wikipedia.org/wiki/Kalman_filter
- [20] Wikipedia – PID controller
http://en.wikipedia.org/wiki/PID_controller
- [21] Eagle PCB software
<http://www.cadsoftusa.com/eagle-pcb-design-software/product-overview/?language=en>
- [22] Tarco power – TEN 25 Series
http://www.tracopower.com/fr/overview/ten25/?tx_mkanydropdownmenu_pi1=TEN%2025%20-----%20%28old%20design%29&cHash=3ae45fc8306c1cf7614ae3b064ded6f
- [23] ADUM1400 datasheet
http://www.analog.com/static/imported-files/data_sheets/ADuM1400_1401_1402.pdf
- [24] AD5734R datasheet
http://www.analog.com/static/imported-files/data_sheets/AD5724R_5734R_5754R.pdf
- [25] MPU6050 product Specification
<http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf>
- [26] Wikipedia – atan2
<http://en.wikipedia.org/wiki/Atan2>
- [27] Practical approach to Kalman filter and how to implement it

<http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>

[28] PID introduction

<http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>