" I hereby declare that I have read through this report entitle "Design, Construction and Analysis of Biped Robot for Walking Motion" and found that it has comply the partial fulfillment for awarding the degree of Bachelor of Mechatronics Engineering.

Signature : ……………………………..

Supervisor's Name : ……………………………..

Date : ……………………………..

DESIGN, CONSTRUCTION AND ANALYSIS

OF BIPED ROBOT

FOR WALKING MOTION

JIM KAI YIAT

A report submitted in partial fulfillment of the requirements for the degree

of Mechatronics Engineering

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2013/2014

I declare that this report entitle "Design, Construction and Analysis of Biped Robot for Walking Motion" is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : ……………………………..

Name : ……………………………..

Date : ……………………………..

ACKNOWLEDGEMENT

Many people have helped and guided me through the whole process of preparing this report. I am much indebted and thankful. In particular, I would like to thank my supervisor Dr Fariz Bin Ali @ Ibrahim, for his patience, teachings, encouragement and motivation in making this report possible. I would also like to thank Cik Nur Maisarah Bt Mohd Sobran and Encik Tarmizi Bin Ahmad Izzuddin for sharing their knowledge in solving my problems. Without any of their helps, this project would not have been the same as presented here.

Besides, all of my fellow friends that have accompanied and advised me are much appreciated. Their point of view and critics are the fundamental to the improvement of my project. Finally, I feel grateful to all the people who have helped even if it is just a small favor. I will also like to express my humble apology to everyone for the trouble I may have caused.

ABSTRACT

This report emphasize on the design, construction and analysis of biped robot for walking motion. This study is important since humanoid robots have been developed for a long period of time which benefits human in terms of aiding orthosis and prosthesis, understanding human body, creating human assistance and improving entertainment field. Therefore, development of biped robot would be the first step in completing this approach. Besides, the objectives of this project are to design and develop a biped robot with humanlike motion capability controlled by Arduino microcontroller and to analyse the biped robot walking motion by static motion sequence and joint angle. The literature review focuses on the design and basic construction of the biped robot including the walking analysis. Similarly, the execution of the project is separated into biped robot construction in FYP1 and analysis in FYP2. Moreover, the methodology of this project is to design the robot structure using SolidWorks, design the electrical and electronic configuration using Proteus ISIS, stability study and controller application. Parameters such as servo motor angle of rotation, center of mass and static walking motion are obtained and analysed in terms of the capability to achieve stable forward walking motion. Error analysis and trajectory analysis will be carried out to see the performance of the biped robot walking motion as compared to the actual human walking motion. Lastly, the outcome of this project is a functional biped robot with capability of walking forward in a smooth surface using static walking motion.

ABSTRAK

Laporan ini memberi penekanan kepada reka bentuk, pembinaan dan analisis robot "biped" untuk pergerakan berjalan. Kajian ini adalah penting kerana robot "humanoid" telah dibangunkan untuk tempoh masa yang panjang serta memberi manfaat kepada manusia dari segi pembangunan "orthosis" dan "prosthesis" , pemahaman tubuh manusia , pembentukan sistem bantuan manusia dan peningkatan bidang hiburan. Oleh itu, pembangunan robot "biped" akan menjadi langkah pertama dalam menyelesaikan pendekatan ini. Selain itu, objektif projek ini adalah untuk mereka bentuk dan membangunkan robot "biped" dengan keupayaan untuk bergerak sama dengan pergerakan manusia dengan kawalan mikropengawal Arduino dan untuk menganalisis robot "biped" berjalan dalam turutan pergerakan statik beserta sudut sendi . Kajian sastera memberi tumpuan kepada reka bentuk dan asas pembinaan robot "biped" dan juga analisis pergerakan berjalan. Lebih-lebih lagi , pelaksanaan projek ini dibahagikan kepada pembinaan robot "biped" dalam PSM1 dan analisis dalam PSM2 . Selain itu, metodologi projek ini meliputi reka bentuk struktur robot menggunakan program SolidWorks , reka bentuk konfigurasi elektrik dan elektronik menggunakan Proteus ISIS , kajian kestabilan dan penggunaan "pengawal" . Ciri-ciri seperti sudut putaran motor servo, pusat jisim dan gerakan berjalan statik akan diperolehi dan dianalisis dari segi keupayaan untuk mencapai pergerakan ke hadapan dalam keadaan berjalan stabil. Analisis ralat dan analisis generasi trajektori akan dilakukan untuk melihat keupayaan robot "biped" dalam pergerakan berjalan berbanding pergerakan berjalan manusia. Akhir sekali , hasil daripada projek ini adalah sebuah robot berkaki dua yang mampu untuk berjalan ke hadapan di permukaan yang licin dengan pergerakan statik.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

M	-	Mass
L	-	Length
θ	-	Angle
X_{COM}	-	Length of COM from reference point
θ_0	-	Initial angle
θ_v	-	Via point angle
θ_g	-	Final angle/ goal angle
H_R	-	Hip of right leg
K_R	-	Knee of right leg
A_R	-	Ankle of right leg
H_L	-	Hip of left leg
K_L	-	Knee of left leg
A_L	-	Ankle of left leg
T	-	θ

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Development of humanoid robot has been around for a long time. It has been used as an experiment tool in many scientific environments.

Initially, humanoid robots are developed for orthosis and prosthesis research purposes. This can be seen in the development of human orthotics and prosthetics equipments such as motorized leg and arm for neural or muscle impaired patient, biological realistic prosthetics, ankle to foot orthotics tool and much more. This is important because the future demand for orthotic and prosthetic services is expected to increase tremendously according to a study carried out by Caroline Nielsen. The exact statistics can be seen in Appendix A. Besides, based on an article published in 2009 [1], there is a total of 125000 patients in needs for prosthetic and orthotics services which is deemed to increase throughout the years.

Secondly, humanoid robots are also built so that the human body structure and behavior (biomechanics) can be learned. This leads to simulation of human body for further understanding and cognition study where human ability to analyse sensory information is being discussed for obtaining intuitive and motor techniques.

Moreover, human assistance is also made possible with the improving technology on humanoid robots. In this modern era, humanoid robots are being modified into a robot that can perform human tasks such as assisting the sick and elders, doing dangerous task such as fixing electrical cable, space exploration and much more. Theoretically, since humanoid is built in a form of a human body, it can basically do any task human are capable of, with a suitable algorithm.

Finally, the gaining popularity of humanoid robot in the entertainment field has brought to its development for the same purpose. For example, Ursula is a female robot capable of singing, dancing and interacting with audience at the Universal studios. Ursula has been developed by the Florida Robotics [2]. These robots have realistic expression and gestures that are comparable to human but they do not have cognition or physical anatomy.

These factors have contributed to the interest in understanding humanoid robot and biped robot development would be a beginning of it.

1.2 Problem Statement

Vast theoretical knowledge in robotics field is needed in designing and constructing a biped robot. These robots are controlled mainly by servo motors, microprocessors and control system. To achieve stable static motion, the robots design must achieve stability during stationary and moving motion, not forgetting the kinematics and dynamics of the robot which plays an important part in robot structure development. In current studies, all of these are possible; hence, the focus of this project is more to the implementation of the microprocessor which is the Arduino, the basic knowledge and skills in developing a biped robot (e.g. servo motor control, kinematics and stability derivation) and most importantly the analysis on the walking motion. The analysis includes the different walking combination implementation to find the best length of stride, height of stride and speed of stride for the walking motion. Besides, there is also error analysis to measure the accuracy of the Dynamixel AX-12A servo motor. Lastly, the trajectory generation analysis is conducted to compare the assumed walking motion of a human with the walking experimental walking motion of the biped robot. This analysis is done to obtain a better understanding on imitation of a human walking motion in a biped robot.

1.3 Objectives

1. To design and develop a biped robot with humanlike motion capability controlled by Arduino microcontroller.
2. To analyse the biped robot walking motion by static motion sequence and joint angle.

1.4 Scopes

1. The robot have 12 D.O.F with 3 axis of rotation (roll, pitch, yaw) and Dynamixel AX-12 servo motor capable of having large angle of rotation that compromises human movement.
2. Arduino is used as the main microcontroller and built-in PID is used as the controller.
3. Biped robot should only move/walk in forward motion by implementing static movement.
4. The frame and parts of the robot will be consisting of plastic/polymer that is lightweight, cheap and robust.
5. Only left leg of the biped robot is developed and the right leg will be developed by another course mate. This is due to the budget limitation and complexity.

CHAPTER 2

LITERATURE REVIEW

2.1 Design and Basic Construction

By using the measurements, dimension and degrees of freedom (D.O.F) of the robot "Denise" which was developed by Delft University [3], a similar robot was built for research use [4]. The robot has a total of 5 degrees of freedom with 1 degree of freedom at the hip, 2 degrees of freedom at the knee and 2 degrees of freedom at the ankle. Besides, it is made of mainly aluminum material. The main actuator used [4], is the pneumatic muscle or the McKibben muscle which expand its size with the increase of pressure causing the muscle to shorten and produces a force to the pneumatic piston attached to the hip. This in turn collaborates with the interlock system in the knee to remain fixed when the leg swings forward and bent during forward movement. Lastly, the author proposes the use of 16F84A microcontroller which is programmed using C language to control the robot. The microcontroller works simply by detecting the signal given by the sensors mounted on the sole and take appropriate action based on the given programming. The finished prototype is shown in Figure 2.1.

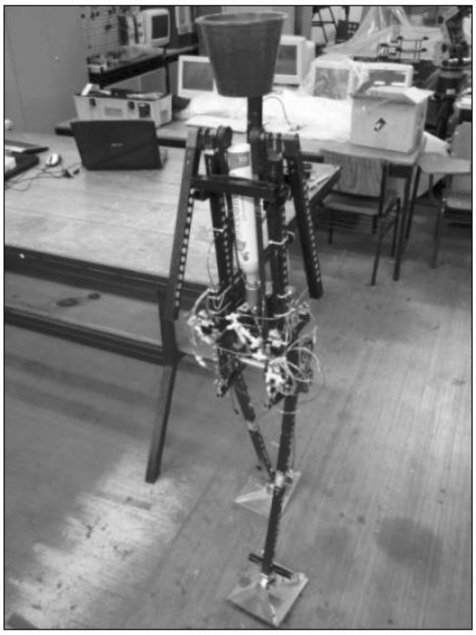


Figure 2.1: "Denise" prototypes [4]

On the contrary, Kanagawa Biped Robot-1 Refined or KBR-1R [5] has a total of 12 degrees of freedom with 6 degrees of freedom at the hip, 2 degrees of freedom at the knee and 4 degrees of freedom at the ankle. The author emphasize that the movable angle of KBR-1R is about the same as a human so that it could simulate the movements similar to a human. Besides, the material primarily used in this robot is aluminum. The D.O.F configuration is shown in Figure 2.2.

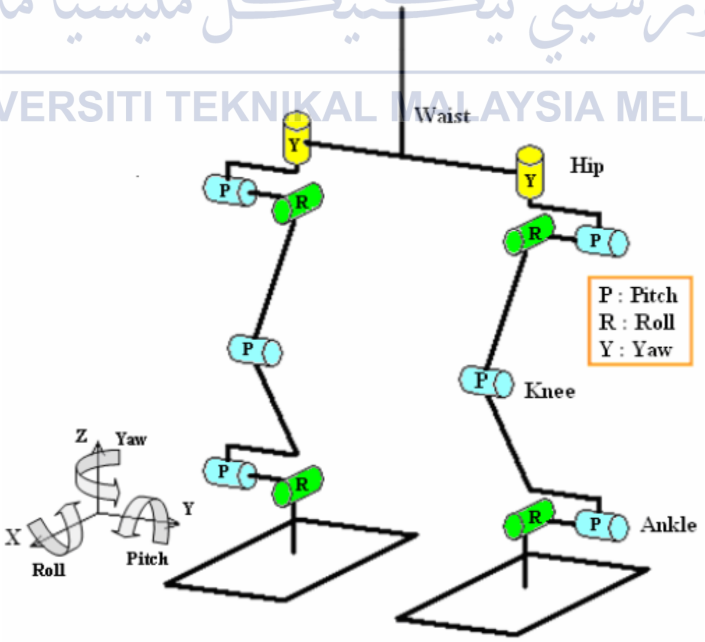


Figure 2.2: D.O.F configuration of KBR-1R [5]

The actuator used in KBR-1R is DC servo motors with the use of timing belts, pulley and harmonic drive gears in the joints to make it smaller. Finally, PC/AT compatible CPU with RT-Linux OS programmed with C language is proposed [5]. The control software has real time module capable of direct communication with the biped robot and non-real time module capable of channeling the input response or output walking pattern to the real time module. The KBR-1R prototype is shown in Figure 2.3:

Figure 2.3: KBR-1R prototype (frontal and lateral view) [5]

Meanwhile, Saika-4 [6] has a total of 30 degrees of freedom with 2 degrees freedom at the head, 14 degrees of freedom at the arm, 12 degrees of freedom at the leg and 2 degrees of freedom at the hand. Since aluminum alloy is light, strong and cheap, it is chosen as the material for the mechanical components while Al-Cu-Mg alloy is used for mechanical pieces and Al-Mg alloy for the outer shell. The D.O.F configuration of Saika-4 is shown in Figure 2.4.

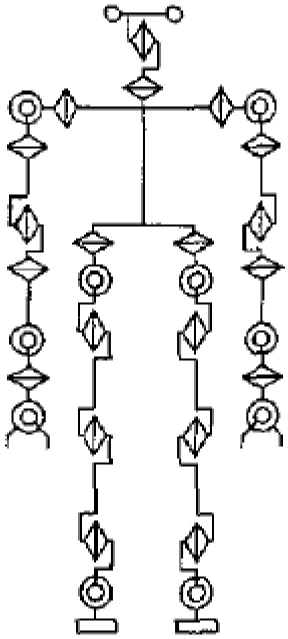


Figure 2.4: D.O.F configuration of Saika-4 [6]

Each joint has a DC servo motor, harmonic drive reduction gear and rotary optical encoder connected with synchronous belt driver and pulley to reduce the space or weight. The main actuator and microcontroller used is the DC servo motor and IBM PC/AT clone with QNX Realtime Platform OS [6]. The microcontroller functions similar to a generic PC. The Saika-4 CAD drawing is shown in Figure 2.5.

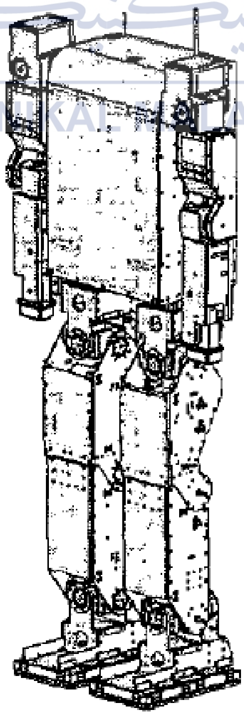


Figure 2.5: Saika-4 CAD drawing [6]

Furthurmore, WABIAN-2 human-like walking robot has a total of 16 degrees of freedom [7]. 2 degrees of freedom is located at the waist, 6 degrees of freedom located at the hip, 2 degrees of freedom located at the knee and 6 degrees of freedom at the ankle. The D.O.F configuration is shown in Figure 2.6.

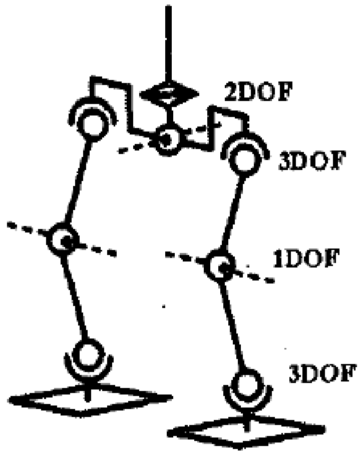


Figure 2.6: WABIAN-2 D.O.F configuration [7]

Besides, duralumin is used as the main material in WABIAN-2 construction. Each joint of WABIAN-2 consists of a DC servo motor, harmonic drive gear, a lug belt and two pulleys to allow high reduction ratio and separation between joint axis and motor axis [7]. Lastly, the microcontroller used in this robot is a PC consisting of PCI CPU board. The WABIAN-2 robot is shown in Figure 2.7.

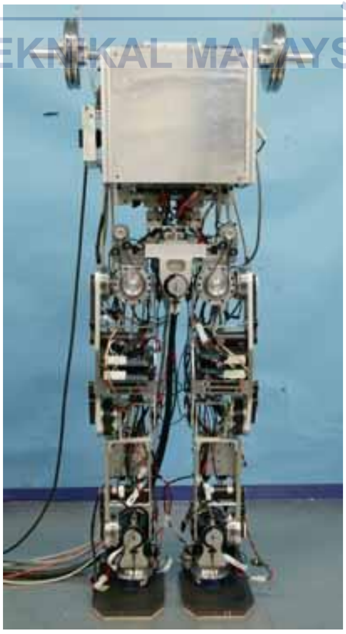


Figure 2.7: WABIAN-2 robot [7]

Table 2.1: Summary of Design and Construction

	Journal 1	Journal 2	Journal 3	Journal 4
Title	Development of a Biped Robot Based on Dynamic Walking	Development of a Biped Walking Robot	Design and Development of a Light-Weight Biped Humanoid Robot Saika-4	Development of a Human-like Walking Robot Having Two 7-DOF Legs and a 2-DOF Waist
Name of Robot	Denise	Kanagawa Biped Robot-1 Refined (KBR-1R)	Saika-4	WABIAN-2
D.O.F	Total = 5 Hip=1, Knee=2, Ankle=2	Total = 12 Hip=6, Knee=2, Ankle=4	Total = 30 Head=2, Arm=14, Leg=12, Hand=2	Total = 16 Waist=2, Hip=6, Knee=2, Ankle=6
Structure Material	Aluminum	Aluminum	Aluminum for mechanical parts, Al-Cu-Mg alloy for mechanical pieces, Al-Mg alloy for outer shell.	Duralumin
Actuators	Pneumatic muscle	DC Servo motor	DC Servo motor	DC Servo motor
Microcontroller	16F84A microcontroller programmed using C language	PC/AT compatible CPU with RT-Linux OS and programmed using C language	IBM PC/AT clone with QNX Realtime Platform OS.	PC consists of a PCI CPU board

Advantages	- Uses basic microcontroller with easy to be programmed C language. - Simple design that is easy to be built.	- Has movable angle similar to human. (roll, pitch, yaw).	- Light weight, low cost and human size. - Has gravity compensation mechanism for better leg performance.	- Capable of knee stretch walking. - Higher DOF allow higher stability and flexibility when walking.
Disadvantages	- The robot is not stable where it will fall forward and backwards. - Pneumatic actuator cannot provide accurate force.	- Uses external power supply.	- Capable of squatting and sitting down only.	- Could not realize stable walking with step length more than 0.2m/step due to mechanical displacement.

2.2 Walking Analysis

The author in a research suggests the analysis of walking gait by calculating the center of mass [8]. Initially, the center of mass is obtained by approximating the density of individual parts of the robot from the 3D design. After fabrication, the actual mass of the joints are measured to find the actual center of mass and to verify the location point. Therefore, by using these data, the movement of the center of mass during walking or other motions can be determined by plotting the trajectory.

Furthermore, an author in another research proposed that the ZMP of the robot is calculated to analyse the stability of the robot [5]. Force data obtained from eight force sensors located at each foot of the KBR-1R is used to calculate the ZMP by using the following formula:

$$\text{ZMP} = \frac{P_R f_{RZ} + P_L f_{LZ}}{f_{RX} + f_{LZ}} \tag{2.1}$$

Where f_{LZ} and f_{RX} are the vertical reaction force acting on the left and right foot while P_L and P_R are the left and right position vectors of the reaction force. Besides, the joint angle response is also measured to show the difference between desired and actual value of the joint angle [5].

Finally, a researcher conducted adaptive dynamic biped walking adapting to human's living floor with unknown shape [9]. From the experiment, the value for the gradient of the landing path surface together with the distance between the upper and lower foot plate is determined to analyse the performance of the robot on human living surface. The experiment is carried out with eight steps walking including the start and end of walk.

Table 2.2: Summary of Walking Analysis

	Journal 1	**Journal 2**	**Journal 3**
Title	Design, Fabrication and Analysis of Bipedal Walking Robot	Development of a Biped Walking Robot KBR-1R	Development of a Dynamic Biped Walking System for Humanoid Development of a Biped Walking Robot Adapting to the Humans' Living Floor
Analysis Method	Calculation for the center of mass (pre-fabrication and post-fabrication)	Stable walking on flat plane with angle response of ankle, knee and hip taken. ZMP is also calculated using the value obtained from the force sensors mounted on the foot.	Dynamic walking testing on human's living floor with unknown shape. (obtaining gradient of landing path surface and distance between upper with lower foot plate).

Advantages	- Easy to calculate and obtain.	- Each performance of the ankle, knee and hip can be seen clearly with the robot's stability tested (ZMP)	- Performance of robot on different surface can be tested.
Disadvantages	- Value approximation during pre-fabrication may deviate far from actual value obtained after fabrication.	- Value obtained from sensors may be inaccurate due to errors.	- The experiment is time consuming (e.g. 50 trials).

2.3 Literature Review Conclusion

Based on the summary in Table 2.1, it can be observed that the biped robot with more than 12 D.O.F for both legs or 6 D.O.F for each leg achieve a much stable walking motion in their experiments. By having such D.O.F configuration also allows better flexibility of the biped robot. Other than that, the summary also shows that majority of the biped robot uses DC Servo motor as their main actuator since it allows adequate torque with high accuracy and high movable angle. Hence, the biped robot proposed in this report uses 6 D.O.F for each leg and DC Servo motor is selected as the main actuator. However, the microcontroller proposed to be used is Arduino which is different from any of the microcontroller used by other researcher shown in the summary. This is to implement new element into this project that is different from other research.

Furthermore, based on the summary in Table 2.2, it can be observed that different analysis method has been proposed by different research to achieve their objectives. Only one of the analysis methods in the summary is suitable for this project which is the center of mass calculation analysis. Since this project only complies on the static walking sequence on smooth surface, ZMP analysis proposed in journal 2 of the summary and walking test on surface with unknown shape analysis proposed in journal 3 of the summary would not be used. Other suitable analysis will be proposed in the methodology section.

CHAPTER 3

METHODOLOGY

The biped robot consists of 6 degrees of freedom for each leg, left and right. Each of the degree of freedom is made up of a Dynamixel AX-12A servo motor and Arduino UNO is used as the microcontroller in controlling the servo motors. Before constructing the biped robot, the structure of the biped robot is designed using the SolidWorks. Meanwhile, Proteus ISIS is used to simulate the basic operation of Arduino UNO microcontroller such as LED blinking and hobby servo motor control before implementing the Dynamixel AX-12A servo motor control. In Proteus ISIS, the Arduino board model can be added and simulation can be done using the desired circuitry before implementing the hardware testing to prevent unnecessary damage to the Arduino board. This step is taken so that the basic algorithm function of Arduino UNO can be understood. Moreover, the stability of the biped robot has been studied to ensure the biped robot can achieve stable static motion which will be further analysed with several experiment. Take note that only left leg of the biped robot is designed and constructed in this report but the whole biped robot is used in the analysis. FYP 1 will be focused on the design and construction of the biped robot while FYP 2 will be focused on the analysis of the biped robot performance. The overall timeline or Gantt chart for this project execution can be seen in Appendix B while the project flowchart can be seen in Appendix C.

3.1 Mechanical Design

The following design is the left leg of the biped robot with 6 degrees of freedom. 3 degrees of freedom are located at the hip (roll, pitch and yaw), 1 degree of freedom is located at the knee (pitch) and 2 degrees of freedom are located at the ankle (roll and pitch).

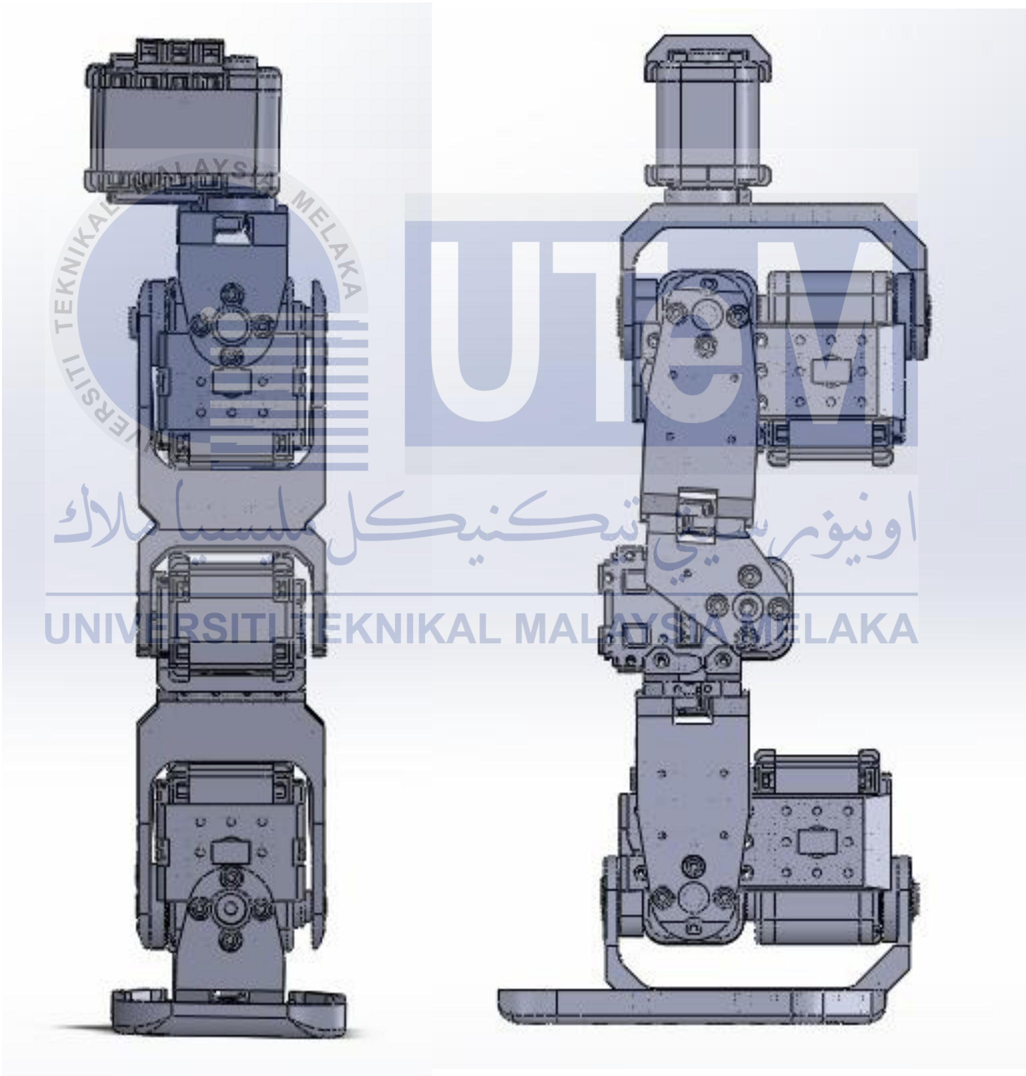


Figure 3.1: Left leg CAD drawing (front and lateral view)

3.1.1 Configuration Method

The biped robot CAD drawing is designed based on the Robotis bioloid premium type A. The microcontroller and circuitry will be mounted on the top of the hip so that the overall biped robot can be balanced. The frames will be fabricated using plastic which is lighter and cheaper. Lastly, the frames and servo motor will be connected using bolts and nuts to secure its position. Other than that, the Dynamixel AX-12A servo motor will be connected via daisy chain as shown in Figure 3.2.

