DESIGN & IMPLEMENTATION OF A TWO-LEGGED HUMANOID ROBOT

By

SAIFUL BIN A'EE

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Program
In Partial Fulfillment of the Requirements
For the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

DESIGN & IMPLEMENTATION
OF A TWO-LEGGED HUMANOID ROBOT

by

Saiful Bin A'ee

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

[illegible]

Mohd Haris Bin Md Khir
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own, except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

Saiful Bin A'ee

ABSTRACT

Nowadays, the development of robotic field is developing rapidly. The effort in developing a robot that can act and think like human has been done by various parties including the institutions of higher learning and the private company. This paper presents the design and implementation of a two-legged humanoid robot that capable of walking forward and backward. The robot is having a total of five degree of freedom (DOF), which comprises of two DOF on each knee, two DOF on each pelvis and one DOF used as balancing mechanism. These DOF is implemented using servomotors and are controlled using microchip PIC16F877 and PIC16F84A. The most critical part in designing this robot is to achieve its stability especially when it begins to walk. The stability of the structure is solves using the counterweight mechanism. The development of this biped is done stage by stage through developing and modifying the structure, constructing the circuit, programming the controller and combining both the hardware and software part. The results that have been achieved are the stable and rigid structure and the walking motion and it will be discussed in detail in the result part of this report.

ACKNOWLEDGEMENTS

In the name of Allah the Beneficent, the Merciful

First of all thank you to my supervisor; Mr. Mohd Haris bin Md Khir for giving me the chance to take his FYP topic. His assistance in completing this project up to this stage is very essential. Thank you also to Mr Mohd Zuki Yusoff and Mr Patrick Sebastian, my Microcontroller lecturers for their assistant in the programming sections. Also to all the FYP coordinator, Ms Azrina Abd Aziz, Ms. Nasreen Badrudin and Ms Illani, your assistance will not be forgotten.

To my family, my mother, my father, my aunt and uncle and to my sister and brother, thanks a lot for giving me a lot of useful advice and guidance throughout this project. Your effort in supporting me is very valuable.

Special thank to Ms. Siti Hawa for always being supportive in giving numerous technical support as an FYP Lab coordinator. Thanks also to Mr Isnani, the coordinator for EE department workshop for his assistant in building the structure. Not forgetting, to all EE department lab technicians whose names might not be included here but contributed also to the success of this project.

Last but not least, to my fellow colleagues, Mr. Mohd Shafiq, Ms Ooh Lay Shan and to those who have contributed directly or indirectly toward the completion of this project, thanks to all of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

UTM	- Universiti Teknologi Malaysia
R&D	- Research and Development
UTP	- Universiti Teknologi PETRONAS
EE	- Electrical & Electronic Engineering
FYP I	- Final Year Project I
FYP II	- Final Year Project II
PIC	– Programmable Intelligent Controller
DOF	- Degree of Freedom
COM	- Center of Mass
COG	- Center of Gravity
I/O	- Input/Output
PWM	- Pulse Width Modulation
PCB	- Printed Circuit Board
DC	- Direct Current

CHAPTER 1
INTRODUCTION

1.1 Project Background

According to the 6th edition of the Oxford Advanced Learner's Dictionary, a robot is defined as a machine that can do some tasks that a human can do and that works automatically or is controlled by a computer. In general, robots deal a lot with control system. From the control system point of view, robots can be viewed as a plant which takes input (desired task or set point) from the user, does some computing in its controller and produces the output to the final element, usually a servomotor or a hydraulic jack. In some cases, the output of the final element will be fed back to the controller and will be compared with the input to determine whether the desired input has been achieved or not.

The most popular type of robot is the two-legged humanoid robot also known as the biped. Nowadays, researchers throughout the globe are trying to develop biped that can act and even think like humans. Figure 1 illustrates the example of bipedal robot.

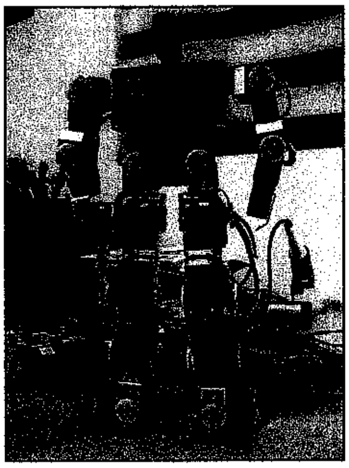


Figure 1 An Example of Bipedal Robot[7]

Looking back, the idea of humanoid robots had begun back in the year 1921 when Czech playwright, Karel Capek (1890-1938) published his best-known work, the R.U.R (Rossum's Universal Robot)[4].

In Malaysia, this field is still new compared to developed nation like U.S, Europe and Japan. The research that involves robotics, especially humanoid robots has only been performed by institution of higher learning. The foremost university in this field, Universiti Teknologi Malaysia (UTM) has performed a lot of research and development (R&D) for the pass several years.

In Universiti Teknologi PETRONAS (UTP), the project on biped robot had been initiated several semesters ago, under the Electrical and Electronic Engineering (EE) Department. However, the first trial was not successful because of the wrong material used. The material chosen is too heavy and most of the servo-motor used had broken down. Since that first trial, the project on the biped robot has been put on hold and the department focused on the development of the autonomous robot. This is because the autonomous robot is easier to develop compared to the biped robot because it does not need a balancing mechanism.