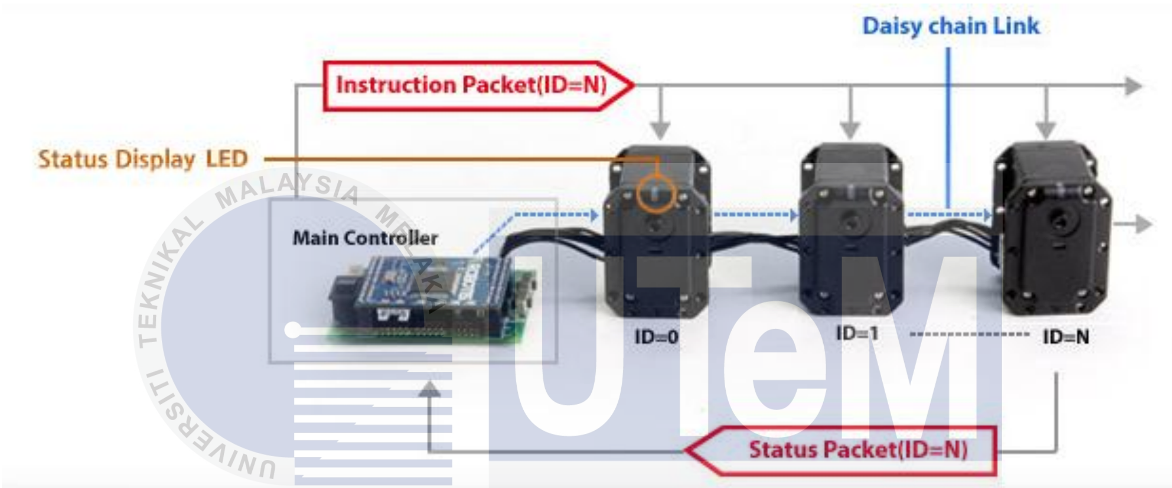


Figure 3.2: Dynamixel AX-12A daisy chain connection

This daisy-chain connection is a series connection between the servo motor since Dynamixel AX-12A servo motor uses serial communication. Hence, the wiring will be tidier since less connection to the microcontroller ports is needed unlike conventional servo motor which uses parallel connection to the microcontroller ports.

3.1.2 Cost Estimation

The cost estimation of the biped robot has been calculated [10][11]. The estimation is divided into left leg cost, right leg cost and overall biped robot cost. Table 3.1 shows the cost estimation for left and right leg of the biped robot excluding microcontroller and Table 3.2 shows the cost estimation for overall biped robot.

Table 3.1: Cost estimation for left and right leg excluding microcontroller

No	Items Descriptions	Quantity	Cost per Unit (RM)	Total Cost (RM)
1	Dynamixel AX-12A Servo Motor	6 units	144.74	868.44
2	Frames	1 set	37.70 per set	37.70
3	Screws and Nuts	1 set	4.6 per set	4.60
			Overall Total	910.74

Table 3.2: Cost estimation for overall biped robot

No	Items Descriptions	Quantity	Cost per Unit (RM)	Total Cost (RM)
1	Arduino Uno Rev3	1 unit	88.76	88.76
2	Dynamixel AX-12A Servo Motor	12 units	144.74	1736.88
3	Frames	2 set	37.70 per set	75.40
4	Screws and Nuts	2 set	4.6 per set	9.20
			Overall Total	1910.24

3.1.3 List of Materials

Table 3.3: List of Materials

No	Items Descriptions	Quantity
1	Arduino Uno Rev3	1 unit
2	Dynamixel AX-12A Servo Motor	12 units (6 per leg)
3	Frames	2 set (1 set per leg)
4	Screws and Nuts	2 set (1 set per leg)
5	Tri-state Buffer Board	1 unit
6	Power Supply Protection Board	1 unit
7	Lithium Polymer Battery Monitor (3 cells)	1 unit
8	1000 mAh Lipo Battery	1 unit
9	Arduino USB Serial Light Adapter	1 unit

3.1.3.1 Functions of Materials

1. Arduino Uno Rev3 serves as the main microcontroller in the construction to send and receive required signals to/from the actuators.
2. Dynamixel AX-12A Servo Motor is the main actuator used in the biped robot for initialising the movement of walking motion with 12 D.O.F.
3. Frames are used to secure the positions of the Dynamixel AX-12A Servo Motor while allowing certain degree of rotational motion in each D.O.F.
4. Screws and Nuts are used to attach the frames to the Dynamixel AX-12A Servo Motor and to other frames.
5. Tri-state Buffer Board serves as the medium to transfer half-duplex communication from Arduino UNO to 12 Dynamixel AX-12A Servo Motors.
6. Power Supply Protection Board serves as the protection circuit to prevent current backflow to the Lipo Battery and on the same time provide additionally current when there is a huge demand of current.
7. Lithium Polymer Battery Monitor (3 cells) is used to check the power supply of the Lipo Battery. Green LED indicates safe to use while Red LED and buzzer sound indicate the power supply is lower than safe usage level.
8. 1000 mAh Lipo Battery serves as the main power supply for the biped robot.
9. Arduino USB Serial Light Adapter provide an additional serial ports from Arduino.

3.1.4 Torque Calculation

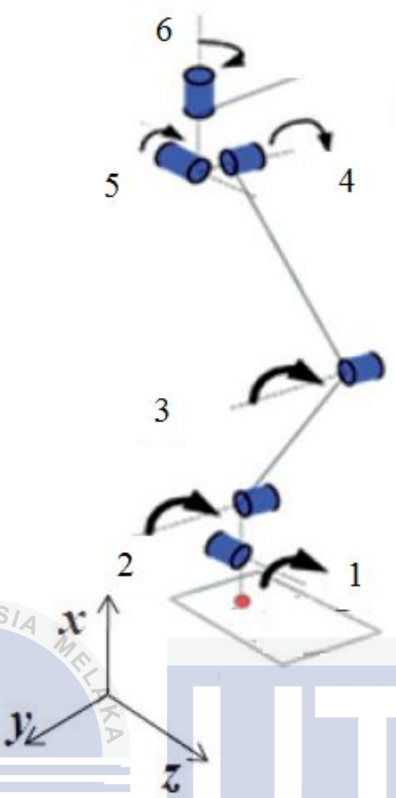


Figure 3.3: DOF configuration for single leg

Figure 3.3 shows the DOF configuration for a single leg of biped robot which have 6 degree of freedoms. To calculate the torque for each DOF, the static torque calculation is used [12]. Hence, the calculation is divided into lateral view calculation and front view calculation. The lateral view calculation consists of motor 2, 3 and 4 while fixing the orientation of motor 1 and 5. Meanwhile, the front view calculation consists of motor 1 and 5 while fixing the orientation of motor 2, 3 and 4. Motor 6 is not included in the torque calculation since it will not be used in the walking motion.

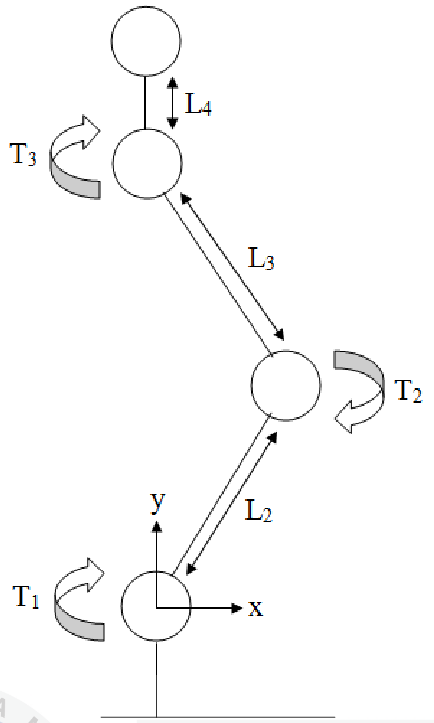


Figure 3.4: Lateral view

Table 3.4: Lateral view D-H parameters

i	**1**	**2**	**3**	**4**
θ	θ_1	θ_2	θ_3	θ_4
d	0	0	0	0
$\mathbf{a_{i-1}}$	0	L_1	L_2	L_3
$\boldsymbol{\alpha_{i-1}}$	0	0	0	0

$$ {}_{1}^{0}T = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_{2}^{1}T = \begin{bmatrix} C2 & -S2 & 0 & L_2 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}_{3}^{2}T = \begin{bmatrix} C3 & -S3 & 0 & L_3 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} C4 & -S4 & 0 & L_4 \\ S4 & C4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Starting from the last link:

$$ {}^{i}_{i}n = {}^{i}_{i+1}n + {}^{i}_{i+1}p \text{ x } {}^{i}_{i+1}f \qquad (3.1) $$

$$ {}^{3}_{3}n = {}^{3}_{4}n + {}^{3}_{4}p \text{ x } {}^{3}_{4}f $$

$$ = 0 + \begin{bmatrix} L_4 \\ 0 \\ 0 \end{bmatrix} \text{ x } \begin{bmatrix} f_x \\ f_y \\ 0 \end{bmatrix} $$

$$ = \begin{bmatrix} 0 \\ 0 \\ L_4 f_y \end{bmatrix} $$

Next link:

$$ {}^{i}_{i}f = {}^{i}_{i+1}R \text{ x } {}^{i+1}_{i+1}f \qquad (3.2) $$

$$ {}^{2}_{2}f = {}^{2}_{3}R \text{ x } {}^{3}_{3}f $$

$$ = \begin{bmatrix} C3 & -S3 & 0 \\ S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} f_x \\ f_y \\ 0 \end{bmatrix} $$

$$ = \begin{bmatrix} f_x C3 - f_y S3 \\ f_x S3 + f_y C3 \\ 0 \end{bmatrix} $$

$$ {}^{i}_{i}n = {}^{i}_{i+1}R \text{ x } {}^{i+1}_{i+1}n + {}^{i}_{i+1}p \text{ x } {}^{i}_{i}f \qquad (3.3) $$

$$ {}^{2}_{2}n = {}^{2}_{3}R \text{ x } {}^{3}_{3}n + {}^{2}_{3}p \text{ x } {}^{2}_{2}f $$

$$ = \begin{bmatrix} C3 & -S3 & 0 \\ S3 & C3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} 0 \\ 0 \\ L_4 f_y \end{bmatrix} + \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} \text{ x } \begin{bmatrix} f_x C3 - f_y S3 \\ f_x S3 + f_y C3 \\ 0 \end{bmatrix} $$

$$ = \begin{bmatrix} 0 \\ 0 \\ L_4 f_y + L_3 f_x S3 + L_3 f_y C3 \end{bmatrix} $$

Next link:

Similarly by using equation (3.2) and (3.3),

$$ {}^{1}_{1}n = \begin{bmatrix} 0 \\ 0 \\ L_4 f_y + L_3 f_x S3 + L_3 f_y C3 + L_2 f_x S2 + L_2 f_y C2 \end{bmatrix} $$

Therefore,

$$ T_1 = L_4 f_y + L_3 f_x S3 + L_3 f_y C3 + L_2 f_x S2 + L_2 f_y C2 \quad (3.4) $$

$$ T_2 = L_4 f_y + L_3 f_x S3 + L_3 f_y C3 \quad (3.5) $$

$$ T_3 = L_4 f_y \quad (3.6) $$

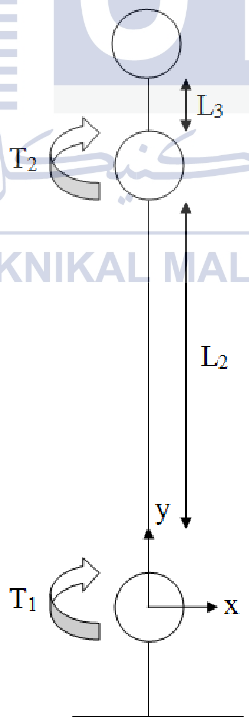


Figure 3.5: Front view

Table 3.5: Front view D-H parameters

i	**1**	**2**	**3**
θ	θ_1	θ_2	θ_3
d	0	0	0
$\mathbf{a_{i-1}}$	0	L_2	L_3
$\boldsymbol{\alpha_{i-1}}$	0	0	0

$$^{0}_{1}T = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}_{2}T = \begin{bmatrix} C2 & -S2 & 0 & L_2 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}_{3}T = \begin{bmatrix} C3 & -S3 & 0 & L_3 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Starting from the last link:

By using equation (3.1),

$$^{2}_{2}n = {}^{2}_{3}n + {}^{2}_{3}p \text{ x } {}^{2}_{3}f$$

$$= 0 + \begin{bmatrix} L_3 \\ 0 \\ 0 \end{bmatrix} \text{ x } \begin{bmatrix} f_x \\ f_y \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ L_3 f_y \end{bmatrix}$$

Next link:

By using equation (3.2),

$$^{1}_{1}f = {}^{1}_{2}R \text{ x } {}^{2}_{2}f$$

$$= \begin{bmatrix} C2 & -S2 & 0 \\ S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} f_x \\ f_y \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} f_x C2 - f_y S2 \\ f_x S2 + f_y C2 \\ 0 \end{bmatrix}$$

By using equation (3.3),

$${}_1^1n = {}_2^1R \text{ x } {}_2^2n + {}_2^1p \text{ x } {}_1^1f$$

$$= \begin{bmatrix} C2 & -S2 & 0 \\ S2 & C2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} 0 \\ 0 \\ L_3 f_y \end{bmatrix} + \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \text{ x } \begin{bmatrix} f_x C2 - f_y S2 \\ f_x S2 + f_y C2 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ L_3 f_y + L_2 f_x S2 + L_2 f_y C2 \end{bmatrix}$$

Therefore,

$$T_1 = L_3 f_y + L_2 f_x S2 + L_2 f_y C2 \quad (3.7)$$

$$T_2 = L_3 f_y \quad (3.8)$$

*Ci = Cosθ_i , Si = Sinθ_i

3.2 Electrical and Electronic Design

3.2.1 Hobby Servo Motor Control

The following is the designed hobby servo motor control electrical and electronic configuration in Proteus ISIS together with the Arduino UNO algorithm:

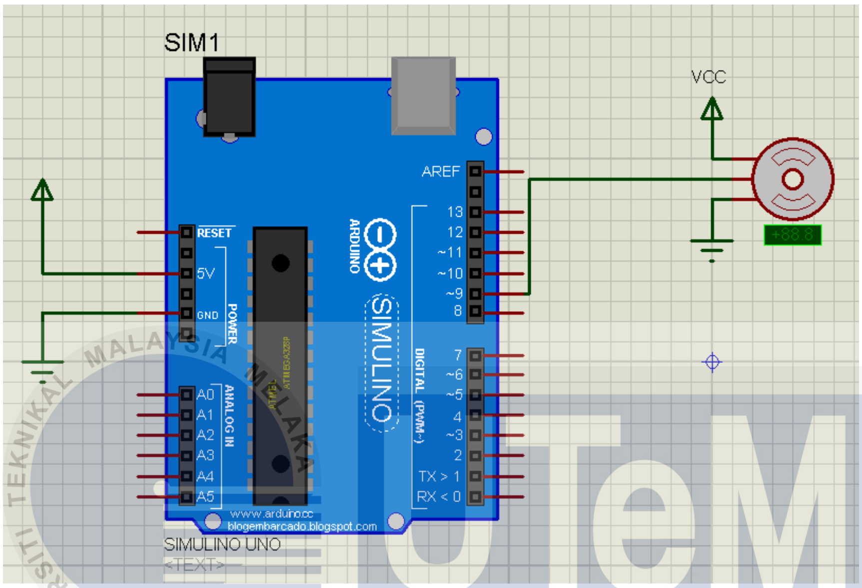


Figure 3.6: Hobby servo motor control ISIS configuration


```
#include <Servo.h>

//Define Pin
int ServoPin = 9;

//Create Servo Object
Servo HobbyServo;

void setup()
{
  //Attaches the Servo to our object
  HobbyServo.attach(ServoPin);
}
```

```
void loop()
{

    HobbyServo.write(i);
    //i = any number for wanted angle
    delay(1000);

}
```

The functionality of the ISIS configuration with the Arduino algorithm is tested in the ISIS simulation and hardware implementation. Besides, positioning experiment is also carried out by taking the angle of the servo motor to check for the error between the desired and actual value. The angle of rotation is measured using a protractor. Average values for multiple measurements are calculated to get a more accurate data and avoid systematical error. Lastly, the data is collected and tabulated as shown in Table 3.6. Analysis is done to see the accuracy of the servo motor rotation.

Table 3.6: Servo Motor Positioning

Desired Value (degree)	**Actual Value (degree)**				**Error (Desired – Average)**
	Value 1	**Value 2**	**Value 3**	**Average**	
30					
60					
90					
120					
150					
180					

3.2.2 Tri-state Buffer Board

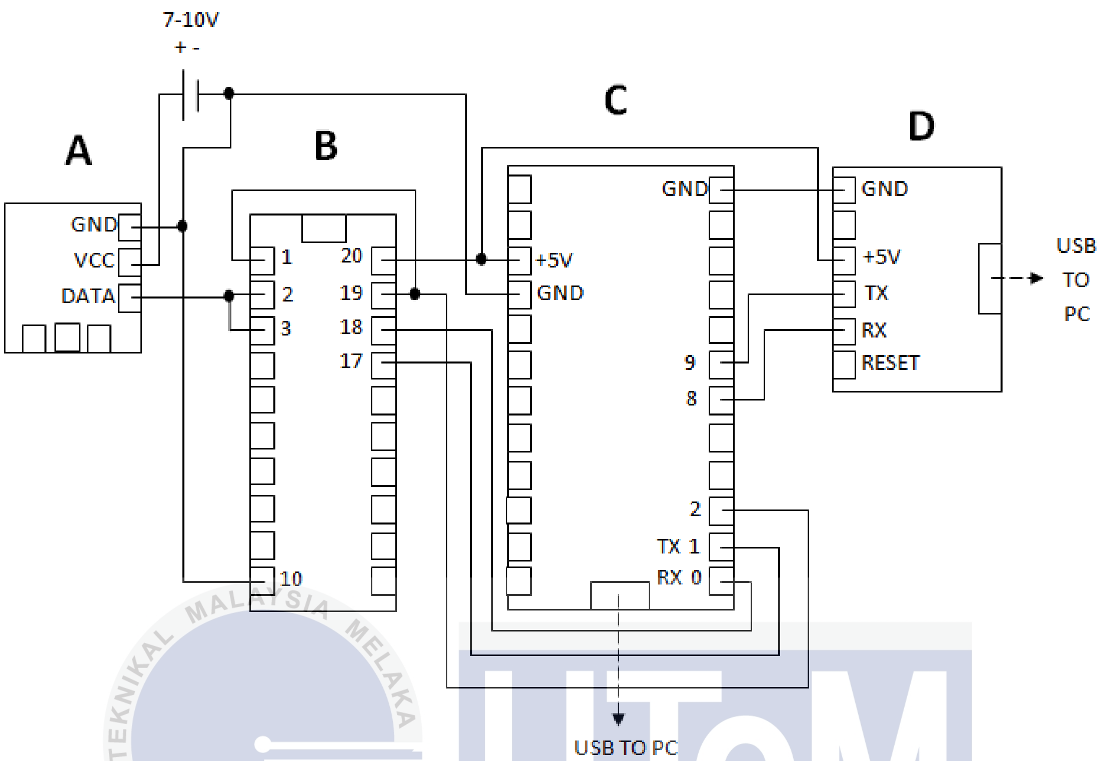


Figure 3.7: Tri-state Buffer Board electronic configuration

A. Dynamixel AX-12A Servo Motor

B. 74LS241N tri-state buffer

C. Arduino Uno Rev3

D. Arduino USB Serial Light Adapter

Figure 3.7 shows the electronic configuration used to build the tri-state buffer board hardware that is required to transfer half-duplex communication between Arduino Uno Rev3 to multiple Dynamixel AX-12A Servo Motors.

3.2.3 Power Supply Protection Board

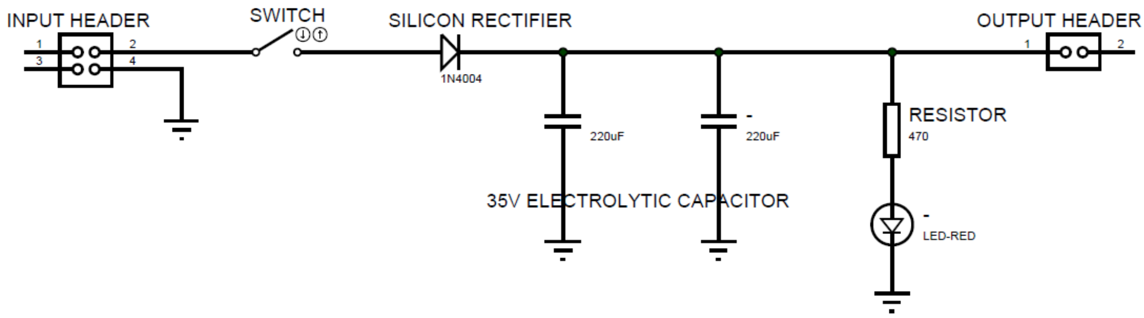


Figure 3.8: Power Supply Protection Board electronic configuration

Figure 3.8 shows the electronic configuration used to build the power supply protection board hardware. The input header will be connected to the 1000mAh Lipo battery and the output header will be connected to the Dynamixel AX-12A servo motors.

3.3 Stability

The control criteria for a static walking robot must maintain its center of mass or center of gravity on the ground directly inside of its support polygon. Therefore, by doing so, the robot will have slow walking speed and it can only move on flat surface [13].

On the contrary, with dynamic walking, the center of mass or center of gravity can fall outside of the support polygon but the zero moment point (ZMP) must be inside [13]. This configuration is hard to achieve since it involves complex derivation to make sure that the ZMP is inside of the support polygon.

Hence, only static walking is focused in this study.

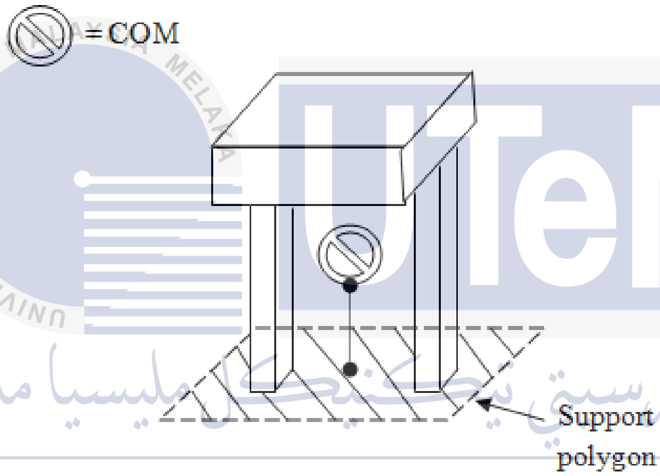


Figure 3.9: Static walking stability

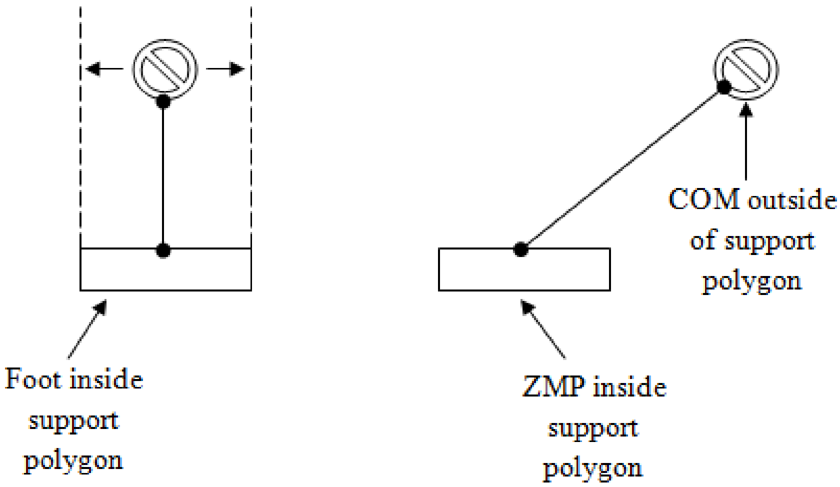


Figure 3.10: Dynamic walking stability

Static walking defines that the robot will be statically stable which means when the robot is stopped, it is practically stable. For single support phase, its support polygon where the COM lies would directly be the foot surface of that one leg while for double support phase; its support polygon where the COM lies would directly be the minimum convex area containing both foot surfaces [13]. Besides, the walking speed must be low so that the inertial force would be negligible.

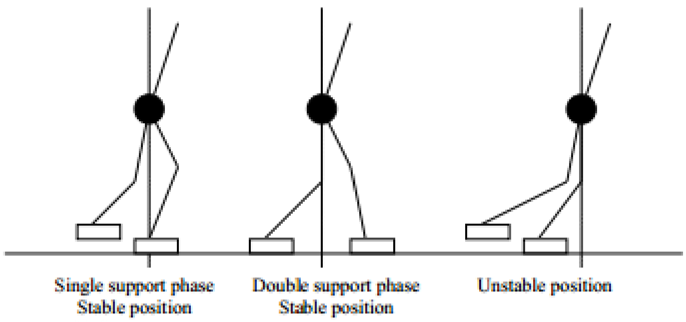


Figure 3.11: Static walking support phase [13]

3.3.1 Center of Mass (COM)

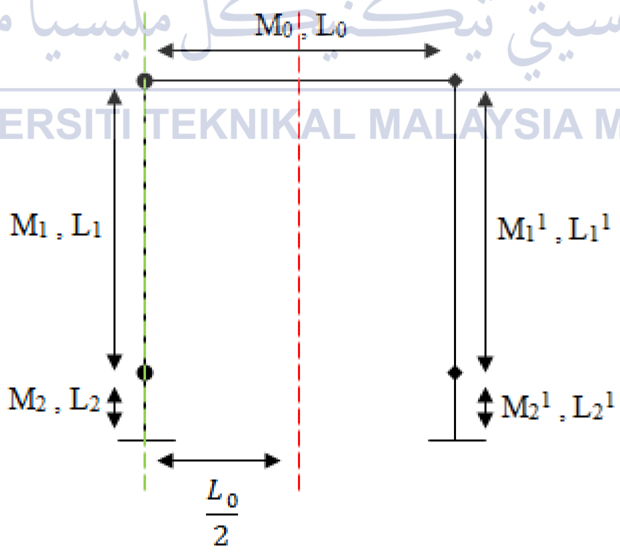


Figure 3.12: Double support phase

The green dashed line shows the reference point and red dashed line shows the middle point of the diagram.

General formula for Center of Mass (COM):

$$X_{COM} = \frac{\Sigma M_i L_i}{\Sigma M} \tag{3.9}$$

Center of mass for single support phase:

$$X_{COM1} = \frac{L_2}{2} \tag{3.10}$$

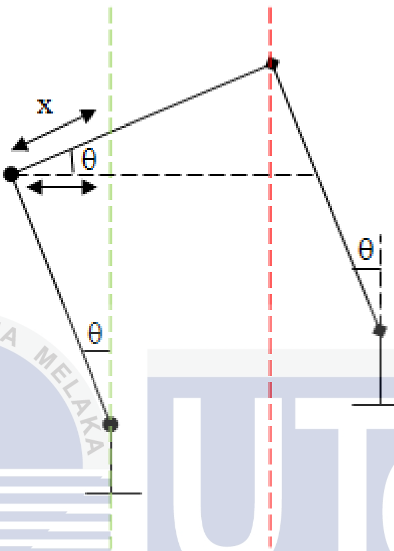


Figure 3.13: Single support phase

$$\cos\theta = \frac{L_1 \sin\theta}{x}$$

$$x = L_1 \tan\theta$$

The mass times length left of the reference point is assumed to be negative while the mass times length right of the reference point is assumed to be positive. Besides the mass distribution is assumed to be on the middle of each parts when it is aligned to a certain θ.

Center of mass for double support phase [14]:

$$X_{COM2} = \{ [M_2 \text{ x } 0] - [M_1 . \frac{L_1 \sin\theta}{2}] + [- (\frac{M_0}{L_0}. L_1 \tan\theta) (\frac{L_1 \sin\theta}{2}) + (\frac{M_0}{L_0}.(L_0 - L_1 \tan\theta)) (\frac{L_0 \cos\theta - L_1 \sin\theta}{2})] + [M_1^1 \text{ x } (- L_1 \sin\theta + L_0 \cos\theta + \frac{L_1^1 \sin\theta}{2})] + [M_2^1 \text{ x } (- L_1 \sin\theta + L_0 \cos\theta + L_1^1 \sin\theta)] \} \div (M_0 + M1 + M_2 + M_1^1 + M_2^1) \tag{3.11}$$

Using the equation (3.3), the θ needed to achieve stable single support phase can be calculated by letting $X_{COM2} = 0$, since the COM should be on the middle of the foot or the reference point. The measurements for the mass and length will be carried out after the construction of the biped robot. Besides, the calculated θ will be used in hardware implementation to verify its functionality. Analysis is done to observe the effect of COM on the stability of the biped robot during static walking. A graph will be plotted to observe the movement of the COM during static walking motion.

3.3.2 Static Walking Sequence

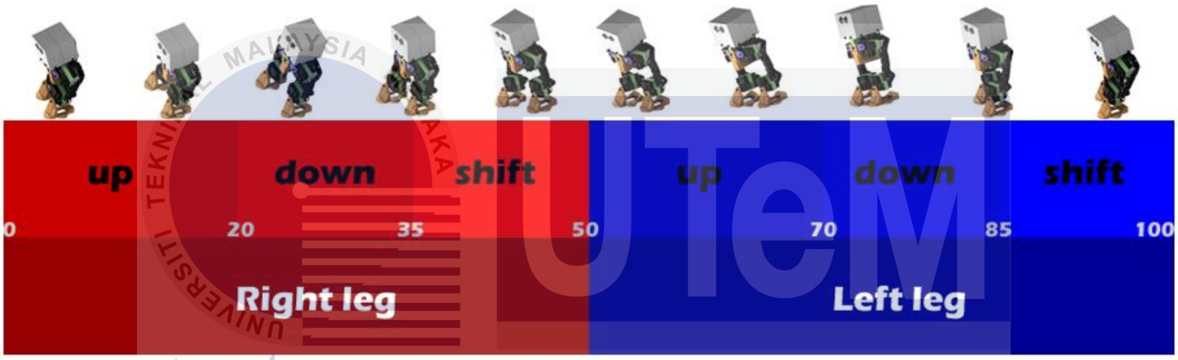


Figure 3.14: Static walking sequence [15]

Figure 3.14 shows a walking cycle of FOBO robot and is it similar to the walking sequence of static walking that is needed [15]. The walking cycle can be broken down into right leg and left leg sections. Both of the sections have the same movement but are mirrored about the central axis of the robot. Each sections are separated into up, down and shift motion. During up motion, one of the foot will be lifted and move. Then during the down motion, the lifted foot will be put down since it has reached the required displacement. Lastly, shift is done to move the weight or center of mass to the opposite leg before the foot can set off to up motion. These processes alternate continuously to produce a static walking.

3.3.2.1 Expected trajectory generation

To get the required angle for static walking motion, an experiment is carried out. Human walking motion is captured and the angle of rotation during lifting the foot and putting down the foot is measured. The hip angle, θ_H, knee angle, θ_K, and ankle angle, θ_A, is obtained. The data is tabulated for the first step and the second step since the first step start from an initial position with both legs on the same spot while the second step and the remaining steps start from the final position of the step taken before it. Figure 3.15 shows the first step taken while Figure 3.16 shows the second step taken. The shift or weight transfer angle is calculated in section 3.3.1 which is the center of mass calculation.

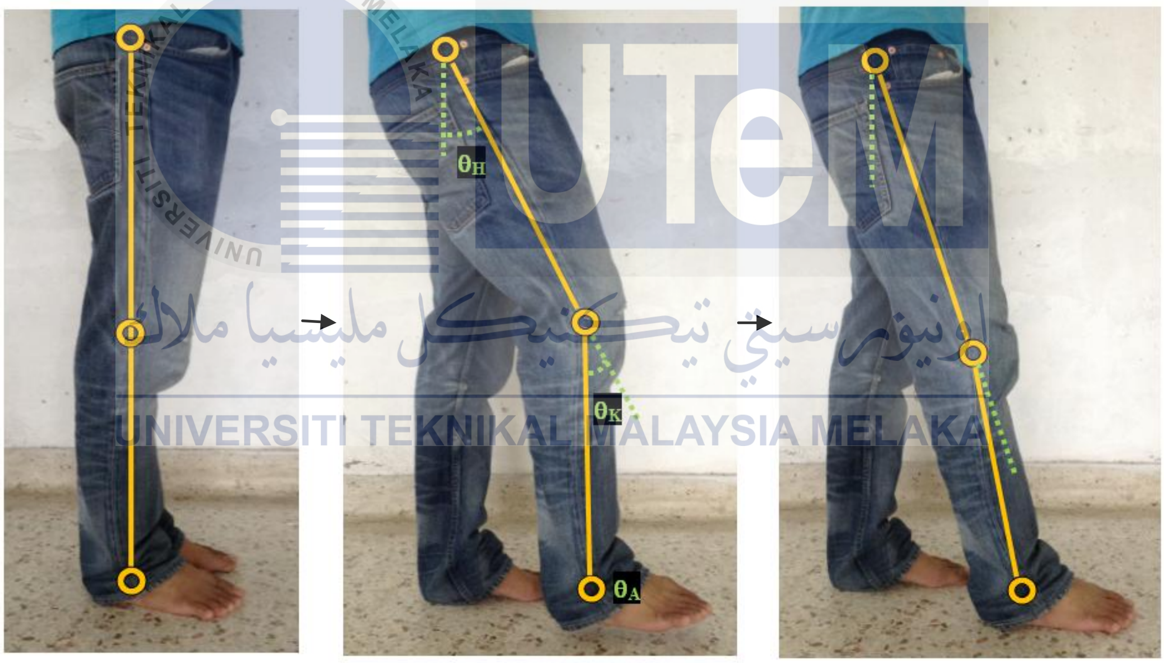


Figure 3.15: Expected Static walking first step

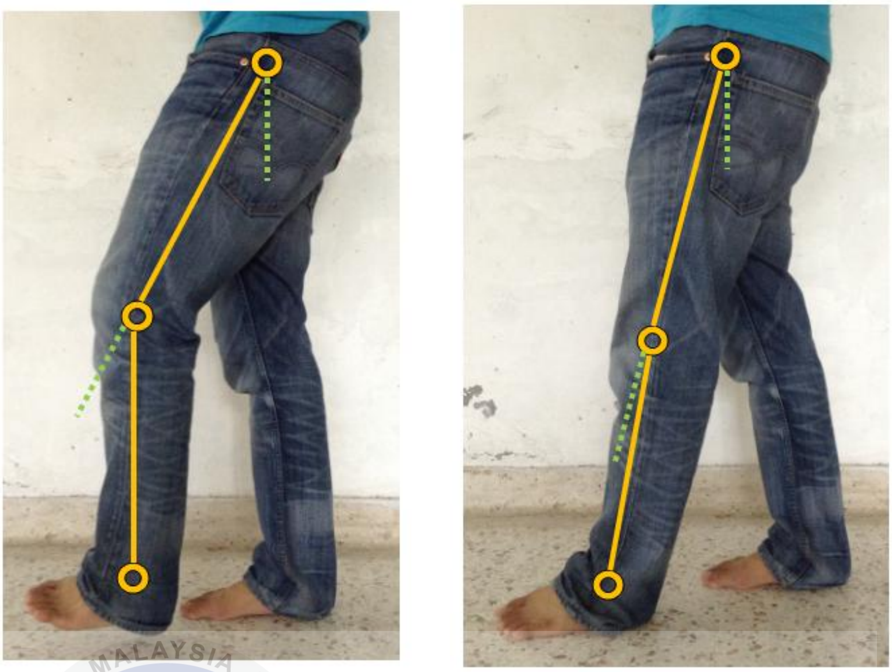


Figure 3.16: Expected Static walking second step

Anticlockwise rotation of the motor is considered to be negative angle while clockwise rotation is to be considered positive angle. Both initial position for first step and second step is assumed to have 0 initial angles for hip, knee and ankle. The data for expected result of human motion is recorded and tabulated as shown in Table 3.10. Finally, the angle of rotation is used to obtain the trajectory generation of the biped robot. The following are the derivation of trajectory generation equation [16].

The foot lifting general equation:

$$\theta_{up}(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3 \qquad (3.12)$$

The putting down foot general equation:

$$\theta_{down}(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3 \qquad (3.13)$$

The constraints to be enforced:

$$\theta_0 = a_{10}$$

$$\theta_v = a_{10} + a_{11}t_{f1} + a_{12}t_{f1}^2 + a_{13}t_{f1}^3$$

$$\theta_v = a_{20}$$

$$\theta_g = a_{20} + a_{21}t_{f2} + a_{22}t_{f2}^2 + a_{23}t_{f2}^3$$

$$0 = a_{11}$$

$$0 = a_{21} + 2a_{22}t_{f2} + 3a_{23}t_{f2}^2$$

$$a_{11} + 2a_{12}t_{f1} + 3a_{13}t_{f1}^2 = a_{21}$$

$$2a_{12} + 6a_{13}t_{f1} = 2a_{22}$$

Solving for the case $t_f = t_{f1} = t_{f2}$, we obtain

$$a_{10} = \theta_0$$

$$a_{11} = 0$$

$$a_{12} = \frac{12\theta_v - 3\theta_g - 9\theta_0}{4t_f^2}$$

$$a_{13} = \frac{-8\theta_v + 3\theta_g + 5\theta_0}{4t_f^3}$$

$$a_{20} = \theta_v$$

$$a_{21} = \frac{3\theta_g - 3\theta_0}{4t_f}$$

$$a_{22} = \frac{-12\theta_v + 6\theta_g + 6\theta_0}{4t_f^2}$$

$$a_{23} = \frac{8\theta_v - 5\theta_g - 3\theta_0}{4t_f^3}$$

3.3.2.2 Walking combination

The static walking sequence will be tested by using the constructed biped robot. Analysis is done by observing the stability of the biped robot during the walking. Besides, this static walking sequence has a strong connection with the COM discussed in section 3.3.1. So, both of the criteria must be balanced to obtain stable walking. Experiment is carried out to determine the best length of stride, height of stride and speed of stride. For each variables, 3 parameters are being tested which include low, medium and high. Initially, the length of stride and speed of stride are fixed with low parameter while height of strides is changed from low to high. From the results obtained, the most suitable height is being chosen. Next, using this height, the speed is still fixed to low while the length of stride is changed from low to high. Again, the best length is being chosen. Finally, using the height and length of stride that have been chosen, the speed is changed from low to high. As a result, the best length, height and speed of stride are selected. Only 5 steps will be performed for each walking combinations. Besides, for each walking combinations, the total length travelled, height of each step, total time taken to complete 5 steps and deviation from straight line will be measured and recorded as shown in Table 3.7. Then these data is used to calculate the length per step and also speed per step as an analysis. The experiment is carried out in a room on a flat surface or floor to avoid random error such as wind, uneven surface and high temperature which may affect the performance of the biped robot.

Table 3.7: Walking Combination experiment

Variables	**Measurements**					
	1	**2**	**3**	**4**	**5**	**Average**
Length (cm)						
Height (cm)						
Time (s)						
Deviation (cm)						

$$\text{Length per step (cm)} = \frac{Average\ length}{5} \quad (3.14)$$

$$\text{Time per step (s)} = \frac{Average\ time}{5} \quad (3.15)$$

$$\text{Speed per step (cm/s)} = \frac{Length\ per\ step}{Time\ per\ step} \quad (3.16)$$

The measurements methods are as shown in Figure 3.17 – 3.20:

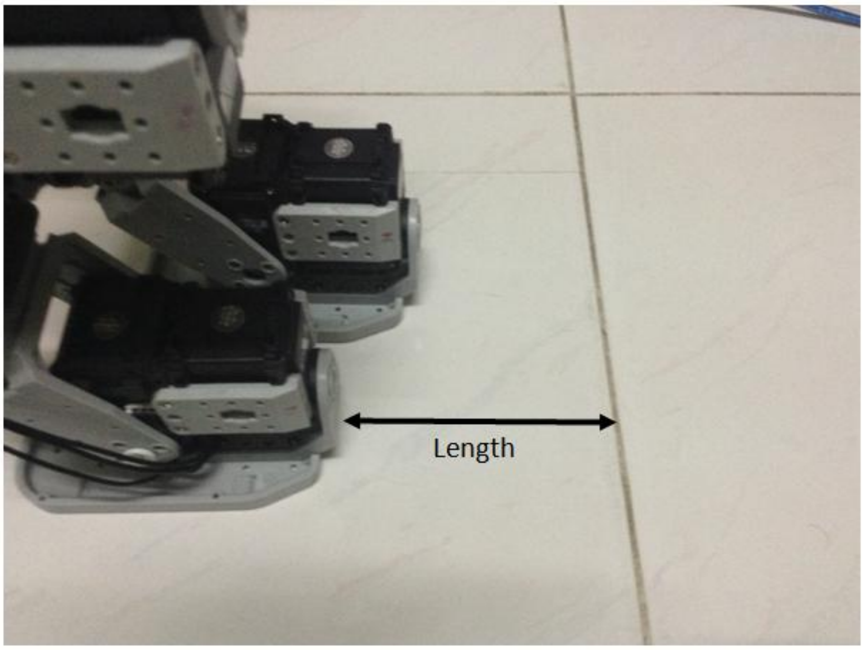


Figure 3.17: Length measurement

In the beginning, the biped robot is positioned to be aligned with the reference line on the back. Then the length is measured at the end of step 5 or the final step where it is the total length traveled. It is measured between the reference line and the final step taken or the furthest length.

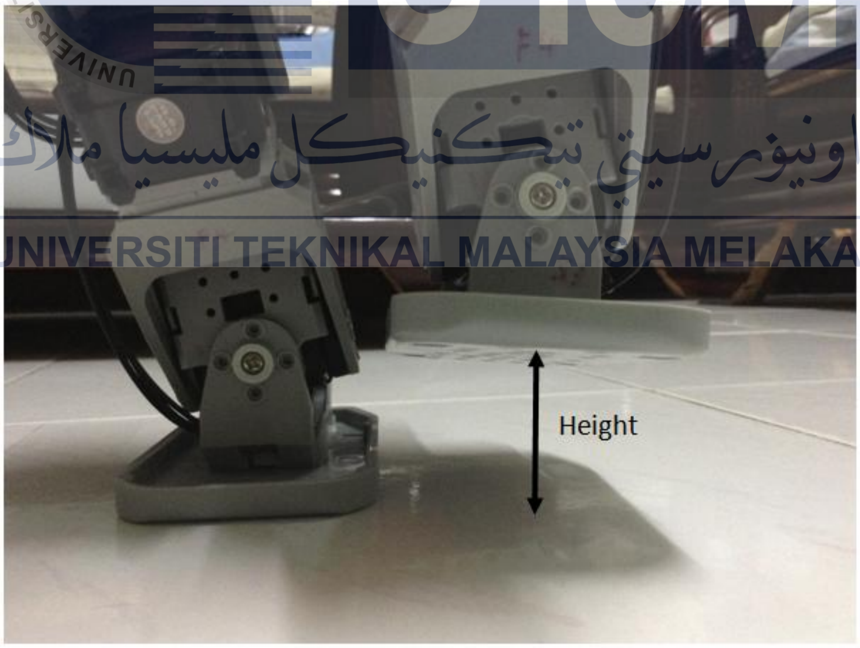


Figure 3.18: Height measurement

The height is measured for each step of a total 5 steps to get a more accurate average height value. The height is measured between the floors to the highest position of the foot lifted.

Figure 3.19: Deviation measurement left

Figure 3.20: Deviation measurement right

In the beginning of the walking motion, the right leg is positioned to be aligned with the reference line drawn. Then deviation is measured at the end of 5th step. It is measured between the reference line drawn and the furthest foot position of the right leg. The value recorded will always be positive no matter the deviation is to the left or right since the main priority is to find the value of deviation.

Nevertheless, the time is measured when the biped robot begins to move until the robot stops at the end of step 5. The time is measured by using a stopwatch while the length, height and deviation are measured by using a metre rule.

3.3.2.3 Error analysis

The next task is carried out to analyse the error of the biped robot. To analyse the error, the angle generation of each motor based on the best walking combination obtained from section 3.3.2.2 is recorded which include motor 1 to motor 12. The angle generation of each motor is further separated into 5 steps motion where each steps consist of θ_0, the initial angle when the foot is at double support phase, θ_v, the maximum angle during the lifting motion or also known as via point and θ_g, the final angle after putting down the leg or also known as goal point. Besides, the data collection is divided into desired value where it is taken from the coding and actual value where it is taken from the Dynamixel AX-12A feedback via Arduino USB serial light adapter. The data is recorded and tabulated as shown in Table 3.8 and Figure 3.21 shows the motor positioning of the biped robot.

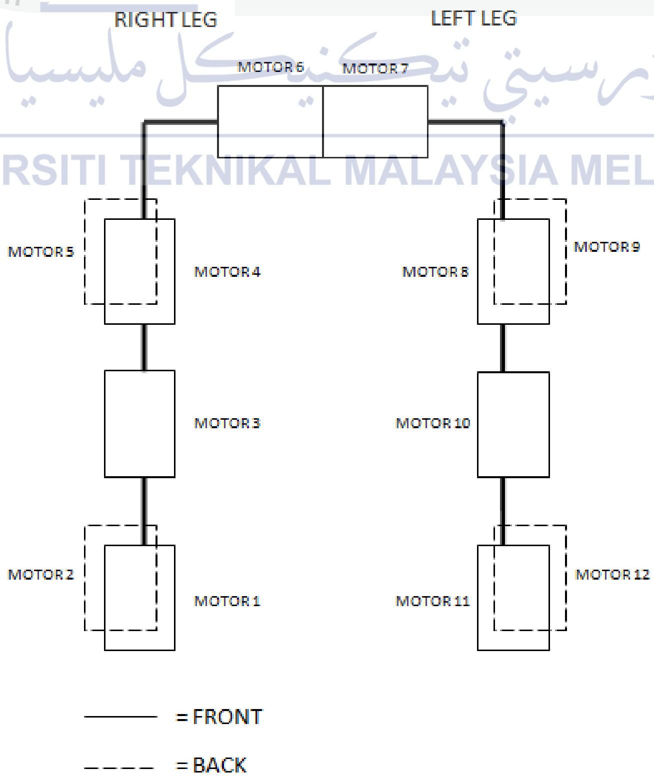


Figure 3.21: Motor Positioning of biped robot

Table 3.8: Angle Generation measurements

Motor	Measurements														
	Step 1			Step 2			Step 3			Step 4			Step 5		
	θ_{01}	θ_{v1}	θ_{g1}	θ_{02}	θ_{v2}	θ_{g2}	θ_{03}	θ_{v3}	θ_{g3}	θ_{04}	θ_{v4}	θ_{g4}	θ_{05}	θ_{v5}	θ_{g5}
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															

Since the value used in the coding ranged from 0 – 1024 steps, it is converted into angles by using simple calculation. Based on the datasheets, the total angle that can be rotated by the Dynamixel AX-12A Servo Motor is 300 degrees as shown in Figure 3.22. By dividing 300 degrees with 1024 steps, the angle per step is obtained with a value of 0.293 degree per step. Thus, the data is recorded in angle rather than steps.

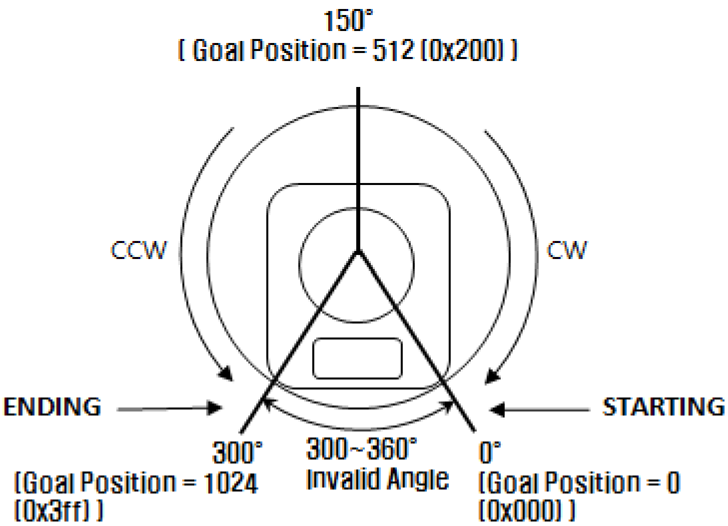


Figure 3.22: Rotation Limit of Dynamixel AX-12A Servo Motor

Moreover, the desired and actual value obtained is used to calculate the error of the angle generation by using simple equation where error = desired value – actual value. The error of the angle generation is calculated only for the hip motor, ankle motor and knee motor for both left and right leg. This is because human mainly uses only hip, knee and ankle in completing a walking motion. Even though other parts of the human legs are used for the walking motion, these usage are significantly lesser compared to hip, knee and also ankle. Hip motor will be motor 4 for right leg and motor 8 for left leg. Knee motor will be motor 3 for right leg and motor 10 for left leg. Finally, the ankle motor will be motor 1 for right leg and motor 11 for left leg. The error of the angle generation is calculated and tabulated as shown in Table 3.9.

Table 3.9: Error for Angle Generation calculation

Motor	**Calculations**														
	Step 1			**Step 2**			**Step 3**			**Step 4**			**Step 5**		
	θ_0	θ_v	θ_g	θ_0	θ_v	θ_g	θ_0	θ_v	θ_g	θ_0	θ_v	θ_g	θ_0	θ_v	θ_g
H_R															
K_R															
A_R															
H_L															
K_L															
A_L															

* H = hip, K = knee, A = ankle, subset R = right leg, subset L = left leg

The tabulated data for error of angle generation calculation is then used to plot error graph with steps as the x-axis and error as the y-axis. The graph is analysed in the analysis and discussion section. Other than that, Table 3.9 is also used to calculate the percentage of accuracy of the motors by using the equation 3.17 for accuracy analysis.

$$\text{Percentage of Accuracy} = 100 - \left(\frac{Error}{Desired\ Value} \times 100\right) \qquad (3.17)$$

3.3.2.4 Actual trajectory generation

Lastly, task to find the biped robot actual trajectory generation is carried out. The angle generation data obtained in section 3.3.2.3 is used to tabulate the biped robot trajectory angle as shown in Table 3.10. The trajectory angle is calculated only for hip angle, θ_H, knee angle, θ_K, and ankle angle, θ_A for first step and second step. This is because as explained before, human mainly uses only hip, knee and ankle for walking motion. Meanwhile, first step start from an initial position with both legs on the same spot while the second step and the remaining steps start from the final position of the step taken before it. Hence, third, fourth and fifth step trajectory angle would be the same as second step. The trajectory angle of the biped robot is measured from the vertical straight position as shown in Figure 3.15. However, for the biped robot construction, the initial position of the Dynamixel AX-12A servo motor will be at 512 step or 150 degree when the biped is at vertical straight position as shown in Figure 3.23. Therefore, the final angle obtained in Table 4.12 must be deducted by 150 degree. Anticlockwise rotation of the motor is considered to be negative angle while clockwise rotation is to be considered positive angle.

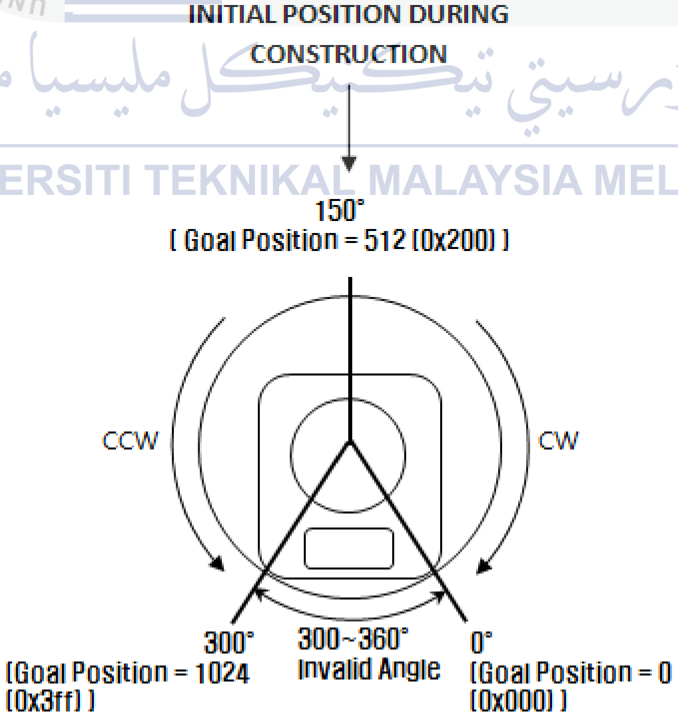


Figure 3.23: Initial Position of Dynamixel AX-12A Servo Motor at vertical straight position

Table 3.10: Trajectory angle

Angle	θ_0	θ_v	θ_g
θ_H			
θ_K			
θ_A			

Actual trajectory generation graph is plotted using the obtained data and analysis is done. Last but not least, the expected trajectory generation graph obtained in section 5.2.1 is compared with actual trajectory generation graph obtained in section 5.2.4. Analysis is done in the analysis and discussion section.

3.4 Controller

The Dynamixel AX-12A Servo Motor has a built-in ATmega chip and potentiometer that allows the servo motor to have an internal PID control system. Therefore, external controller is not being implemented in this project.

CHAPTER 4

RESULTS

4.1 Completed Hardware

Figure 4.1 – Figure 4.8 shows the completed hardware configuration.

Figure 4.1: Completed hardware front view

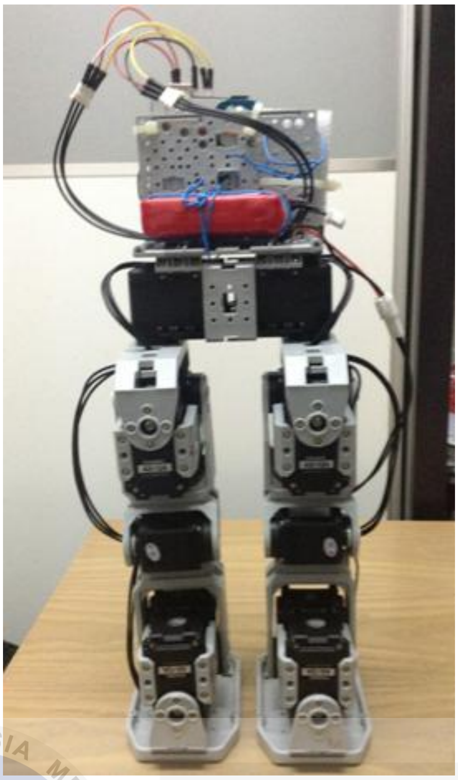


Figure 4.2: Completed hardware back view

Figure 4.3: Completed hardware left side view

Figure 4.4: Completed hardware right side view

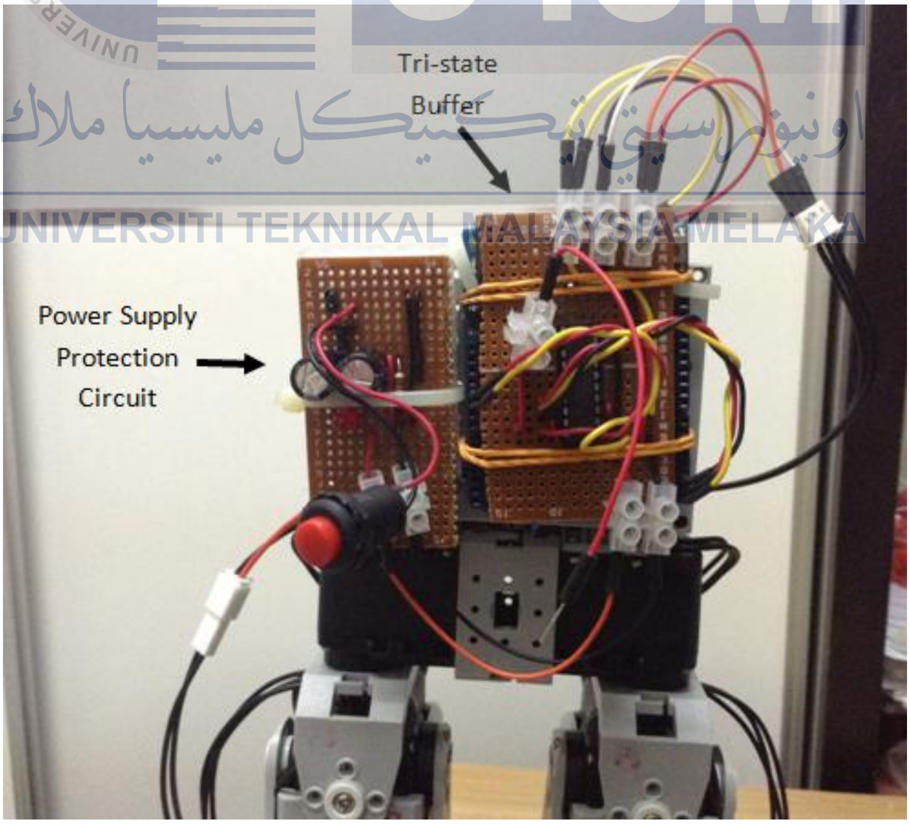


Figure 4.5: Tri-state buffer and power supply protection circuit

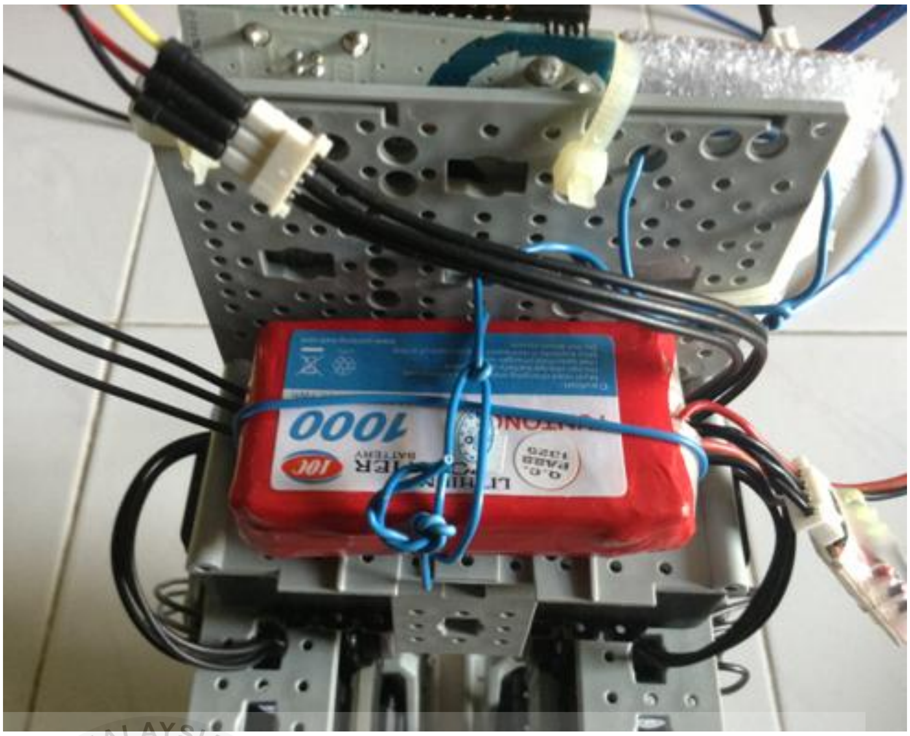


Figure 4.6: 1000mAh Lipo battery

Figure 4.7: Arduino USB Serial Light Adapter

Figure 4.8: Lithium Polymer Battery Monitor (3 cells)

4.2 Electrical and Electronic Design

4.2.1 Hobby Servo Motor Control

This experiment is carried out to test the performance of hobby servo motor as compared to other motors before selecting the suitable motor and also to familiarize with the Arduino algorithm. Therefore, only one servo motor has been used to prevent wastage since there is a limited budget.

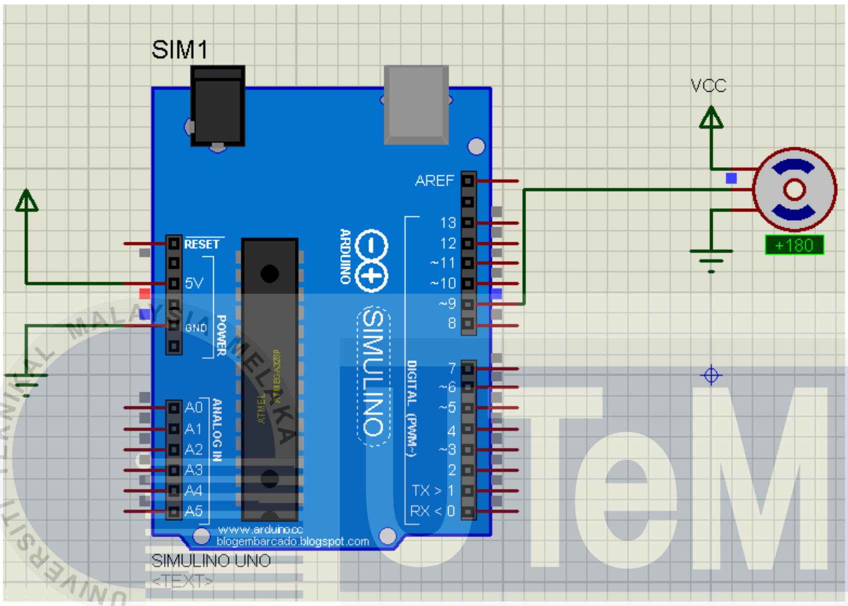


Figure 4.9: ISIS simulation (180 degree rotation)

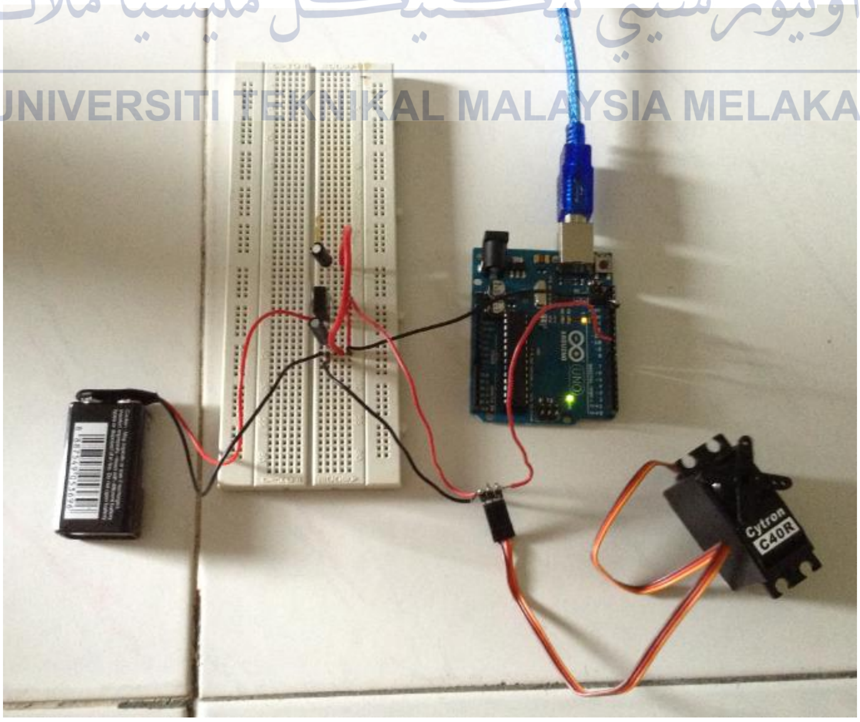


Figure 4.10: Initial hardware implementation for Servo control

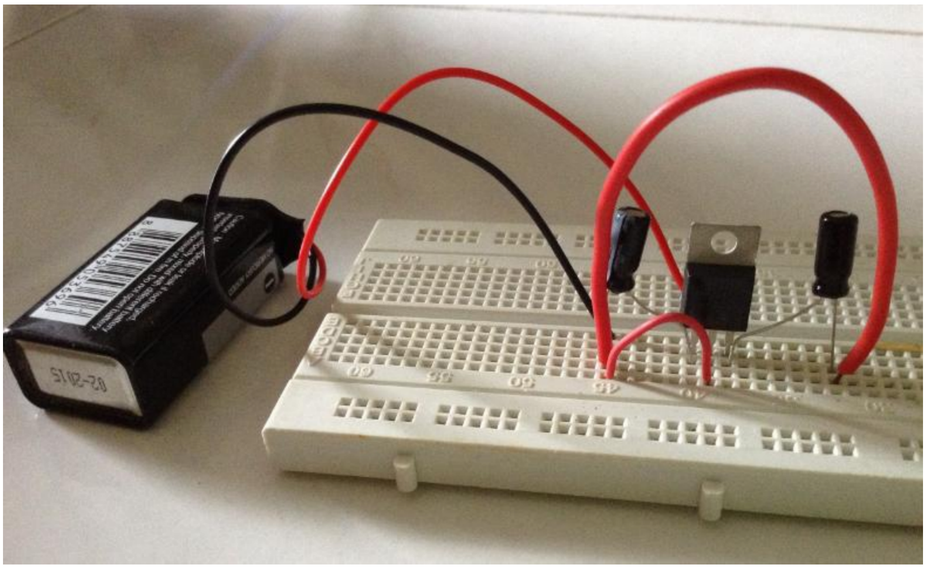


Figure 4.11: 5V voltage regulator with decoupling capacitor

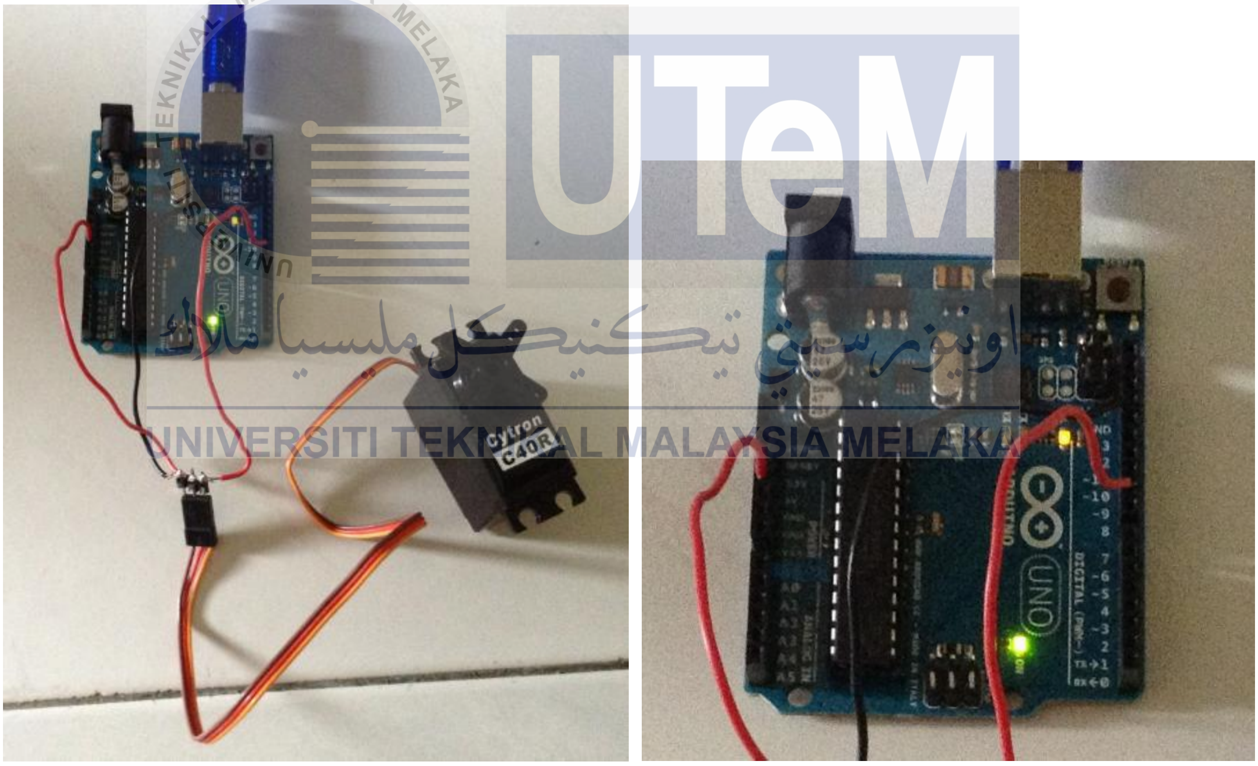


Figure 4.12: Final hardware implementation configuration

Initially, the hardware implementation is set to be like Figure 4.10 with the voltage regulator and decoupling capacitor as shown in Figure 4.11. This is to provide and external 5V DC to the servo motor with noise filtered using the decoupling capacitor to avoid taking the voltage supply from the Arduino board. However, during the experiment, the power supply seems to be inadequate causing the servo motor to vibrate and not moving. Therefore, the configuration is changed to Figure 4.12, where the power supply is taken from the Arduino board (5V pin). Both the simulation and hardware implementation works according to the applied Arduino algorithm but there is still some problem. In the ISIS simulation, the rotation tends to jitter around the desired value and it does not stop at a stable value. Meanwhile for the hardware implementation, the rotation will keep rotate to an angle higher than the desired value and then to an angle lower than the desired value. This pattern repeats for a few times before the servo motor reach a stable position. Besides, after reaching the stable position, the servo motor will still vibrates. It is deduced that this problem is may be caused by unstable PWM pulse due to the interrupt command in the library, noisy sensors in servo motor causing inaccurate encoding and insufficient power supplied to the servo motor. After troubleshooting, the cause of the problem has been found which is due to insufficient power supply from the pc USB port. The problem can be solved by using a power adapter or using another computer (due to my computer having power malfunction in the USB port).

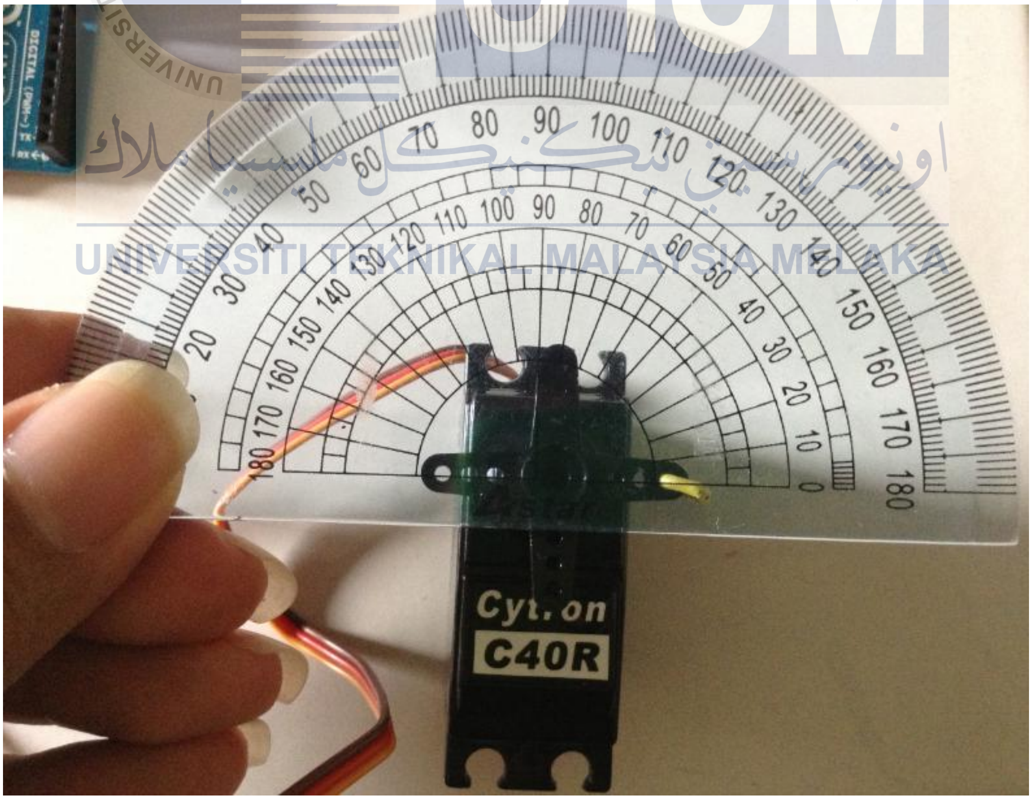


Figure 4.13: Servo angle measurement method

The measurements of the rotation angle are taken based on Figure 4.13. A wire is attached to one side of the plastic rotor as a reference and a protractor is used to measure the angle. The angle is taken few times to eliminate systematic error as suggested in section 3.2.1. However, to get more accurate results, alternative method can be used such as using an encoder. This method will be taken into consideration for use in the future. The data is collected and tabulated as shown in Table 4.1.

Table 4.1: Servo Motor Positioning results

Desired Value (degree)	**Actual Value (degree)**				**Error (Desired – Average)**
	Value 1	**Value 2**	**Value 3**	**Average**	
30	32	35	35	34.00	-4.00
60	57	58	59	58.00	2.00
90	86	88	88	87.33	2.67
120	114	115	113	114.00	6.00
150	135	138	139	137.33	12.67
180	165	167	168	166.67	13.33

4.3 Static Walking Sequence

4.3.1 Expected Trajectory Generation

4.3.1.1 Expected First Step Trajectory

Table 4.2: Expected First step angle distribution

Angle	θ_0	θ_v	θ_g
θ_{H1}	0	-27^o	-19^o
θ_{K1}	0	29^o	9^o
θ_{A1}	0	0	0

*let t_f = 1.1s

The general equation is obtained by inserting the angle distribution data into the formulas in section 3.3.2.1. θ_0 is the initial angle when the foot is at double support phase, θ_v is the maximum angle during the lifting motion or also known as via point and θ_g is the final angle after putting down the leg or also known as goal point. Then the general equation is used to plot the trajectory using matlab. The matlab algorithm can be seen in Appendix D.

General equation for hip, θ_H:

$$\theta_{upH1}(t) = 29.86t^3 - 55.17t^2$$

$$\theta_{downH1}(t) = - 22.73t^3 + 43.39t^2 - 12.95t - 27$$

General equation for knee, θ_K:

$$\theta_{upK1}(t) = - 38.5t^3 + 66.32t^2$$

$$\theta_{downK1}(t) = 35.12t^3 - 60.74t^2 + 6.14t + 29$$

4.3.1.2 Expected Second Step Trajectory

Table 4.3: Expected Second step angle distribution

Angle	θ_0	θ_v	θ_g
θ_{H2}	0	28°	16°
θ_{K2}	0	-30°	-7°
θ_{A2}	0	0	0

*let t_f = 1.1s

General equation for hip, θ_H:

$\theta_{upH2}(t) = -33.06t^3 + 59.5t^2$

$\theta_{downH2}(t) = 27.05t^3 - 49.59t^2 + 10.91t + 28$

General equation for knee, θ_K:

$\theta_{upK2}(t) = 41.13t^3 - 70.04t^2$

$\theta_{downK2}(t) = -38.5t^3 + 65.7t^2 - 4.77t - 30$

4.3.2 Walking combination

4.3.2.1 Low Length, Low Height, Low Speed

Table 4.4: Walking Combination 1

Variables	Measurements					
	1	**2**	**3**	**4**	**5**	**Average**
Length (cm)	11.2	11.1	11.1	11.2	11.2	11.16
Height (cm)	0.8	1.2	1.5	1.8	1.7	1.4
Time (s)	28.0	27.9	28.0	27.8	27.8	27.9
Deviation (cm)	0.6	0.8	0.8	0.8	0.7	0.74

Length per step (cm) = 2.232

Speed per step (cm/s) = 0.4

4.3.2.2 Low Length, Medium Height, Low Speed

Table 4.5: Walking Combination 2

Variables	Measurements					
	1	**2**	**3**	**4**	**5**	**Average**
Length (cm)	13.7	14.4	14.3	13.7	13.7	13.96
Height (cm)	1.4	1.8	2.2	2.3	2.3	2.0
Time (s)	28.0	27.8	27.8	27.9	27.8	27.86
Deviation (cm)	0.3	0.4	0.3	0.5	0.5	0.4

*selected as best height

Length per step (cm) = 2.792

Speed per step (cm/s) = 0.501

4.3.2.3 Low Length, High Height, Low Speed

Table 4.6: Walking Combination 3

Variables	Measurements					
	1	2	3	4	5	Average
Length (cm)	10.3	10.4	10.5	10.5	10.3	10.4
Height (cm)	2.9	3.7	3.8	3.9	3.9	3.64
Time (s)	32.9	32.8	32.9	32.8	32.8	32.84
Deviation (cm)	0.9	0.7	0.8	0.6	0.7	0.74

Length per step (cm) = 2.08

Speed per step (cm/s) = 0.317

4.3.2.4 Medium Length, Medium Height, Low Speed

Table 4.7: Walking Combination 4

Variables	Measurements					
	1	2	3	4	5	Average
Length (cm)	20.5	20.6	20.6	20.5	20.6	20.56
Height (cm)	1.3	2.0	2.3	2.5	2.4	2.1
Time (s)	28.0	28.1	28.0	27.9	28.0	28.0
Deviation (cm)	0.7	1.3	1.5	1.1	1.3	1.18

*selected as best length

Length per step (cm) = 4.112

Speed per step (cm/s) = 0.734

4.3.2.5 High Length, Medium Height, Low Speed

Table 4.8: Walking Combination 5

Variables	Measurements					
	1	2	3	4	5	Average
Length (cm)	23.5	23.4	23.5	23.3	23.4	23.42
Height (cm)	1.4	2.4	2.6	2.8	2.6	2.36
Time (s)	28.0	27.9	27.9	27.9	27.9	27.92
Deviation (cm)	0.3	0.1	0.3	0.6	0.4	0.34

Length per step (cm) = 4.684

Speed per step (cm/s) = 0.839

4.3.2.6 Medium Length, Medium Height, Medium Speed

Table 4.9: Walking Combination 6

Variables	Measurements					
	1	2	3	4	5	Average
Length (cm)	18.7	19.0	19.1	18.9	19.0	18.94
Height (cm)	1.2	2.0	2.2	2.5	2.3	2.04
Time (s)	11.3	10.7	10.9	10.9	10.9	10.94
Deviation (cm)	0.3	0.2	0.2	0.5	0.5	0.34

*selected as best speed

Length per step (cm) = 3.788

Speed per step (cm/s) = 1.731

4.3.2.7 Medium Length, Medium Height, High Speed

Table 4.10: Walking Combination 7

Variables	Measurements					
	1	2	3	4	5	Average
Length (cm)	21.4	20.9	20.8	20.9	21.0	21.0
Height (cm)	1.3	2.0	2.2	2.6	2.3	2.08
Time (s)	7.0	7.1	7.1	7.1	7.1	7.08
Deviation (cm)	0.2	0.1	0.3	0.1	0.1	0.16

Length per step (cm) = 4.2

Speed per step (cm/s) = 2.966

4.3.3 Error analysis

4.3.3.1 Desired Angle Generation

Table 4.11: Desired angle generation measurements

Motor	Desired Measurements (degree)														
	Step 1			Step 2			Step 3			Step 4			Step 5		
	θ_{01}	θ_{v1}	θ_{g1}	θ_{02}	θ_{v2}	θ_{g2}	θ_{03}	θ_{v3}	θ_{g3}	θ_{04}	θ_{v4}	θ_{g4}	θ_{05}	θ_{v5}	θ_{g5}
1 (A_R)	180.2	180.2	188.1	188.1	188.1	180.2	180.2	180.2	193.1	193.1	193.1	180.2	180.2	180.2	193.1
2	150.0	164.9	150.0	150.0	135.1	150.0	150.0	164.9	150.0	150.0	135.1	150.0	150.0	164.9	150.0
3 (K_R)	89.95	89.95	89.95	89.95	70.0	89.95	89.95	89.95	89.95	89.95	70.0	89.95	89.95	89.95	89.95
4 (H_R)	120.1	120.1	128.0	128.0	100.2	120.1	120.1	120.1	133.0	133.0	100.2	120.1	120.1	120.1	133.0
5	150.0	150.0	150.0	150.0	140.0	150.0	150.0	150.0	150.0	150.0	140.0	150.0	150.0	150.0	150.0
6	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9	104.9
7	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8	194.8
8 (H_L)	180.2	200.1	180.2	180.2	180.2	167.3	167.3	200.1	180.2	180.2	180.2	167.3	167.3	200.1	180.2
9	150.0	159.9	150.0	150.0	150.0	150.0	150.0	159.9	150.0	150.0	150.0	150.0	150.0	159.9	150.0
10 (K_L)	209.8	229.7	209.8	209.8	209.8	209.8	209.8	229.7	209.8	209.8	209.8	209.8	209.8	229.7	209.8
11 (A_L)	120.1	120.1	120.1	120.1	120.1	107.2	107.2	107.2	120.1	120.1	120.1	107.2	107.2	107.2	120.1
12	150.0	164.9	150.0	150.0	135.1	150.0	150.0	164.9	150.0	150.0	135.1	150.0	150.0	164.9	150.0

4.3.3.2 Actual Angle Generation

Table 4.12: Actual angle generation measurements

Motor	Actual Measurements (degree)														
	Step 1			Step 2			Step 3			Step 4			Step 5		
	θ_{01}	θ_{v1}	θ_{g1}	θ_{02}	θ_{v2}	θ_{g2}	θ_{03}	θ_{v3}	θ_{g3}	θ_{04}	θ_{v4}	θ_{g4}	θ_{05}	θ_{v5}	θ_{g5}
1 (A_R)	179.6	179.6	188.1	188.1	188.1	180.2	180.2	180.2	193.1	193.1	193.1	180.2	180.2	180.2	193.1
2	150.0	164.4	150.0	150.0	135.1	149.7	149.7	164.4	149.7	149.7	134.8	149.7	149.7	164.4	149.7
3 (K_R)	90.5	90.5	90.5	90.5	70.32	90.5	90.5	90.8	90.8	90.8	70.32	90.5	90.5	90.8	90.8
4 (H_R)	118.7	118.7	126.9	126.9	99.3	119.5	119.5	119.5	131.3	131.3	99.3	118.9	118.9	118.9	131.3
5	149.7	149.7	149.7	149.7	140.3	149.7	149.7	149.7	149.7	149.7	140.3	149.7	149.7	150.0	149.7
6	104.3	104.3	104.3	104.3	104.3	104.3	104.3	104.3	104.0	104.0	104.0	104.0	104.0	104.0	104.0
7	194.6	194.3	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6	194.6
8 (H_L)	180.8	200.4	180.8	180.8	180.8	168.2	168.2	200.4	180.8	180.8	180.8	168.5	168.5	200.4	181.1
9	149.7	159.4	149.7	149.7	149.7	149.7	149.7	159.4	149.7	149.7	149.7	149.7	149.7	159.7	150.0
10 (K_L)	208.6	228.8	208.3	208.3	208.3	208.3	208.3	228.8	208.6	208.6	208.6	208.6	208.6	228.8	208.6
11 (A_L)	119.5	119.5	119.5	119.5	119.3	106.9	106.9	106.9	119.5	119.5	119.5	106.9	106.9	106.9	119.5
12	149.7	164.4	149.7	149.7	134.8	149.7	149.7	164.4	149.7	149.7	134.8	149.7	149.7	164.4	149.7

4.3.3.3 Angle Generation Error Calculation

Table 4.13: Angle generation error calculation

Motor	Error Calculations (degree)														
	Step 1			Step 2			Step 3			Step 4			Step 5		
	θ_{01}	θ_{v1}	θ_{g1}	θ_{02}	θ_{v2}	θ_{g2}	θ_{03}	θ_{v3}	θ_{g3}	θ_{04}	θ_{v4}	θ_{g4}	θ_{05}	θ_{v5}	θ_{g5}
H_R	1.4	1.4	1.1	1.1	0.9	0.6	0.6	0.6	1.7	1.7	0.9	1.2	1.2	1.2	1.7
K_R	-0.6	-0.6	-0.6	-0.6	-0.3	-0.6	-0.6	-0.9	-0.9	-0.9	-0.3	-0.6	-0.6	-0.9	-0.9
A_R	0.6	0.6	0	0	0	0	0	0	0	0	0	0	0	0	0
H_L	-0.6	-0.3	-0.6	-0.6	-0.6	-0.9	-0.9	-0.3	-0.6	-0.6	-0.6	-1.2	-1.2	-0.3	-0.9
K_L	1.2	0.9	1.5	1.5	1.5	1.5	1.5	0.9	1.2	1.2	1.2	1.2	1.2	0.9	1.2
A_L	0.6	0.6	0.6	0.6	0.8	0.3	0.3	0.3	0.6	0.6	0.6	0.3	0.3	0.3	0.6

* H = hip, K = knee, A = ankle, subset R = right leg, subset L = left leg

4.3.3.4 Angle Generation Percentage of Accuracy Calculation

Table 4.14: Angle generation percentage of accuracy calculation

Motor	**Percent of Accuracy Calculations (%)**														
	Step 1			**Step 2**			**Step 3**			**Step 4**			**Step 5**		
	θ_{01}	θ_{v1}	θ_{g1}	θ_{02}	θ_{v2}	θ_{g2}	θ_{03}	θ_{v3}	θ_{g3}	θ_{04}	θ_{v4}	θ_{g4}	θ_{05}	θ_{v5}	θ_{g5}
H_R	98.8	98.8	99.1	99.1	99.1	99.5	99.5	99.5	98.7	98.7	99.1	99.0	99.0	99.0	98.7
K_R	99.3	99.3	99.3	99.3	99.6	99.3	99.3	99.0	99.0	99.0	99.6	99.3	99.3	99.0	99.0
A_R	99.7	99.7	100	100	100	100	100	100	100	100	100	100	100	100	100
H_L	99.7	99.9	99.7	99.7	99.7	99.5	99.5	99.9	99.7	99.7	99.7	99.3	99.3	99.9	99.5
K_L	99.4	99.6	99.3	99.3	99.3	99.3	99.3	99.6	99.4	99.4	99.4	99.4	99.4	99.6	99.4
A_L	99.5	99.5	99.5	99.5	99.3	99.7	99.7	99.7	99.5	99.5	99.5	99.7	99.7	99.7	99.5

* H = hip, K = knee, A = ankle, subset R = right leg, subset L = left leg

4.3.4 Actual trajectory generation

4.3.4.1 Actual First Step Trajectory

Table 4.15: Actual First step angle distribution

Angle	θ_0	θ_v	θ_g
θ_{H1}	-30.8°	-50.4°	-30.8°
θ_{K1}	58.6°	78.8°	58.3°
θ_{A1}	-30.5°	-30.5°	-30.5°

*t_f = 1.1s

The general equation is obtained by inserting the angle distribution data into the formulas in section 3.3.2.1. θ_0 is the initial angle when the foot is at double support phase, θ_v is the maximum angle during the lifting motion or also known as via point and θ_g is the final angle after putting down the leg or also known as goal point. Then the general equation is used to plot the trajectory using matlab. The matlab algorithm can be seen in Appendix D.

General equation for hip, θ_H:

$$\theta_{upH1}(t) = 29.45t^3 - 48.6t^2 - 30.8$$

$$\theta_{downH1}(t) = -29.45t^3 + 48.6t^2 - 50.4$$

General equation for knee, θ_K:

$$\theta_{upK1}(t) = -30.52t^3 + 50.27t^2 + 58.6$$

$$\theta_{downK1}(t) = 30.63t^3 - 50.45t^2 - 0.2t + 78.8$$

General equation for ankle, θ_A:

$$\theta_{upA1}(t) = -30.5$$

$$\theta_{downA1}(t) = -30.5$$

4.3.4.2 Actual Second Step Trajectory

Table 4.16: Actual Second step angle distribution

Angle	θ_0	θ_v	θ_g
θ_{H2}	23.1^o	50.7^o	30.5^o
θ_{K2}	-59.5^o	-79.7^o	-59.5^o
θ_{A2}	38.1^o	38.1^o	30.2^o

*t_f = 1.1s

General equation for hip, θ_H:

$\theta_{upH2}(t) = - 37.3t^3 + 63.84t^2 + 23.1$

$\theta_{downH2}(t) = 34.52t^3 - 59.26t^2 + 5.05t + 50.7$

General equation for knee, θ_K:

$\theta_{upK2}(t) = 30.35t^3 – 50.1t^2 - 59.5$

$\theta_{downK2}(t) = -30.35t^3 + 50.1t^2 – 79.7$

General equation for ankle, θ_A:

$\theta_{upA2}(t) = - 4.45t^3 + 4.9t^2 + 38.1$

$\theta_{downA2}(t) = 7.42t^3 – 9.79t^2 – 5.39t + 38.1$

CHAPTER 5

ANALYSIS AND DISCUSSIONS

5.1 Hobby Servo Motor Control

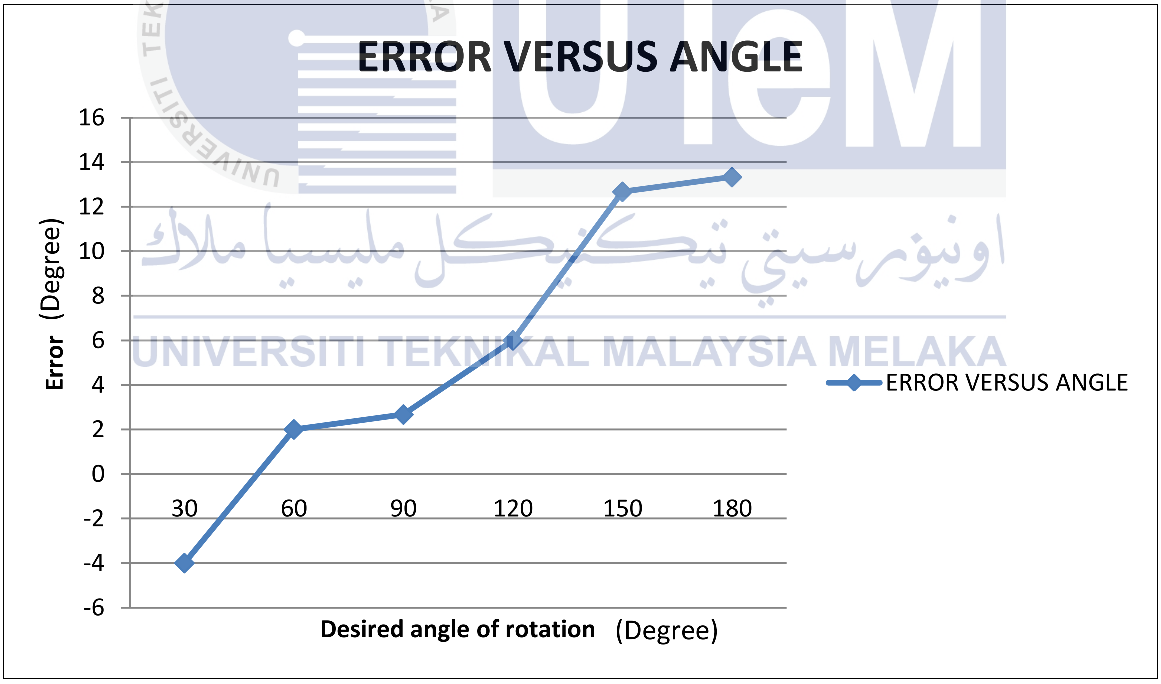


Figure 5.1: Error versus Angle graph

It can be observed from the graph in Figure 5.1 that the error is increasing with the increasing desired angle of rotation. Besides, the maximum angle reachable by the servo motor during the experiment is 166.67 degree which is less than the maximum servo motor angle of 180 degree. This may be due to the output clock frequency of the Arduino to be less than the required frequency of the servo motor causing the maximum angle of rotation cannot be reached (e.g. PWM pulse generated by Arduina has shorter period than required by servo motor). Since servo motor uses PWM pulse for its rotation, having a different pulse will affect its rotation. For example, the servo motor needs a 20ms pulse or period but the output pulse of the Arduino may be 18ms which is lower than the required specs. Therefore, there will be some deviation from the desired value or rotation. Since the performance is inadequate, Dynamixel AX-12A Servo Motor is selected as the main actuator in this project.

5.2 Static Walking Sequence

5.2.1 Expected Trajectory Generation

5.2.1.1 Expected First Step Trajectory

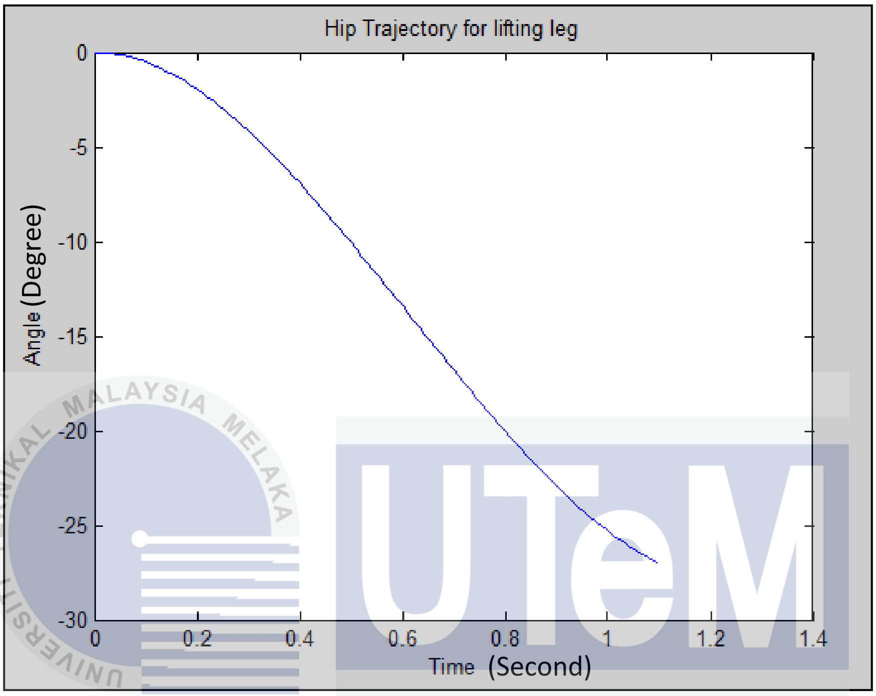


Figure 5.2: Expected Trajectory generation for $\theta_{upH1}(t)$

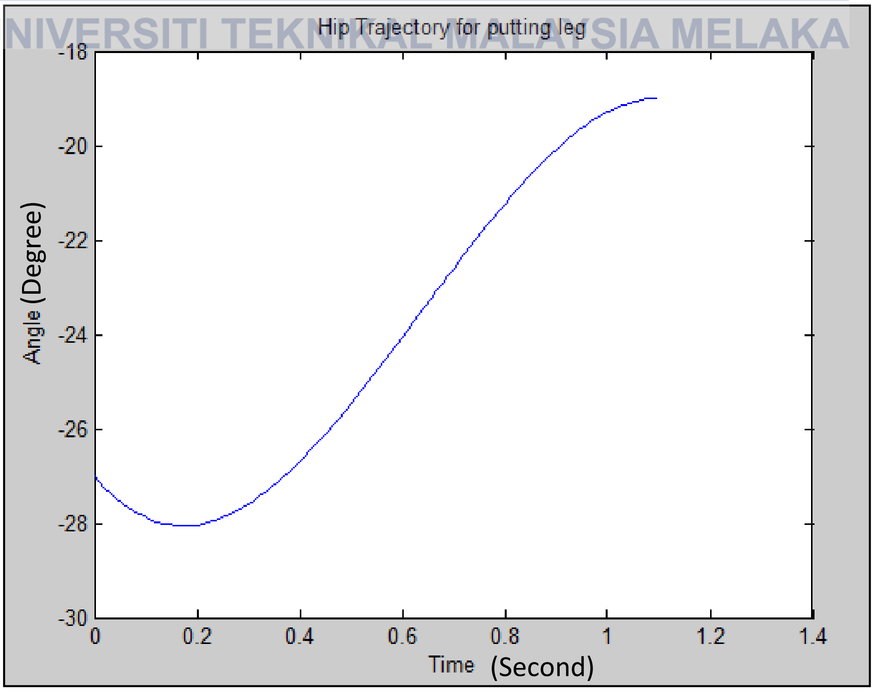


Figure 5.3: Expected Trajectory generation for $\theta_{downH1}(t)$

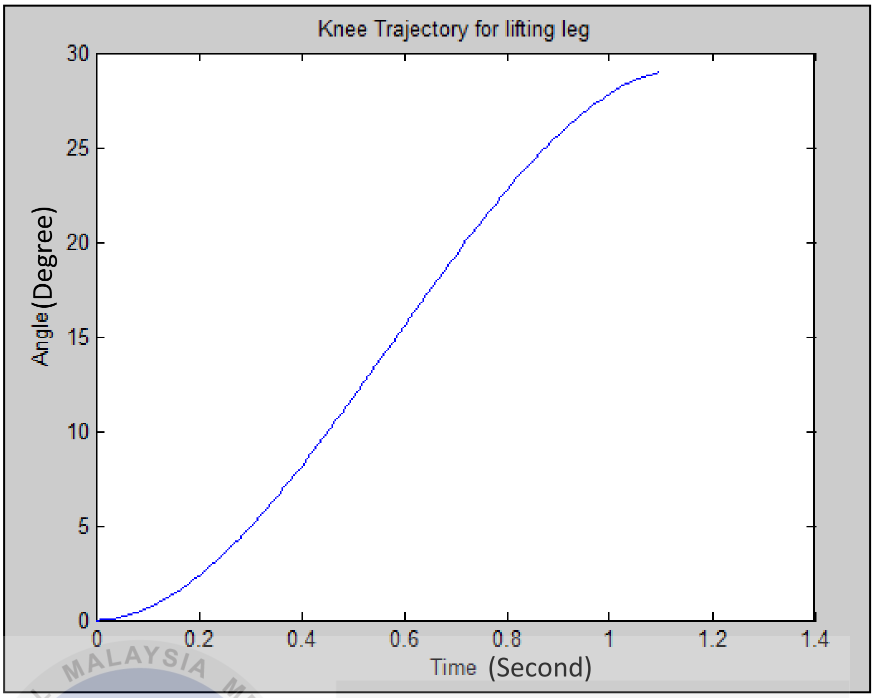


Figure 5.4: Expected Trajectory generation for $\theta_{upK1}(t)$

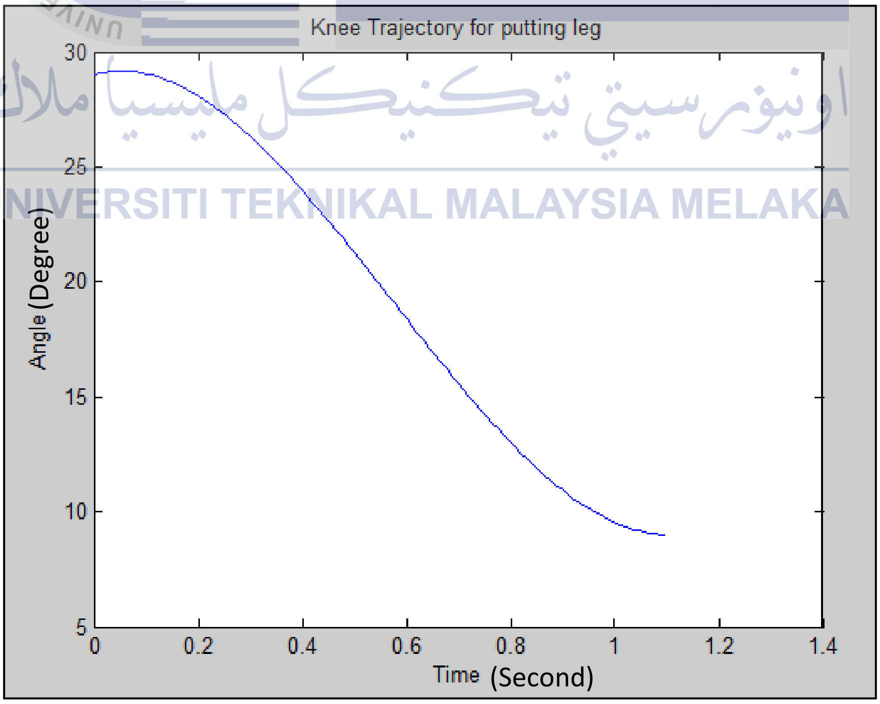


Figure 5.5: Expected Trajectory generation for $\theta_{downK1}(t)$

Figure 5.2 show the trajectory generation for the hip during lifting motion where it starts from 0^o and reaches – 27^o at the end of 1.1s while Figure 5.3 shows the trajectory generation for the hip during putting down leg motion where it starts from -27^o and reaches -19^o at the end of 1.1s. Similarly, Figure 5.4 show the trajectory generation for the knee during lifting motion where it starts from 0^o and reaches 29^o at the end of 1.1s while Figure 5.5 shows the trajectory generation for the knee during putting down leg motion where it starts from 29^o and reaches 9^o at the end of 1.1s. The starting angle and ending angle can be seen in Table 4.2. Besides, the execution time for each step is 1.1s according to the selected walking combination time per step, anticlockwise rotation of motor is assumed to be negative angle and clockwise rotation of motor is assumed to be positive angle. Figure 5.2 - Figure 5.5 is the trajectory generations for first step of the biped robot.

5.2.1.2 Expected Second Step Trajectory

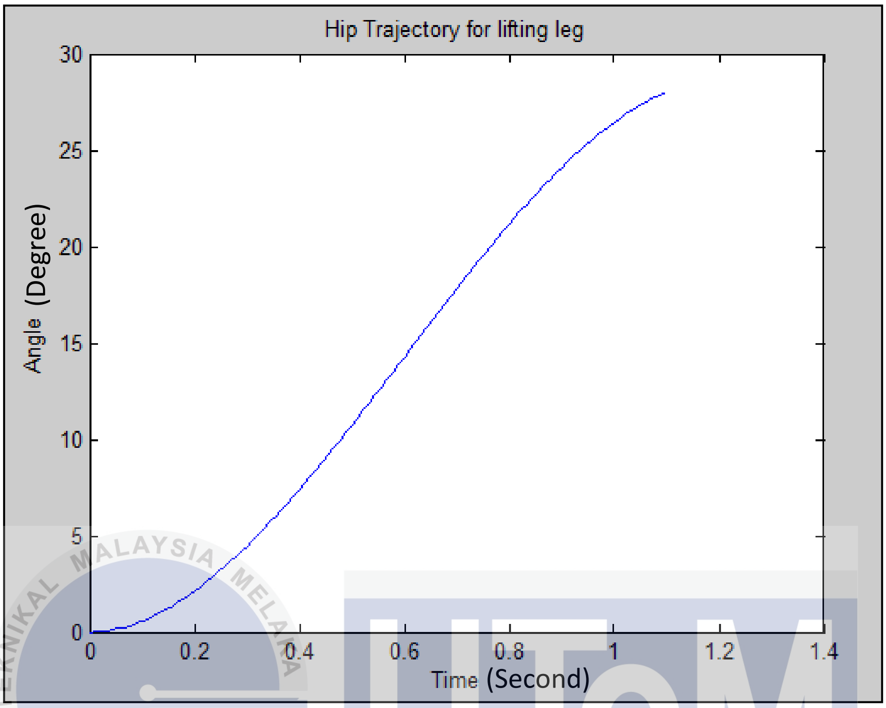


Figure 5.6: Expected Trajectory generation for $\theta_{upH2}(t)$

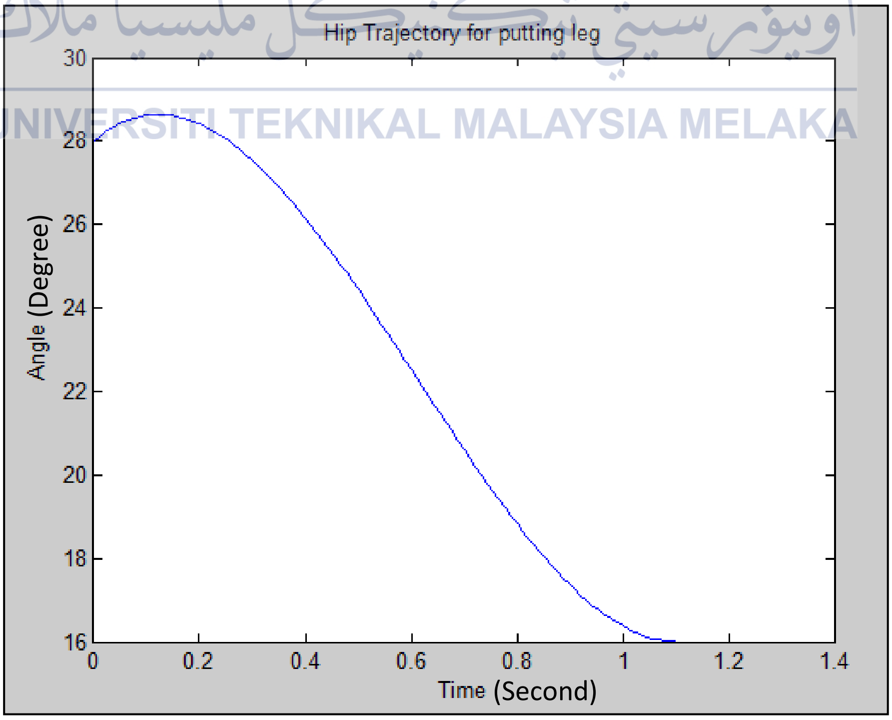


Figure 5.7: Expected Trajectory generation for $\theta_{downH2}(t)$

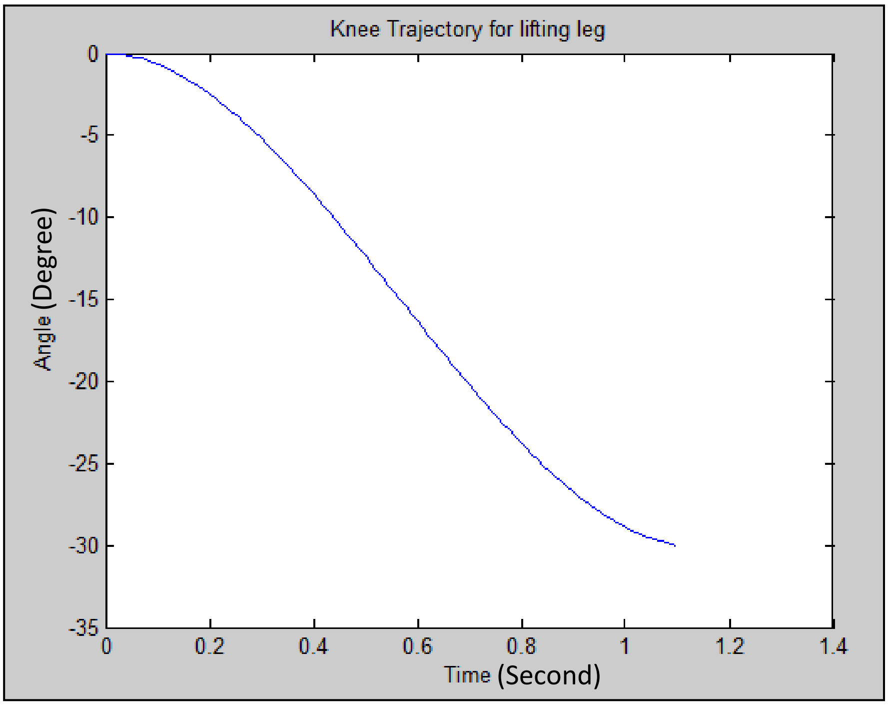


Figure 5.8: Expected Trajectory generation for $\theta_{upK2}(t)$

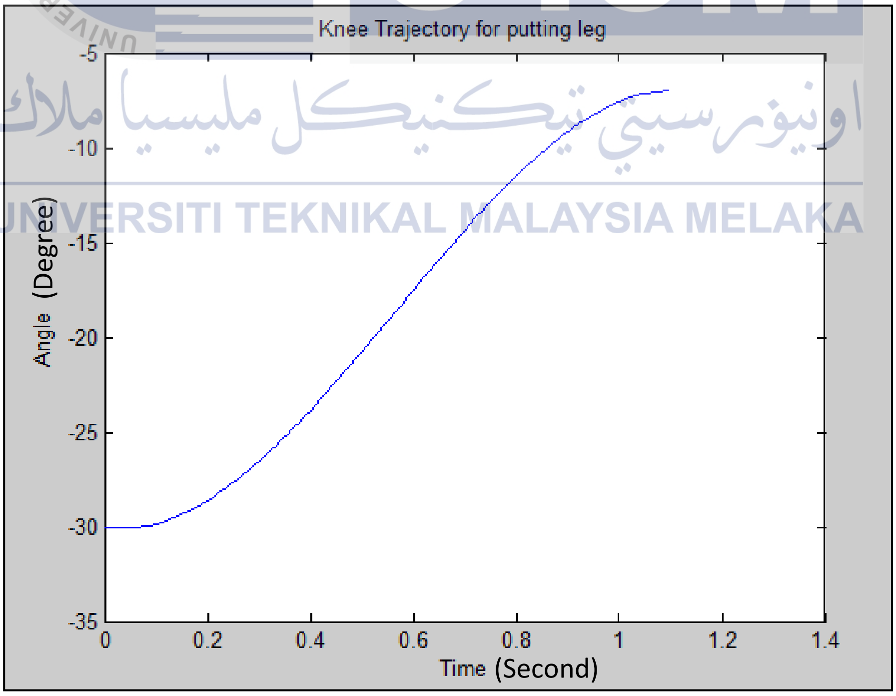


Figure 5.9: Expected Trajectory generation for $\theta_{downK2}(t)$

Figure 5.6 show the trajectory generation for the hip during lifting motion where it starts from 0^o and reaches 28^o at the end of 1.1s while Figure 5.7 shows the trajectory generation for the hip during putting down leg motion where it starts from 28^o and reaches 16^o at the end of 1.1s. Similarly, Figure 5.8 show the trajectory generation for the knee during lifting motion where it starts from 0^o and reaches - 30^o at the end of 1.1s while Figure 5.9 shows the trajectory generation for the knee during putting down leg motion where it starts from - 30^o and reaches - 7^o at the end of 1.1s. The starting angle and ending angle can be seen in Table 4.3. Figure 5.6 - Figure 5.9 is the trajectory generations for second step of the biped robot.

5.2.2 Walking combination

This experiment is carried to find the most suitable combination of length of stride, height of stride and also speed of stride in a walking motion. Therefore, initially, the length of stride and speed of stride is kept constant as low while the height of stride is altered. 3 combinations of walking motion are produced which includes “low length, low height, low speed”, “low length, medium height, low speed” and “low length, high height, low speed”. From these 3 combinations, the best height of stride is chosen. Speed per step and deviation from straight line is taken into consideration when choosing since both of these criteria is affected by the height. Based on Table 4.4 to Table 4.6, the walking combination that has the highest speed per step and lowest deviation from straight line is walking combination 2 with $0.501cms^{-1}$ per step and 0.4cm deviation. Hence, medium height of stride is chosen as the best height.

Next, the height of stride is kept constant as medium and the speed of stride are kept constant as low while the length of stride is being altered. 2 combinations of walking motion are produced which include “medium length, medium height, low speed” and “high length, medium height, low speed”. From these 2 combinations, the best length of stride is chosen. Similarly, the speed per step and deviation from straight line is taken into consideration when choosing since both of these criteria is affected by the length. Based on Table 4.7 and Table 4.8, the walking combination that has the highest speed per step is walking combination 4 with $0.734cms^{-1}$ per step while the lowest deviation from straight line is walking combination 5 with 0.34cm deviation. Even though walking combination 5 have the lower deviation from straight line, walking combination 4 is still being chosen as the better length of stride. This is because during the experiment, by implementing walking combination 5, the biped robot tends to be unstable with more sways and slips. Thus, medium length of stride is chosen as the best length.

Finally, the length of stride and height of stride is kept constant as medium while the speed of stride is being altered. 2 combinations of walking motion are produced which include “medium length, medium height, medium speed” and “medium length, medium height, high speed”. From these 2 combinations, the best speed of stride is chosen. The speed per step and deviation from straight line is taken into consideration when choosing since both of these criteria is affected by the speed per step. Based on Table 4.9 and Table 4.10, the walking combination that has the highest speed per step and deviation from straight line is walking combination 7. However, walking combination 6 is still being chosen as the better speed. This is because when implementing walking combination 7, the biped robot has the worse instability, large sways and also larger slips. Therefore, medium speed of stride is being chosen as the best speed.

As a conclusion, the best walking combination for the biped robot walking motion is the walking combination 6 with medium length of stride, medium height of stride and medium speed of stride. The walking algorithm or coding for the selected walking combination 6 is provided at Appendix E and the algorithm or coding to get feedback from the Dynamixel AX-12A Servo Motor is provided at Appendix F.

5.2.3 Error analysis

The angle generation value is taken based on the selected walking combination 6 with medium length of stride, medium height of stride and medium speed of stride. Desired angle generation value will be taken directly from the coding and the actual angle generation value will be taken from the Dynamixel AX-12A Servo Motor feedback via the Arduino USB Serial Light Adapter as an additional serial port.

5.2.3.1 Angle Generation Error Calculation

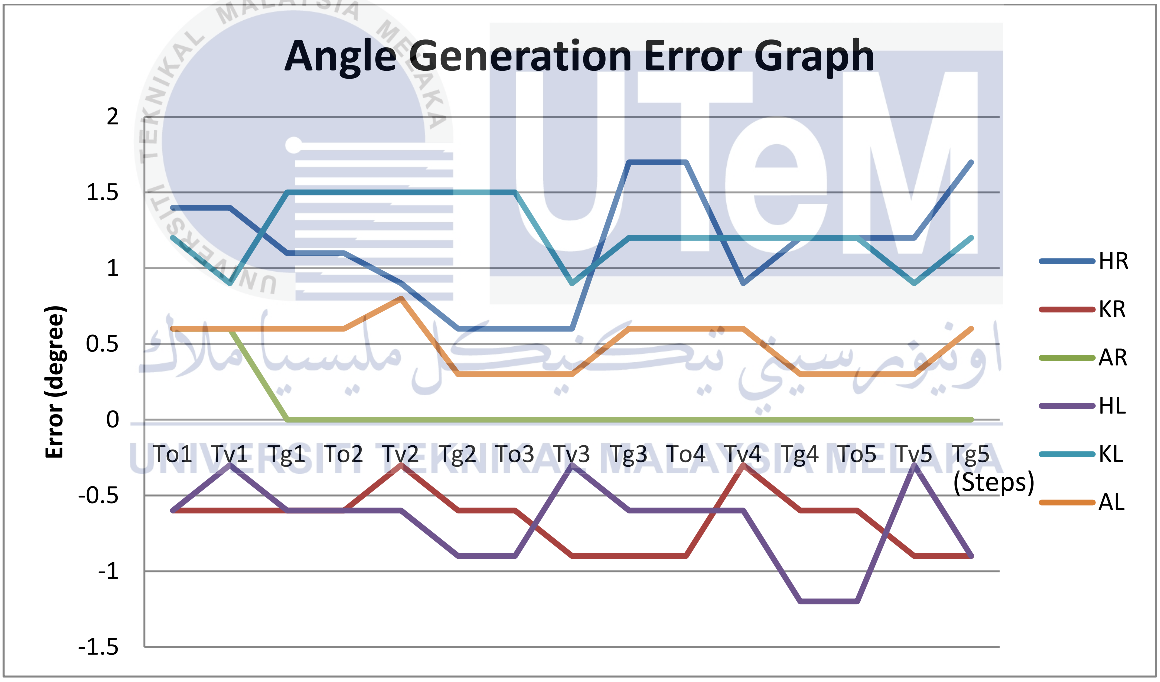


Figure 5.10: Angle generation error graph

This graph is produced based on the data in Table 4.13 and T in this graph represents the symbol θ. Based on Figure 5.10, for the right leg, the highest error occurs at the hip motor, H_R followed by knee motor, K_R and the lowest error occurs at the ankle motor, A_R. The highest error for H_R is 1.7^o at T_{g3}, T_{o4} and T_{g5} while the lowest error for A_R is 0^o from T_{g1} until T_{g5}. On the other hand, for the left leg, the highest error occurs at the knee motor, K_L followed by hip motor, H_L and the lowest error occurs at the ankle motor,

A_L. The highest error for K_L is 1.5^o at T_{g1}, T_{o2}, T_{v2}, T_{g2} and T_{o3} while the lowest error for A_L is 0.3^o at T_{g2}, T_{o3}, T_{v3}, T_{g4}, T_{o5} and T_{v5}. From Figure 5.10 also, it can be observed that parts that experiences high possibility of having higher error are the hip and knee motor. This is because in the walking motion algorithm, hip and knee motors are the main actuators that are frequently used with higher angle rotation compared to other motors such as ankle motor. Therefore, ankle part will experience less error.

5.2.3.2 Angle Generation Percentage of Accuracy Calculation

Based on Table 4.14, for right leg, it can be observed that the highest percentage of accuracy is at the ankle motor, A_R with 100% from T_{g1} until T_g5 while the lowest percentage of accuracy is at the hip motor, H_R with 98.7% at T_{g3}, T_{o4} and T_{g5}. This prove that the error analysis done at section 5.2.3.1 is true where lower error will have higher accuracy neglecting the amount of total angle rotated. Meanwhile, for the left leg, the highest percentage of accuracy happens at the hip motor, H_L with 99.9% at T_{v1}, T_{v3} and T_{v5} while the lowest percentage of accuracy occurs at the knee motor, K_L with 99.3% at T_{g1}, T_{o2}, T_{v2}, T_{g2}, T_{o3}. According to section 5.2.3.1, the lowest error for left leg occurs at ankle motor, A_L rather than hip motor, H_L. This can be explained by the total angle rotated by the motors. When implementing the walking algorithm, hip and knee motor have larger angle rotation compared to ankle. Hence, when the total angle rotated is larger, the effect of error on the calculation of percentage of accuracy is significant smaller. Besides, the difference between the highest and lowest percentage of accuracy between H_L and A_L is only 0 – 6% which is a small value.

As a conclusion, Dynamixel AX-12A has low error and high accuracy in its performance test. Thus, it is suitable to be used in the biped robot project.

5.2.4 Actual trajectory generation

5.2.4.1 Actual First Step Trajectory

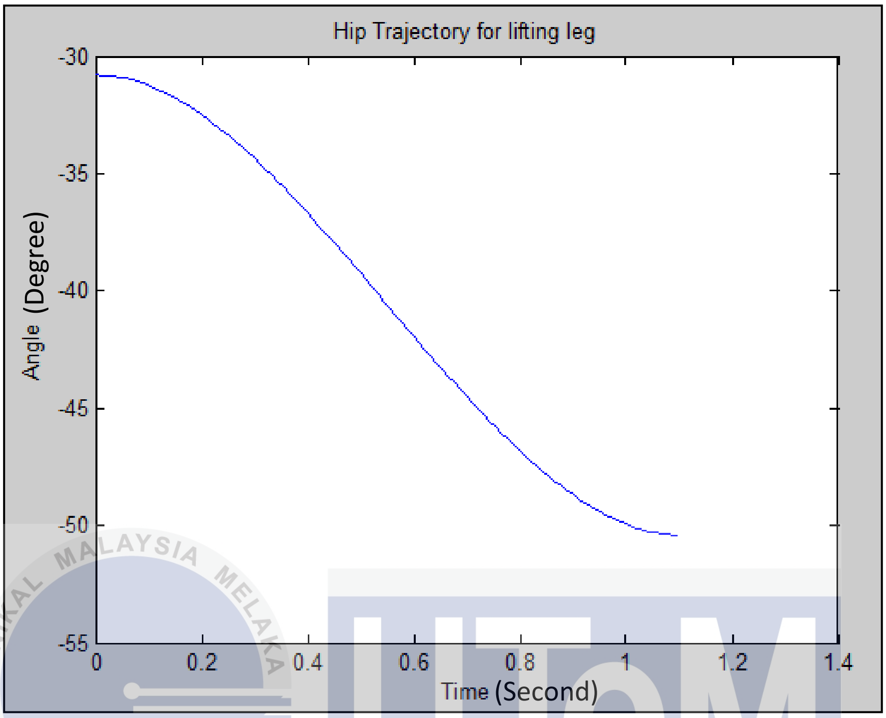


Figure 5.11: Actual trajectory generation for $\theta_{upH1}(t)$

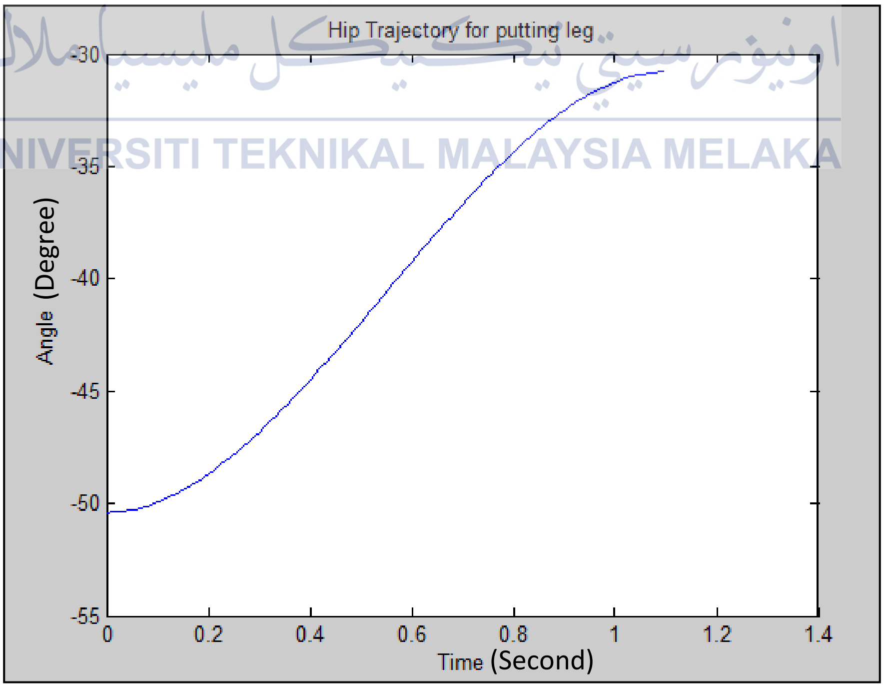


Figure 5.12: Actual trajectory generation for $\theta_{downH1}(t)$

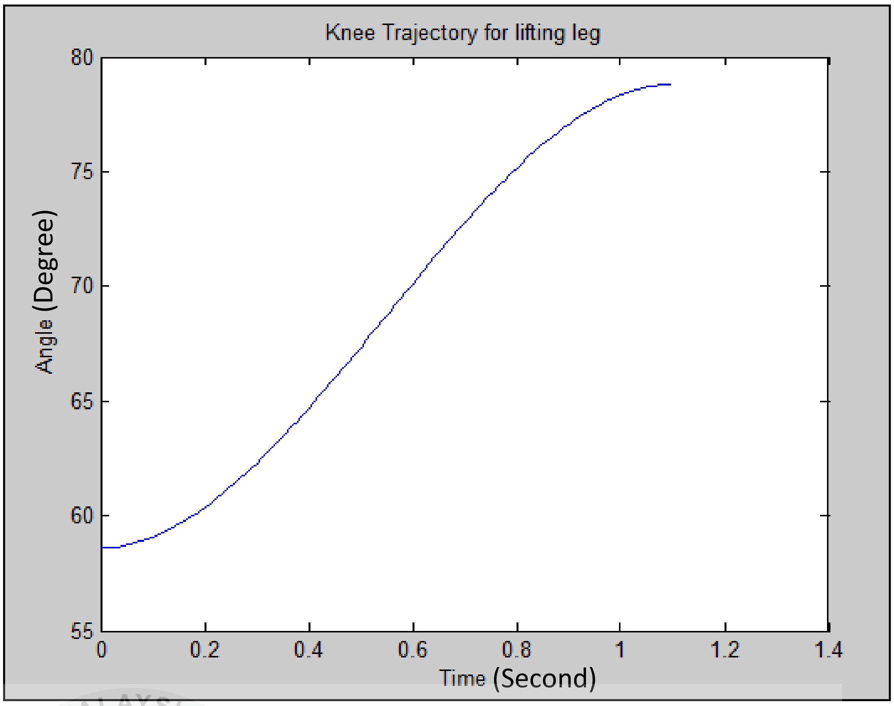


Figure 5.13: Actual trajectory generation for $\theta_{upK1}(t)$

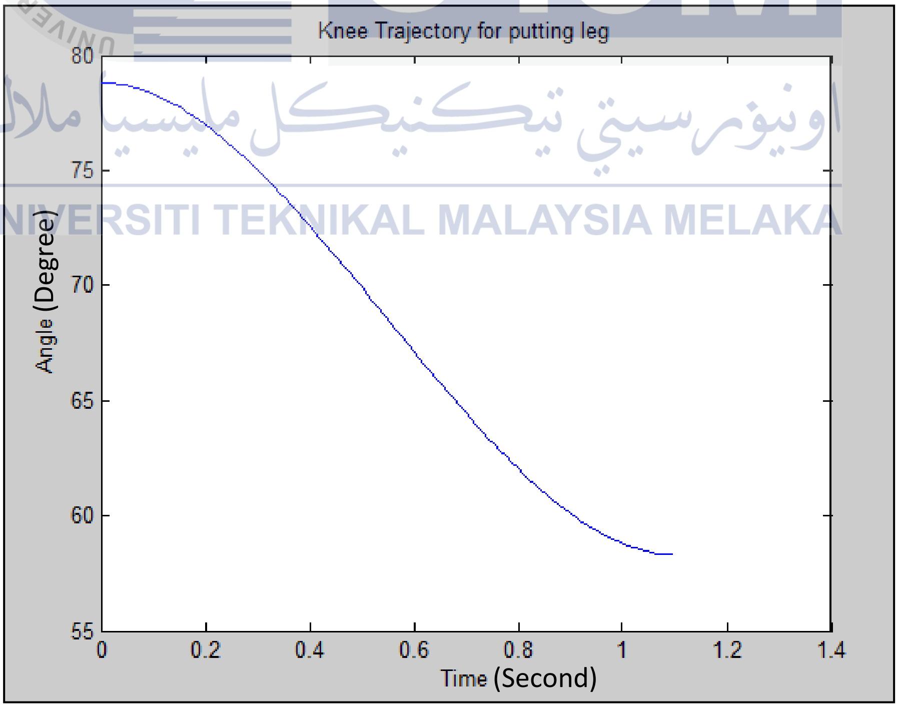


Figure 5.14: Actual trajectory generation for $\theta_{downK1}(t)$

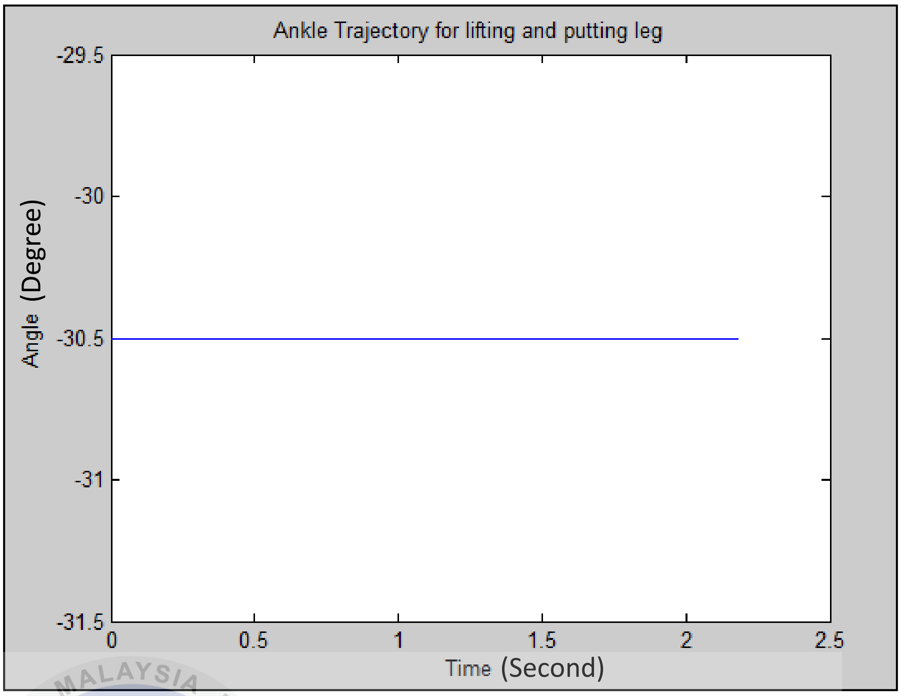


Figure 5.15: Actual trajectory generation for $\theta_{upA1}(t)$ and$\theta_{downA1}(t)$

Figure 5.11 show the trajectory generation for the hip during lifting motion where it starts from -30.8^o and reaches – 50.4^o at the end of 1.1s while Figure 5.12 shows the trajectory generation for the hip during putting down leg motion where it starts from -50.4^o and reaches -30.8^o at the end of 1.1s. Similarly, Figure 5.13 show the trajectory generation for the knee during lifting motion where it starts from 58.6^o and reaches 78.8^o at the end of 1.1s while Figure 5.14 shows the trajectory generation for the knee during putting down leg motion where it starts from 78.8^o and reaches 58.3^o at the end of 1.1s. Figure 5.15 shows the trajectory generation for ankle during lifting and putting down leg motion where it starts from -30.5^o and maintain it until the end of 2.188s. The starting angle and ending angle can be seen in Table 4.15. Besides, the execution time for each step is 1.1s according to the selected walking combination time per step, anticlockwise rotation of motor is assumed to be negative angle and clockwise rotation of motor is assumed to be positive angle. Figure 5.11 - Figure 5.15 is the actual trajectory generations for first step of the biped robot.

5.2.4.2 Actual Second Step Trajectory

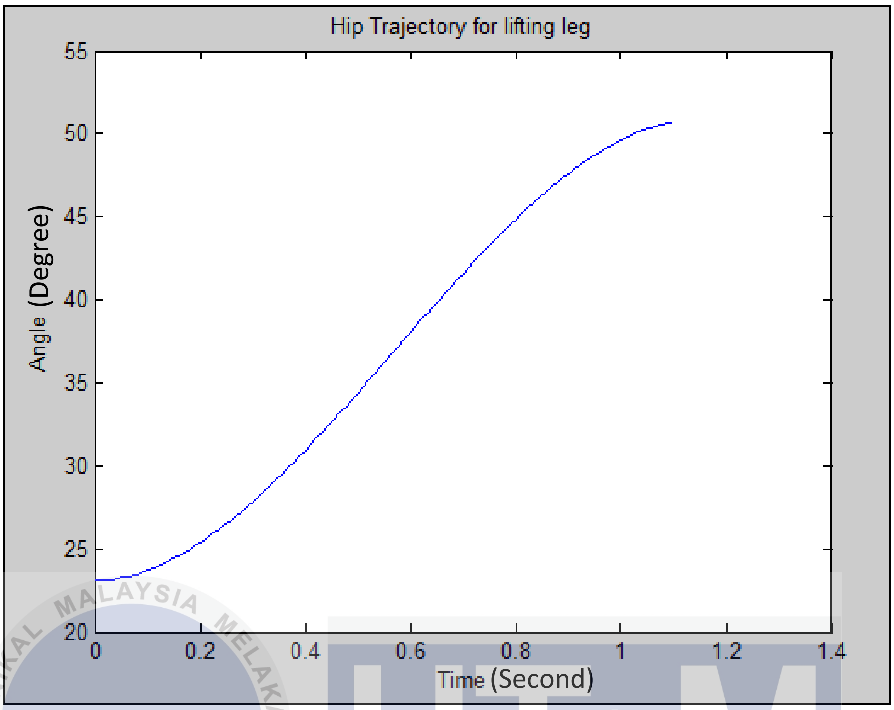


Figure 5.16: Actual trajectory generation for $\theta_{upH2}(t)$

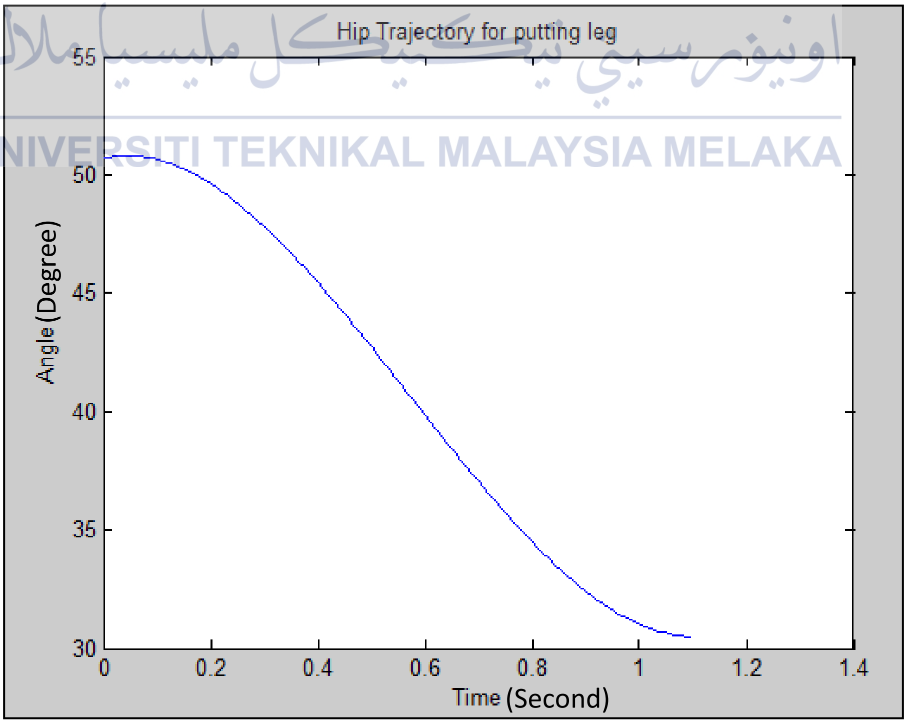


Figure 5.17: Actual trajectory generation for $\theta_{downH2}(t)$

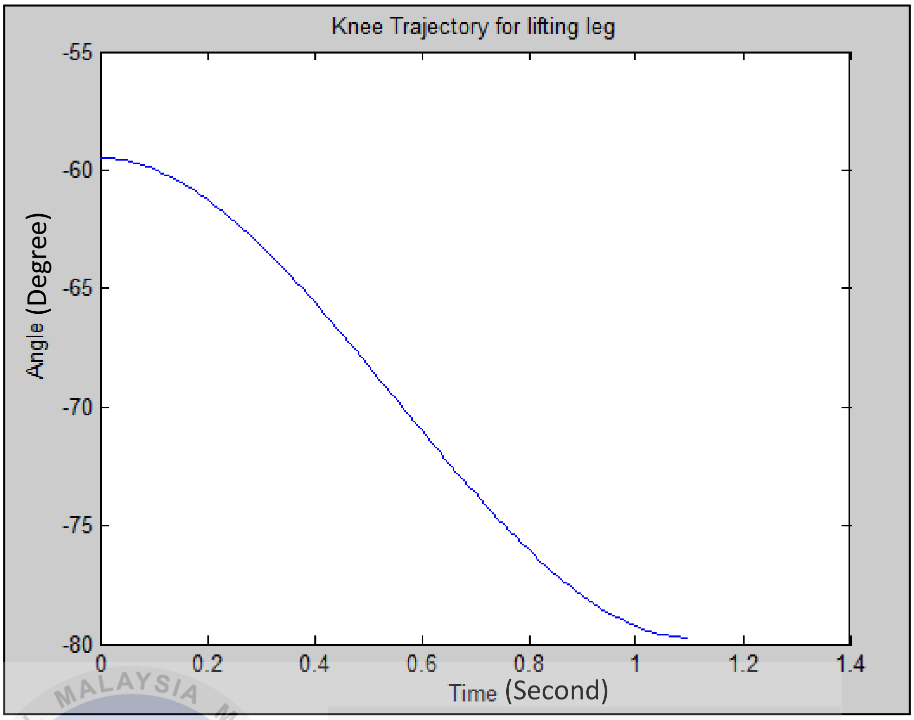


Figure 5.18: Actual trajectory generation for $\theta_{upK2}(t)$

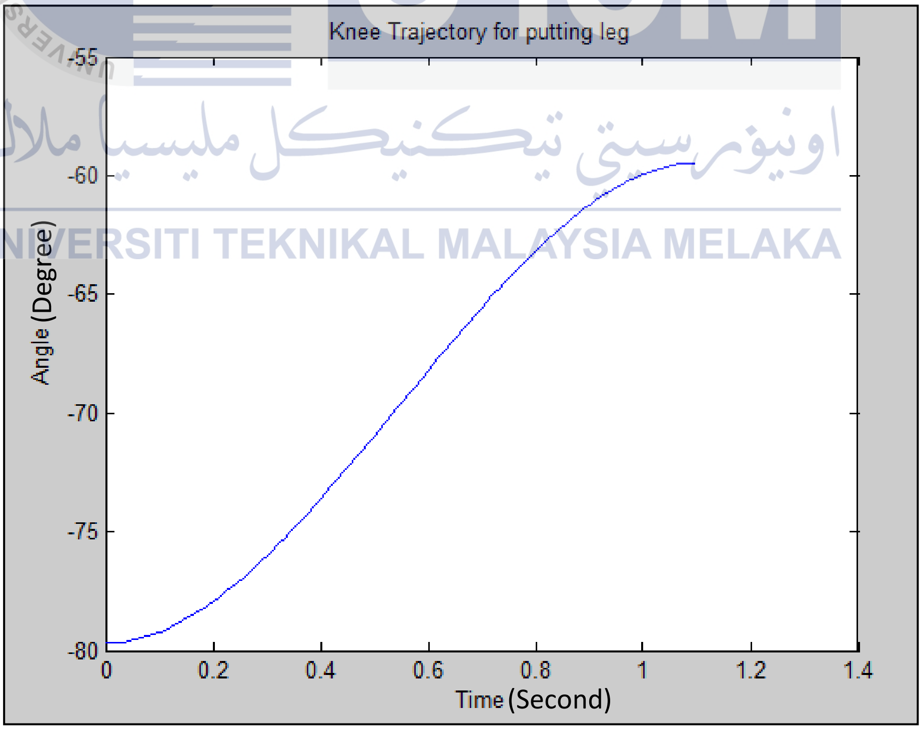


Figure 5.19: Actual trajectory generation for $\theta_{downK2}(t)$

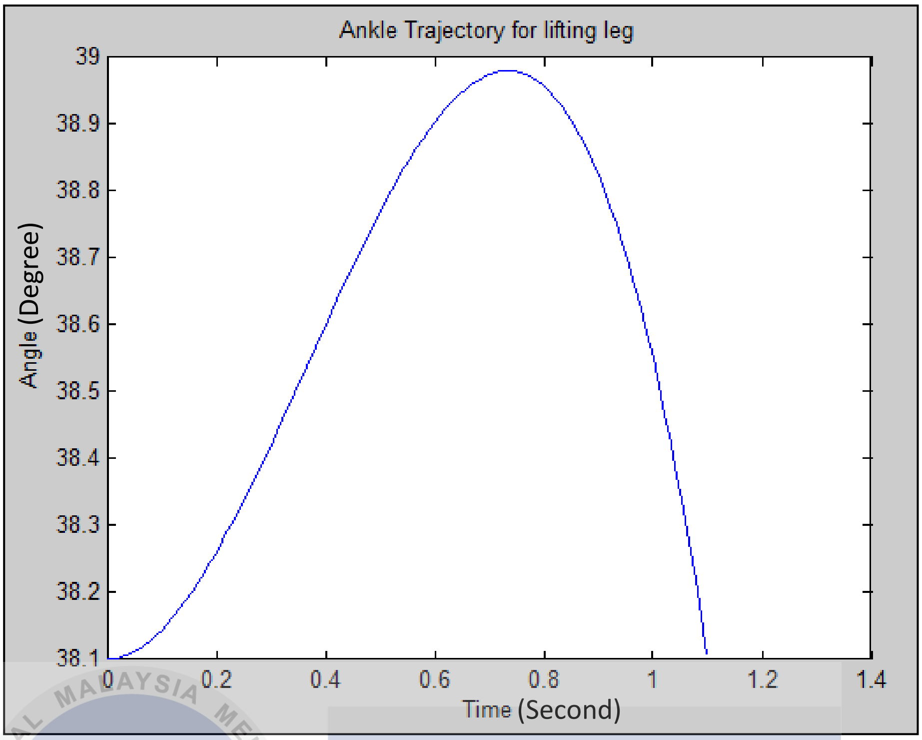


Figure 5.20: Actual trajectory generation for $\theta_{upA2}(t)$

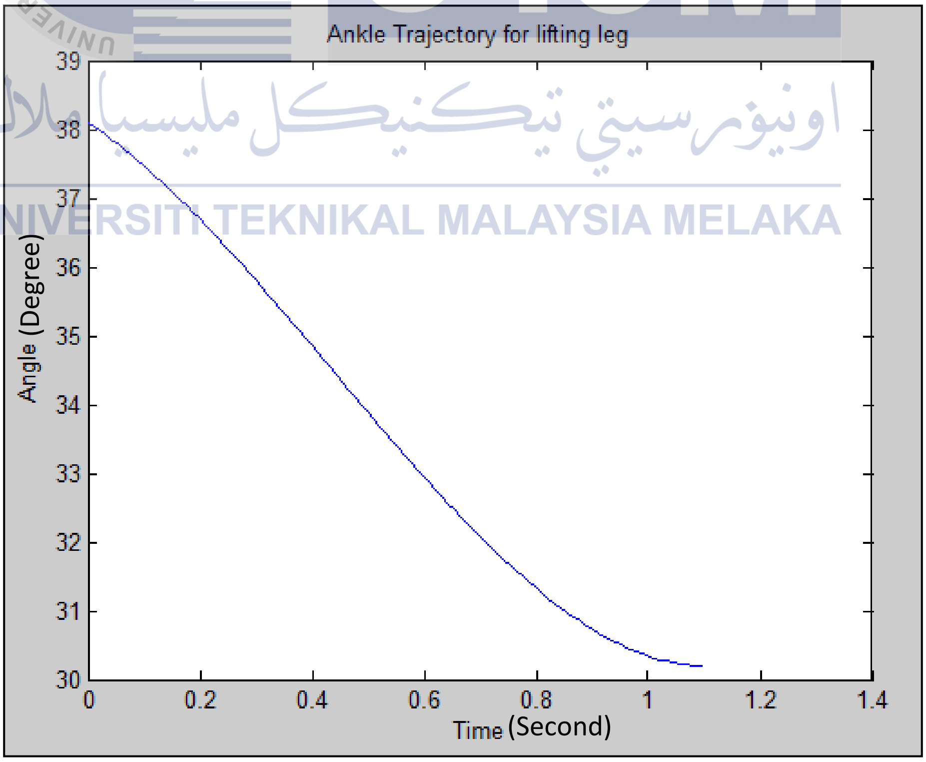


Figure 5.21: Actual trajectory generation for $\theta_{downA2}(t)$

Figure 5.16 show the trajectory generation for the hip during lifting motion where it starts from 23.1^o and reaches 50.7^o at the end of 1.1s while Figure 5.17 shows the trajectory generation for the hip during putting down leg motion where it starts from 50.7^o and reaches 30.5^o at the end of 1.1s. Similarly, Figure 5.18 show the trajectory generation for the knee during lifting motion where it starts from -59.5^o and reaches – 79.7^o at the end of 1.1s while Figure 5.19 shows the trajectory generation for the knee during putting down leg motion where it starts from - 79.7^o and reaches – 59.5^o at the end of 1.1s. Figure 5.20 show the trajectory generation for the ankle during lifting motion where it starts from -38.1^o, fluctuate due to noise before it returns back to 38.1^o and maintain it until the end of 1.1s while Figure 5.21 shows the trajectory generation for the ankle during putting down leg motion where it starts from 38.1^o and reaches 30.2^o at the end of 1.1s. The starting angle and ending angle can be seen in Table 4.16. Figure 5.16 - Figure 5.21 is the actual trajectory generations for second step of the biped robot.

5.2.5 Comparison between Expected and Actual trajectory generation

5.2.5.1 First Step Trajectory Comparison

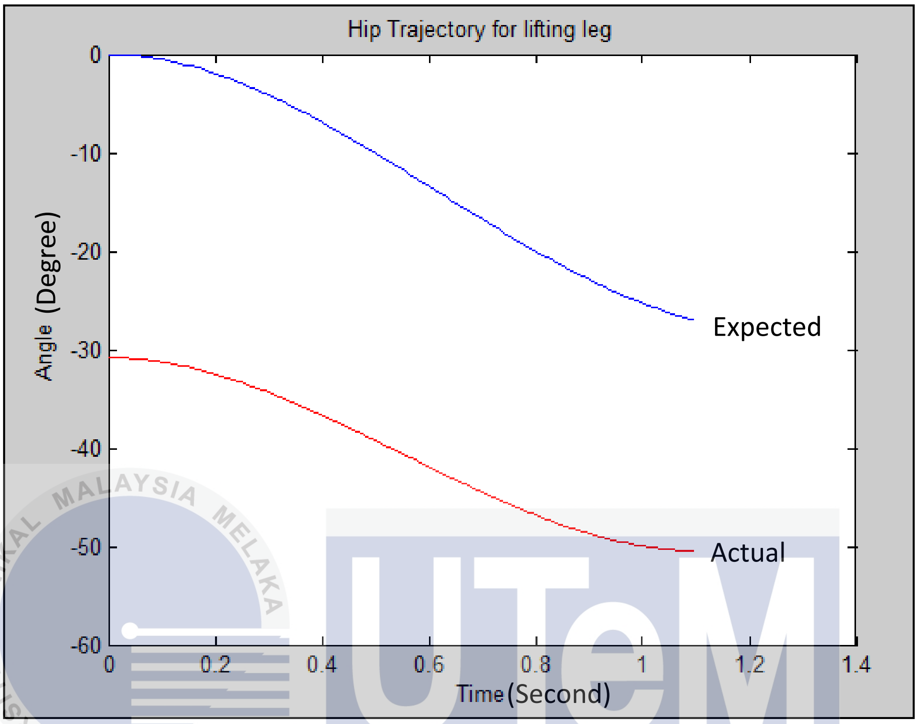


Figure 5.22: Comparison of trajectory generation for $\theta_{upH1}(t)$

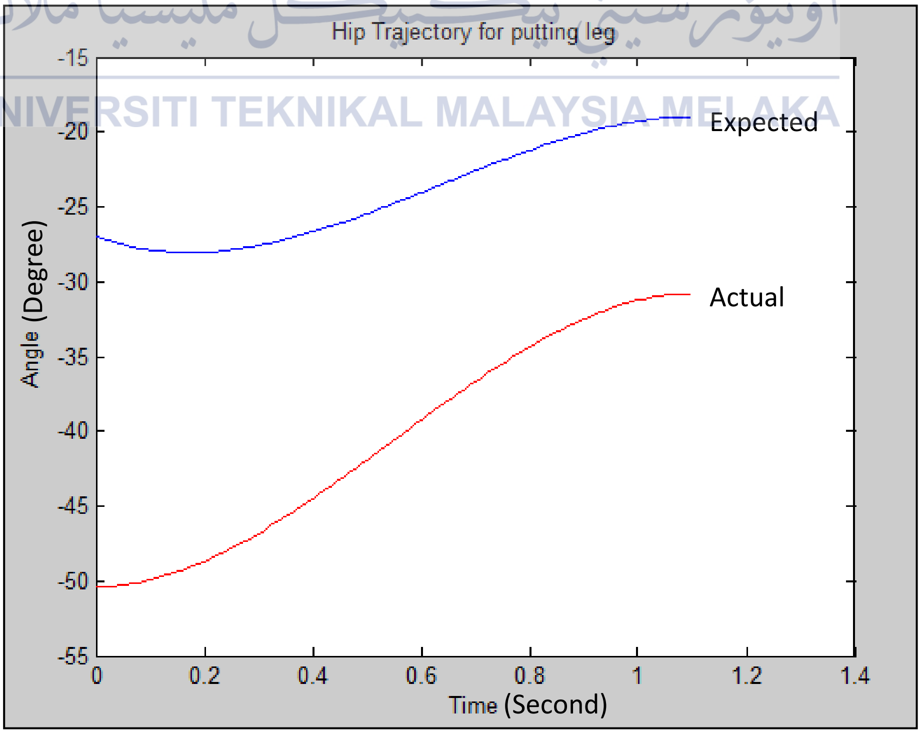


Figure 5.23: Comparison of trajectory generation for $\theta_{downH1}(t)$

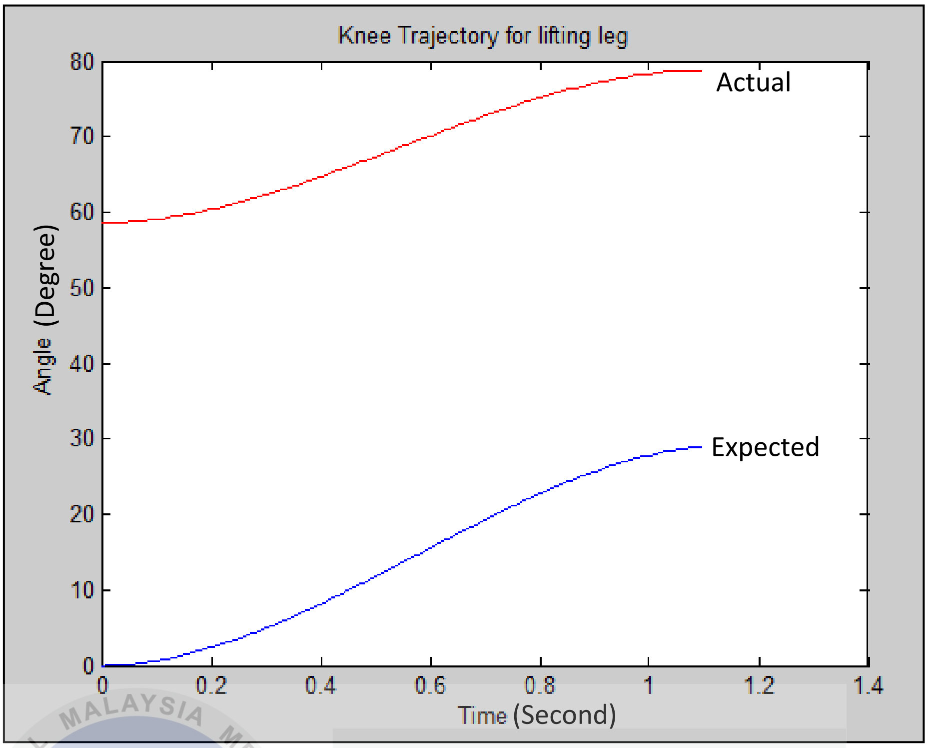


Figure 5.24: Comparison of trajectory generation for $\theta_{upK1}(t)$

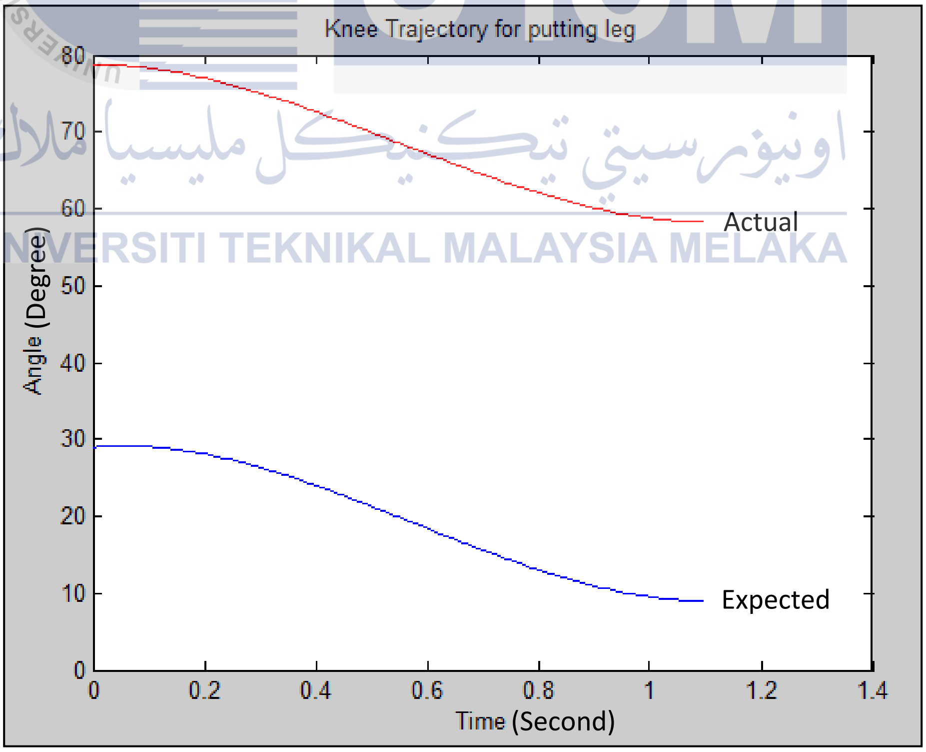


Figure 5.25: Comparison of trajectory generation for $\theta_{downK1}(t)$

5.2.5.2 Second Step Trajectory Comparison

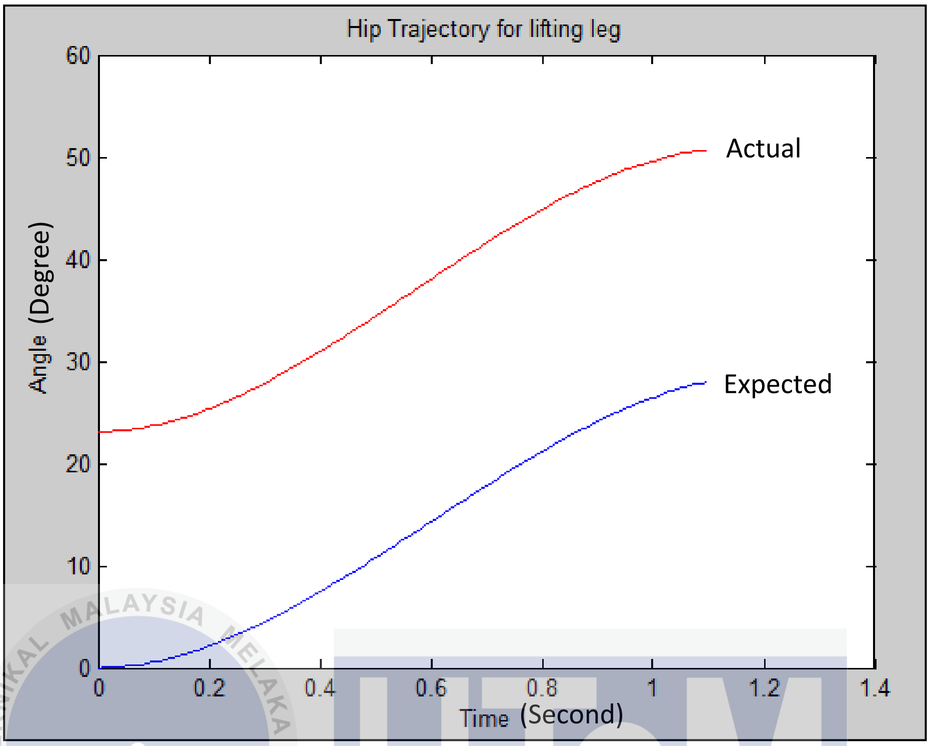


Figure 5.26: Comparison of trajectory generation for $\theta_{upH2}(t)$

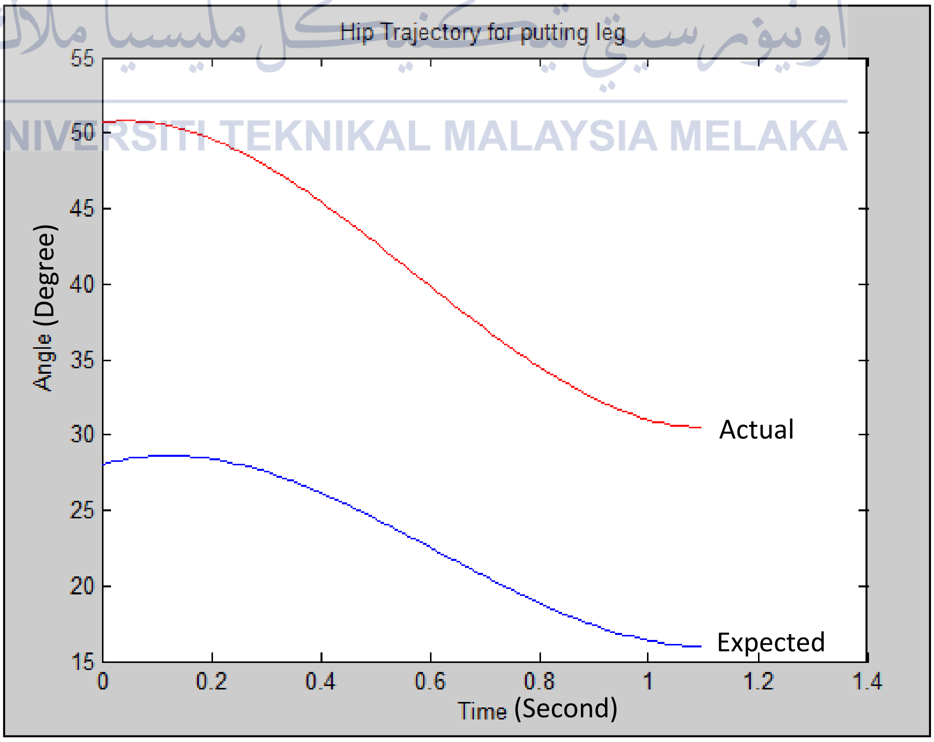


Figure 5.27: Comparison of trajectory generation for $\theta_{downH2}(t)$

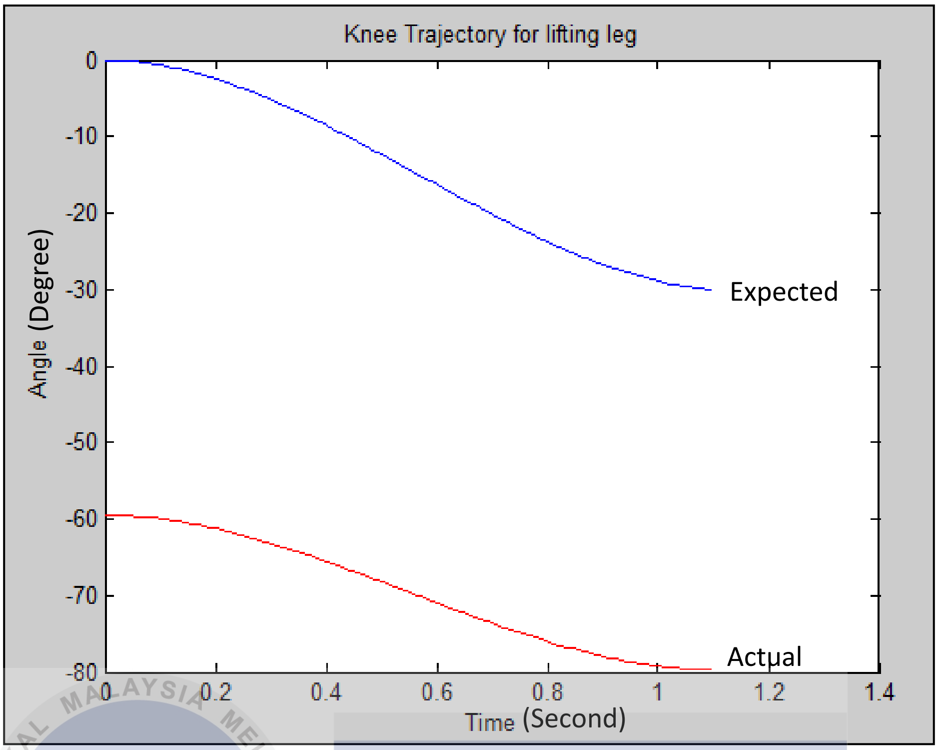


Figure 5.28: Comparison of trajectory generation for $\theta_{upK2}(t)$

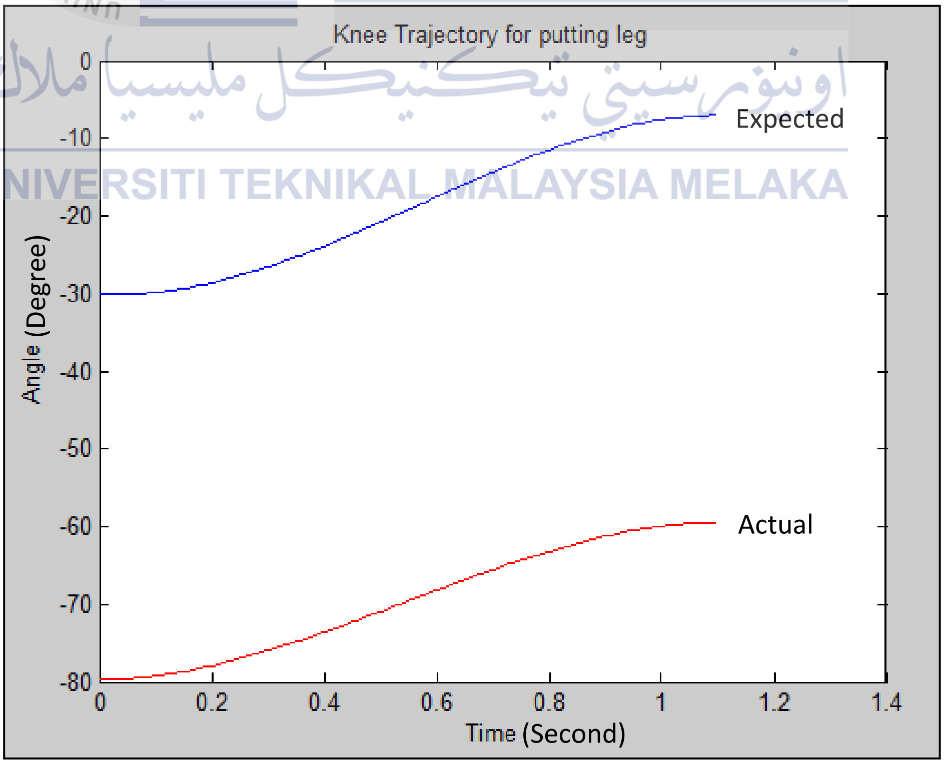


Figure 5.29: Comparison of trajectory generation for $\theta_{downK2}(t)$

Red line is for the actual trajectory generation and Blue line is for the expected trajectory generation. The ankle trajectory generation comparison is not included since there is no ankle trajectory generation for the first and second step from the expected trajectory generation. From Figure 5.22 until Figure 5.29, it can be observed that all of the expected and actual trajectory generations have the same pattern however there is angle gaps between them. This can be explained by the 'home' position assumed by the biped robot during the walking motion implementation. To stay stable before, during and after walking, the biped robot is required to be in a so called 'home' position as shown in Figure 5.30 where the hip motor rotate for 30^o, knee motor rotate for 60^o and ankle motor rotate for 30^o. This is the reason as to why there is a gap between expected and actual trajectory generation. In fact, by reducing the initial angle from the actual trajectory generation, it can be assumed that the gap decrease to a significant small value. Thus, it can be said that, the biped walking motion are able to imitate human walking motion if the 'home' position can be eliminated. More works have to be done so that the biped robot is able to obtain stable position at vertical straight position.

Figure 5.30: Biped robot 'home' position

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

Through methodology, the basic construction and analysis method done by other researcher can be learned which in turns serve as a fundamental to this project development. Using the knowledge gained, the required parameters such as the degrees of freedom, microcontroller and actuator has been determined. Then, the designing of the structure or the CAD drawing of the biped robot has also been completed using SolidWorks software. Nevertheless, basic experiment has been conducted to understand more on the Arduino microcontroller usage which is part of the electrical and electronic design. Moreover, hobby servo motor performance has been tested by using error analysis and it is not suitable to be used in this project. Then, the hardware construction for the biped robot is completed and by using the completed biped robot, experiments have been carried out to analyse the walking motion by using error analysis approach and also trajectory approach. The error analysis shows that Dynamixel AX-12A is the suitable actuator to be used in this project. Other than that, trajectory generation proves that the biped robot is able to imitate human walking motion if the initial 'home' position can be eliminated. Therefore, with all these being completed, it can be concluded that objective 1 which is to design and develop a biped robot with humanlike motion capability controlled by Arduino microprocessor and objective 2 which is to analyse the biped robot walking motion by static motion sequence and joint angle analysis have been achieved. In the future, it is possible to include the implementation of sensor such as gyro sensor for better stability analysis.

REFERENCES

[1] Prosthetics and Orthotics, [online]. Available at http://www.um.edu.my/ . [accessed 15 December 2013].

[2] Humanoid Robotics: *Service & Entertainment Robots*, [online]. Available at https://inlportal.inl.gov/ . [accessed 15 December 2013].

[3] M. Wisse, F. Guillaume,F. Jan Van, and B. M.oyer. “Passive-Based

Walking Robot”, IEEE Robotics & Automation Magazine. pp. 52 - 62.

June 2007.

[4] C.H. Guzman Valdivia, A. Blanco Ortega, E. Quintero Marmol Marquez and M. A. Oliver Salazar. “Development of a Biped Robot Based on Dynamic Walking”, Proceedings of the *8th International Conference on Intelligent Environments*, pp. 95-102, 2012.

[5] H. Lim, K. Tajima, T. Yoshida and K. Watanabe. “Development of Biped Walking Robot KBR-1R”, Proceedings of the *ICROS-SICE International Joint Conference*, pp. 5728-5733, 2009.

[6] S. Shirata, A. Konno and M. Uchiyama. “Design and Development of a Light-Weight Biped Humanoid Robot Saika-4”, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 148-153, 2004.

[7] Y. Ogura, H. Aikawa, H. Lim and A. Takanishi. “Development of a Human-like Walking Robot Having Two 7-DOF Legs and a 2-DOF Waist”, *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 134-139, 2004.

[8] V.T. Vaidyanathan, R. Sivaramakrishnana, “Design, Fabrication and Analysis of Bipedal Walking Robot”, *Department of Production Technology, Madras Institute of Technology, Anna University*, India.

[9] J. Yamaguchi, N. Kinoshita, A. Takanishi and I. Kato. "Development of a Dynamic Biped Walking System for Humanoid –Development of a Biped Walking Robot Adapting to the Humans' Living Floor- ", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 232-239, 1996.

[10] Robotis Product: *RobotisShop*, [online]. Available at http://www.robotis-shop-en.com/index.php . [accessed 15 December 2013].

[11] Store: *Arduino Boards*, [online]. Available at http://store.arduino.cc/ . [accessed 15 December 2013].

[12] JJ, Craig, Introduction to Robotics: *Static forces in manipulators*, pp. 155-156. Third Edition. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

[13] Chapter 5, Robot's walking: *Static Walk and Dynamic Walk*, [online]. pp. 94-96, Available at http://www.diss.fu-berlin.de/ . [accessed 11 October 2013].

[14] Biped Robot, Step3: Calculations for the Stability of the robot, [online]. Available at http://inobotics.blogspot.com/2012/09/biped-robot.html [accessed 11 October 2013.

[15] Project Biped: *Walking Cycle*, [online]. Available at http://www.projectbiped.com/ . [accessed 11 October 2013].

[16] JJ, Craig, Introduction to Robotics: *Trajectory Generation*, pp. 205-209. Third Edition. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2005.

APPENDIX A

O & P Trends & Statistics

O&P Trends & Statistics

Future Demand for Orthotic and Prosthetic Services

In the coming years, there will be a growing need for certified orthotists and prosthetists across the country. Certified professionals who enter the field will find numerous job opportunities. Those who are in the O&P profession today realize that they have found both a lifelong passion and meaningful profession.

- The total number of persons with paralysis, deformity or orthopedic impairments that use orthoses is expected to reach 7.3 million by the year 2020.
- The total number of persons with an amputation, and those using a prosthesis, is expected to reach 2.4 million by the year 2020.
- In 2007, 5,484 practitioners were certified by the American Board for Certification in Orthotics, Prosthetics & Pedorthics. Nearly 1,100 of these practitioners are 55 or older and likely to consider retirement within the next ten years.
- The goal of the Academy's national recruitment and awareness campaign is to increase the number of qualified graduates entering the profession to provide care for those who will require O&P services in the future.

Trends in Prosthetic Services

The incidence of diabetes and diabetes-related amputations continues to increase.

- 20.8 million children and adults — 7 percent of the population — have diabetes. The number of Americans with diabetes is projected to reach 29 million by 2050.
- It is estimated that approximately 1.9 million people in the US have had an amputation. More than 60 percent of non-traumatic lower-limb amputations are a result of diabetes.
- The rate of amputation for people with diabetes is 10 times higher than for people without diabetes.

Trends in Orthotic Services

- Obesity is a primary driver of orthotic service needs. There is a significant link between obesity and diabetes which creates the need for both orthotic and prosthetic services.
- The National Center for Health Statistics notes that the percent of non-hospitalized adults age 20 years and over who are overweight or obese is an astounding 66.3 percent.
- The Centers for Disease Control and Prevention notes that arthritis is on the rise in America and projects that 67 million people will be afflicted by 2030. Orthoses are frequently used to stabilize joints, reduce pain and improve function in those suffering from arthritis.
- Due to medical advances, more people are surviving strokes, but are left with long-term physical disabilities, frequently necessitating orthotic management to improve function.
- As the Baby Boomer population ages, it is at great risk of back injuries and paralysis. These individuals will have an increasing need for orthotic services.

Trends in Cost-Effective Health Care

- The ability to provide the most cost-effective and clinically appropriate O&P care will be dependent on having a large enough population of well-educated, certified orthotists and prosthetists.
- Involving certified and licensed orthotists and prosthetists in reimbursement policy decisions regarding new technological advancements will be cost-effective and enable individuals to have appropriate life-changing and life-enhancing O&P devices.

For more information on the American Academy of Orthotists and Prosthetists and details on careers in O&P, visit *www.oandp.org*.

The information outlined above is from a National Commission on Orthotic and Prosthetic Education (NCOPE) study by Caroline Nielsen, PhD, on "Issues Affecting the Future Demand for Orthotists and Prosthetists" (2002) specific to practitioners certified by the American Board for Certification in Orthotics, Prosthetics & Pedorthics (ABC), a 2006 Workforce Demand Study prepared for NCOPE and the American Orthotic and Prosthetic Association (AOPA) by Coruthers Health Consulting, LLC, and from the American Diabetes Association (2005).

American Academy of Orthotists and Prosthetists • www.oandp.org

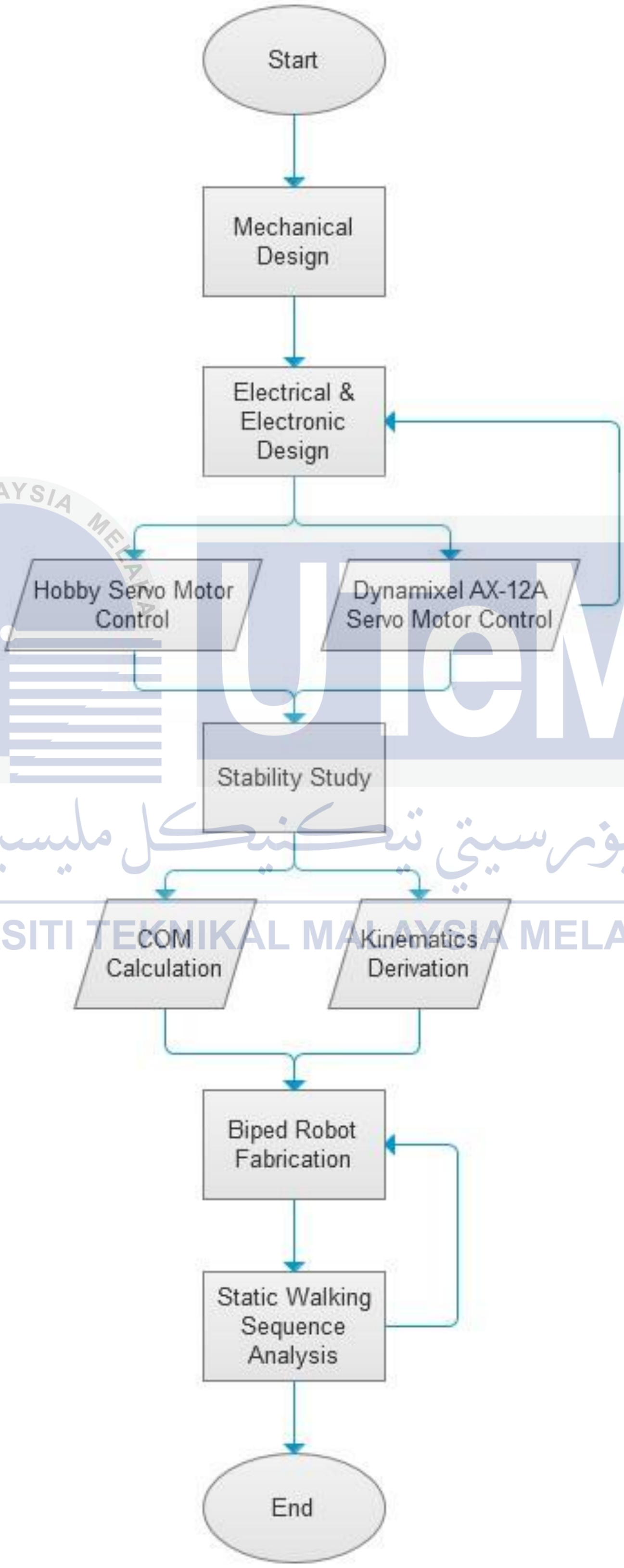
APPENDIX B

Gantt Chart

ACTIVITY	WEEK																											
	FYP1														FYP2													
WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Determining the objectives and scopes	■	■																										
Obtaining literature review			■	■	■																							
Preparing the methodology outline					■	■	■																					
Drawing the biped design using SolidWorks						■	■																					
Designing electrical & electronic configuration via ISIS						■	■																					
Data gathering from simulation and experiment						■	■																					
Progress report writing								■	■																			
Presentation										■																		
Progres report improvement & hand in											■	■	■	■														
Data gathering from simulation and experiment															■	■												
Biped robot construction																	■	■										
Identifying the kinematics of biped robot																		■	■									
calculating the center of mass for biped robot																		■	■									
Setting up Arduino algorithm for walking motion																			■	■								
Performing biped robot forward walking motion																					■	■	■					
Report writing																								■	■			
Presentation																										■		
Report hand in																											■	■

APPENDIX C

Flow Chart



APPENDIX D

Matlab Coding

```
p= [a b c d];
...a=coefficient of t^3
...b=coefficient of t^2
...c=coefficient of t
...d=constant
x= linspace(0,2);
...(0,2)=range of x-axis
y=polyval(p,x);
plot (x,y);
title('Hip Trajectory for lifting leg')
xlabel('Time')
ylabel('Angle')
```

APPENDIX E

Walking Algorithm Coding for 'medium length, medium height, medium speed'

```
#include <DynamixelSerial.h>

void setup(){
Dynamixel.begin(1000000,2);  // Inicialize the servo at 1Mbps and Pin Control 2

delay(1000);

}

void loop(){

  delay(5000); //home

  Dynamixel.moveSpeed(2,512,100);
  Dynamixel.moveSpeed(5,512,100);
  Dynamixel.moveSpeed(10,512,100);
  Dynamixel.moveSpeed(18,512,100);
  Dynamixel.moveSpeed(6,358,100);
  Dynamixel.moveSpeed(7,665,100);
  Dynamixel.moveSpeed(1,615,100);
  Dynamixel.moveSpeed(13,410,100);
  Dynamixel.moveSpeed(4,410,100);
  Dynamixel.moveSpeed(8,615,100);
  Dynamixel.moveSpeed(3,307,200);
  Dynamixel.moveSpeed(11,716,200);

  delay(1000); //weight shift

  Dynamixel.moveSpeed(2,563,50);
  Dynamixel.moveSpeed(18,563,50);
  Dynamixel.moveSpeed(10,546,50);

  delay(500); //first step

  Dynamixel.moveSpeed(8,683,100);
  Dynamixel.moveSpeed(11,784,100);

  delay(500);

  Dynamixel.moveSpeed(11,716,100);
  Dynamixel.moveSpeed(13,478,100);


  delay(500);

  Dynamixel.moveSpeed(5,512,50);
```

```
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(4,437,50);
Dynamixel.moveSpeed(1,642,50);
Dynamixel.moveSpeed(8,615,100);
Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,410,100);

delay(500); //weight shift

Dynamixel.moveSpeed(5,478,50);
Dynamixel.moveSpeed(2,461,50);
Dynamixel.moveSpeed(18,461,50);

delay(500); //second step

Dynamixel.moveSpeed(4,342,100);
Dynamixel.moveSpeed(3,239,100);

delay(500);

Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(1,547,100);

delay (500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(8,571,50);
Dynamixel.moveSpeed(13,366,50);
delay(80);
Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(4,410,100);
Dynamixel.moveSpeed(1,615,100);

delay(500); //weight shift

Dynamixel.moveSpeed(2,563,50);
Dynamixel.moveSpeed(18,563,50);
Dynamixel.moveSpeed(10,546,50);

delay(500); //third step

Dynamixel.moveSpeed(8,683,100);
Dynamixel.moveSpeed(11,784,100);
```

```
delay(500);

Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,478,100);

delay(500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(4,454,50);
Dynamixel.moveSpeed(1,659,50);
delay(80);
Dynamixel.moveSpeed(8,615,100);
Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,410,100);

delay(500); //weight shift

Dynamixel.moveSpeed(5,478,50);
Dynamixel.moveSpeed(2,461,50);
Dynamixel.moveSpeed(18,461,50);

delay(500); //fourth step

Dynamixel.moveSpeed(4,342,100);
Dynamixel.moveSpeed(3,239,100);

delay(500);

Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(1,547,100);

delay (500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(8,571,50);
Dynamixel.moveSpeed(13,366,50);
delay(80);
Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(4,410,100);
Dynamixel.moveSpeed(1,615,100);
```

```
  delay(500); //weight shift

  Dynamixel.moveSpeed(2,563,50);
  Dynamixel.moveSpeed(18,563,50);
  Dynamixel.moveSpeed(10,546,50);

  delay(500); //fifth step

  Dynamixel.moveSpeed(8,683,100);
  Dynamixel.moveSpeed(11,784,100);

  delay(500);

  Dynamixel.moveSpeed(11,716,100);
  Dynamixel.moveSpeed(13,478,100);

  delay(500);

  Dynamixel.moveSpeed(5,512,50);
  Dynamixel.moveSpeed(2,512,50);
  Dynamixel.moveSpeed(18,512,50);
  Dynamixel.moveSpeed(10,512,50);

  Dynamixel.moveSpeed(4,454,50);
  Dynamixel.moveSpeed(1,659,50);
  delay(80);
  Dynamixel.moveSpeed(8,615,100);
  Dynamixel.moveSpeed(11,716,100);
  Dynamixel.moveSpeed(13,410,100);

  delay(1000000);//stop

}
```

APPENDIX F

Walking Algorithm Coding for 'medium length, medium height, medium speed' with feedback

```
#include <DynamixelSerial.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(9, 8);

void setup(){
Dynamixel.begin(1000000,2);  // Inicialize the servo at 1Mbps and Pin Control 2
mySerial.begin(9600);

delay(1000);

}

void loop(){

  delay(5000); //home

  Dynamixel.moveSpeed(2,512,100);
  Dynamixel.moveSpeed(5,512,100);
  Dynamixel.moveSpeed(10,512,100);
  Dynamixel.moveSpeed(18,512,100);
  Dynamixel.moveSpeed(6,358,100);
  Dynamixel.moveSpeed(7,665,100);
  Dynamixel.moveSpeed(1,615,100);
  Dynamixel.moveSpeed(13,410,100);
  Dynamixel.moveSpeed(4,410,100);
  Dynamixel.moveSpeed(8,615,100);
  Dynamixel.moveSpeed(3,307,200);
  Dynamixel.moveSpeed(11,716,200);

  delay(1000);

  int var1 = Dynamixel.readPosition(1);
  int var2 = Dynamixel.readPosition(2);
  int var3 = Dynamixel.readPosition(3);
  int var4 = Dynamixel.readPosition(4);
  int var5 = Dynamixel.readPosition(5);
  int var6 = Dynamixel.readPosition(6);
  int var7 = Dynamixel.readPosition(7);
  int var8 = Dynamixel.readPosition(8);
  int var10 = Dynamixel.readPosition(10);
  int var11 = Dynamixel.readPosition(11);
  int var13 = Dynamixel.readPosition(13);
  int var18 = Dynamixel.readPosition(18);
  mySerial.print(" Number1: ");
```

```
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(1000); //weight shift

Dynamixel.moveSpeed(2,563,50);
Dynamixel.moveSpeed(18,563,50);
Dynamixel.moveSpeed(10,546,50);

delay(500); //first step

Dynamixel.moveSpeed(8,683,100);
Dynamixel.moveSpeed(11,784,100);

delay(1000);

var1 = Dynamixel.readPosition(1);
var2 = Dynamixel.readPosition(2);
var3 = Dynamixel.readPosition(3);
var4 = Dynamixel.readPosition(4);
var5 = Dynamixel.readPosition(5);
var6 = Dynamixel.readPosition(6);
var7 = Dynamixel.readPosition(7);
var8 = Dynamixel.readPosition(8);
var10 = Dynamixel.readPosition(10);
var11 = Dynamixel.readPosition(11);
var13 = Dynamixel.readPosition(13);
var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
```

```
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500);

Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,478,100);


delay(500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(4,437,50);
Dynamixel.moveSpeed(1,642,50);
Dynamixel.moveSpeed(8,615,100);
Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,410,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
```

```
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("       ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500); //weight shift

Dynamixel.moveSpeed(5,478,50);
Dynamixel.moveSpeed(2,461,50);
Dynamixel.moveSpeed(18,461,50);

delay(500); //second step

Dynamixel.moveSpeed(4,342,100);
Dynamixel.moveSpeed(3,239,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
```

```
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500);

Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(1,547,100);

delay (500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(8,571,50);
Dynamixel.moveSpeed(13,366,50);
delay(80);
```

```
Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(4,410,100);
Dynamixel.moveSpeed(1,615,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500); //weight shift

Dynamixel.moveSpeed(2,563,50);
Dynamixel.moveSpeed(18,563,50);
Dynamixel.moveSpeed(10,546,50);
```

```
delay(500); //third step

Dynamixel.moveSpeed(8,683,100);
Dynamixel.moveSpeed(11,784,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("       ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500);

Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,478,100);
```

```
delay(500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(4,454,50);
Dynamixel.moveSpeed(1,659,50);
delay(80);
Dynamixel.moveSpeed(8,615,100);
Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,410,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
```

```
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500); //weight shift

Dynamixel.moveSpeed(5,478,50);
Dynamixel.moveSpeed(2,461,50);
Dynamixel.moveSpeed(18,461,50);

delay(500); //fourth step

Dynamixel.moveSpeed(4,342,100);
Dynamixel.moveSpeed(3,239,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("     ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
```

```
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500);

Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(1,547,100);

delay (500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(8,571,50);
Dynamixel.moveSpeed(13,366,50);
delay(80);
Dynamixel.moveSpeed(3,307,100);
Dynamixel.moveSpeed(4,410,100);
Dynamixel.moveSpeed(1,615,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
mySerial.print(" Number6: ");
```

```
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500); //weight shift

Dynamixel.moveSpeed(2,563,50);
Dynamixel.moveSpeed(18,563,50);
Dynamixel.moveSpeed(10,546,50);

delay(500); //fifth step

Dynamixel.moveSpeed(8,683,100);
Dynamixel.moveSpeed(11,784,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
mySerial.println(var1);
mySerial.print(" Number2: ");
mySerial.println(var2);
mySerial.print(" Number3: ");
mySerial.println(var3);
mySerial.print(" Number4: ");
mySerial.println(var4);
mySerial.print(" Number5: ");
mySerial.println(var5);
```

```
mySerial.print(" Number6: ");
mySerial.println(var6);
mySerial.print(" Number7: ");
mySerial.println(var7);
mySerial.print(" Number8: ");
mySerial.println(var8);
mySerial.print(" Number10: ");
mySerial.println(var10);
mySerial.print(" Number11: ");
mySerial.println(var11);
mySerial.print(" Number13: ");
mySerial.println(var13);
mySerial.print(" Number18: ");
mySerial.println(var18);

delay(500);

Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,478,100);

delay(500);

Dynamixel.moveSpeed(5,512,50);
Dynamixel.moveSpeed(2,512,50);
Dynamixel.moveSpeed(18,512,50);
Dynamixel.moveSpeed(10,512,50);

Dynamixel.moveSpeed(4,454,50);
Dynamixel.moveSpeed(1,659,50);
delay(80);
Dynamixel.moveSpeed(8,615,100);
Dynamixel.moveSpeed(11,716,100);
Dynamixel.moveSpeed(13,410,100);

delay(1000);

 var1 = Dynamixel.readPosition(1);
 var2 = Dynamixel.readPosition(2);
 var3 = Dynamixel.readPosition(3);
 var4 = Dynamixel.readPosition(4);
 var5 = Dynamixel.readPosition(5);
 var6 = Dynamixel.readPosition(6);
 var7 = Dynamixel.readPosition(7);
 var8 = Dynamixel.readPosition(8);
 var10 = Dynamixel.readPosition(10);
 var11 = Dynamixel.readPosition(11);
 var13 = Dynamixel.readPosition(13);
 var18 = Dynamixel.readPosition(18);
mySerial.println("        ");
mySerial.print(" Number1: ");
```

```
  mySerial.println(var1);
  mySerial.print(" Number2: ");
  mySerial.println(var2);
  mySerial.print(" Number3: ");
  mySerial.println(var3);
  mySerial.print(" Number4: ");
  mySerial.println(var4);
  mySerial.print(" Number5: ");
  mySerial.println(var5);
  mySerial.print(" Number6: ");
  mySerial.println(var6);
  mySerial.print(" Number7: ");
  mySerial.println(var7);
  mySerial.print(" Number8: ");
  mySerial.println(var8);
  mySerial.print(" Number10: ");
  mySerial.println(var10);
  mySerial.print(" Number11: ");
  mySerial.println(var11);
  mySerial.print(" Number13: ");
  mySerial.println(var13);
  mySerial.print(" Number18: ");
  mySerial.println(var18);

  delay(1000000);//stop

}
```

APPENDIX G

Instruction Manual to Control Biped Robot using Arduino in Windows

1) Get an Arduino board and USB cable

Use Arduino board such as Arduino Uno, Arduino Mega 2560 and so on. You also need a standard USB cable (A plug to B plug): the kind you would connect to a USB printer, for example.

2) Download the Arduino environment

Get the latest version of Arduino from arduino.cc page. When the download finishes, unzip the downloaded file and install the Arduino software into the window.

3) Connect the board

The Arduino UNO board automatically draw power from either the USB connection to the computer or an external power supply. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between USB and power jacks. Connect the Arduino board to the computer using USB cable. The green power LED (labelled PWR) should go on.

4) Install the drivers

Install the drivers for Arduino Uno or Arduino Mega 2560 with Windows7, Vista or XP.

- Plug in your board and wait for Windows to begin it's driver installation process. After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)"
- Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno's driver file named "Arduino UNO.inf"
- Windows will finish up the driver installation from there.

5) Launch the Arduino application

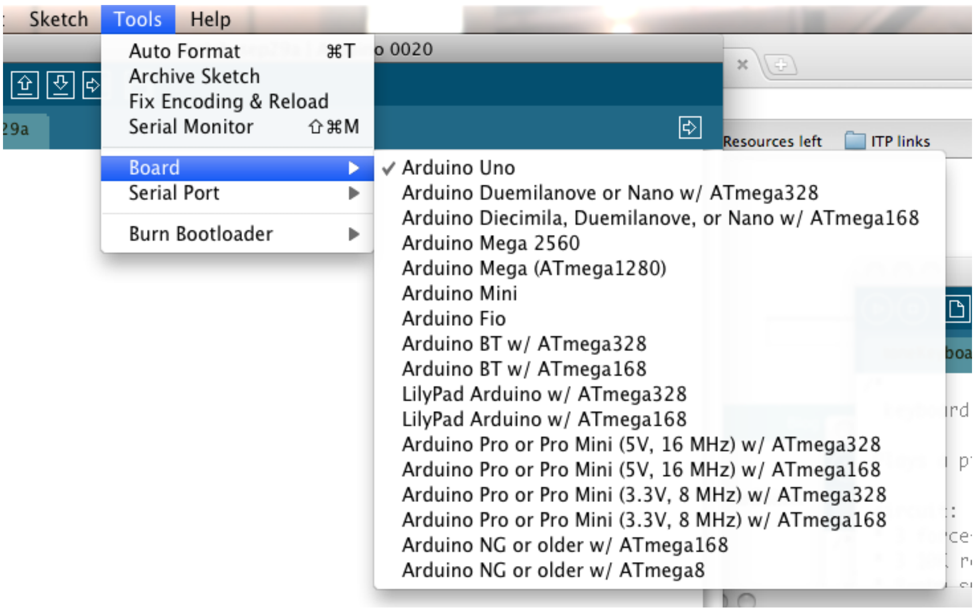
Double-click the Arduino application. (Note: if the Arduino software loads in the wrong language, you can change it in the preferences dialog.)

6) Open the source code

Open the source code file from your computer: File > Open > small_steps_method_1_turn_180

7) Select your board

You'll need to select the entry in the **Tools** > **Board** menu that corresponds to your Arduino.

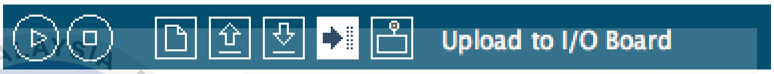


8) Select your serial port

Select the serial device of the Arduino board from the **Tools** > **Serial Port** menu. This is likely to be **COM3** or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappear should be the Arduino board. Reconnect the board and select that serial port.

9) Upload the program

Now, simply click the "**Upload**" button in the environment. Wait for a few seconds and you should see the **RX** and **TX** leds on the board flashing. If the upload is successful, the message "**Done uploading.**" will appear in the status bar.



A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino up-and-running.

10) Select the serial port for USB to Serial converter (Optional)

In order the retrieve the reading from the Arduino board back to the computer, you must first connect the USB-to-Serial Converter to the Arduino board. Then you can select the other port by going to **Tools** > **Serial Port** menu.

11) Open serial monitor (Optional)

Open the serial monitor window by going to **Tools** > **Serial Monitor** or pressing the shortcut key **Ctrl+Shift+M**.