1.2 Problem Definition

The main intention of this project is to design and implemented two-legged humanoid robot that is capable of walking forward and backward. Several problems on this project have been defined to simplify the development process. The problems are:

- Material selection
- Mechanical design and fabrication for the legs
- Software programming and electronic circuit design
- Design the balancing mechanism of the robot

1.2.1 Materials Selection

The most suitable materials to fabricate the structure is those that is light and have strength. This is very important to minimize the overall weight of the structure. Otherwise, the servomotor will not be able to pull up the leg and to perform the desired turning degree. Among the materials that can be considered to fabricate the structure are aluminum, perspex, plastic polymer and carbon fiber.

In choosing the fabrication materials, the aspect of availability of the materials, the overall cost and the flexibility to be shaped, should also be taken into consideration. Thus among the four materials considered, the aluminum alloy is the most ideal material to be chosen as fabrication material.

1.2.2 Mechanical Design and Fabrication for the Legs

The mechanical legs form the main part of the biped robot. The main problem in this part is on how to attach the servomotor onto the leg to achieve the desired turning degree level. Besides, the way on how to combine the leg component should also be considered in order to achieve the intended degree of freedom.

1.2.3 Software Programming and Electronics Circuit Design

The electronic part is used to control the movement of the leg component. For this purpose, the PIC microchip or micro-controller is used. The micro-controller needs to have its own electronic circuit and needs to be programmed to enable it to control the leg movement.

1.2.4 Balancing Mechanism

For this beginning level of design and fabrication, the need to design one specific balancing mechanism for the robot is necessary. Looking at the human body system, the stability mechanism is the ear which works together with the brain. This concept will be applied to the design. This balancing mechanism is necessary because once the biped starts to pull up one of its legs (to begin walking), it tends to collapse, thus needing a mechanism to balance it.

1.3 Objectives

The objectives of this project can be divided into four parts as follows:

1.3.1 To Design and Implement a Two-legged Humanoid Robot

This is the main objective of this project. This main objective is further divided into two which are:

- Structure design and selection of material (FYP I).
- Hardware and software implementation (FYP II).

1.3.2 To Design a Stable, Strong and Light Leg Structure

In a biped robot, the stability of the structure is the most critical issue. As mentioned earlier in the problem definition, as one of the legs is pulled upwards, the standing leg tends to lose its balance and this makes the robot collapse, thus leading it to the balancing mechanism as a solution. In addition to that, the leg needs to be as light as possible. The material for the leg structure also needs to be strong and rigid. One possible material is the L-shape aluminum.

1.3.3 To Produce a Smooth Walking Movement of the Structure

This is the objective of the second phase of the project, the FYP II. Upon achieving the stable, strong and light structure, the next step is to make the structure walk smoothly. This can be achieved through programming the controller of the structure and planning for the best walking algorithm. The hardware (controllers) that are used to control the robot leg are the PIC16F877 and PIC16F84A and the software that is used to program this PIC is the C-language.

1.4 The Relevancy of the Project

Currently, a lot of effort has been put to develop the humanoid robot that is capable of copying human motion. In developed nations such as U.S, Europe, Japan and Korea, not only the institution of higher learning but various big companies have developed their own walking robot. The development of ASIMO by Honda Motor Corporation and the SDR series by SONY Corporation for example, are the indication that private companies are also looking very seriously into developing the humanoid robot. Figure 2 below illustrates the humanoid robot design by both companies.

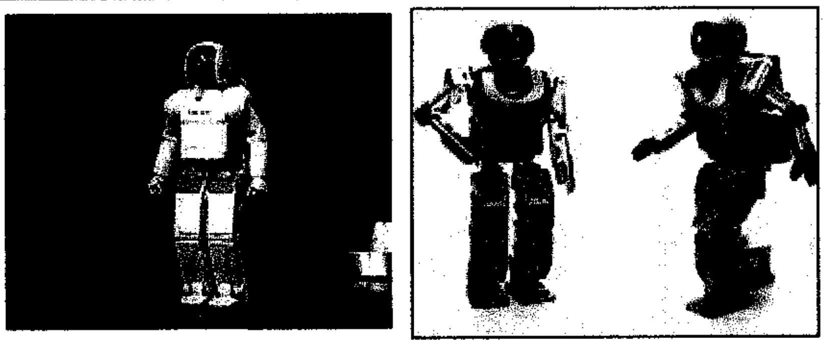


Figure 2 : Example of Commercial Humanoid Robot (Courtesy of Honda and Sony)

In parallel with the university's vision, which is to be the leader in technology and education and the center for creativity and innovation, this project is very ideal to develop. In addition to that, the university's main competitor, UTM has successfully developed their own humanoid robot.

1.5 Time Frame and Project Planning

Time frame and project planning is very important in order to achieve the goals for this project within the time period. The project will be performed in two semesters and therefore, the time frame will be separated into two periods. The first semester goal is to fabricate the hardware and achieve some simple motion, such as pulling on leg upward. While during the second semester, the focus is more on the circuit, the software and optimizing the design. Towards the end of the second semester, the biped prototype should be ready and can perform the desired motion, which is walking forward and backwards. The project timeline for both semesters (FYP I and FYP II) is included in Appendix A and Appendix B respectively.

CHAPTER 2
LITERATURE REVIEW & THEORY

2.1 Literature Review

Vast numbers of literature reviews have been done on the topic to gather information especially on how to fabricate the leg component. The sources of information are the internet, books on robotics and previous year thesis. From these various sources the idea is generated and some of the problems are solved.

2.1.1 Internet

A lot of web pages in the internet provide information on the biped project. Such web page are the android world. Androidworld is a web page for individuals who are involved in the development of the humanoid to provide or share their design to the world. Figure 3 illustrates one of the interesting biped projects. It was done by Alexander Vogler from Vienna Austria. The biped is named V-3 and its total weight is only 1 kg with maximum height of 30 cm. The V-3 has a total of 12 degrees of freedom.

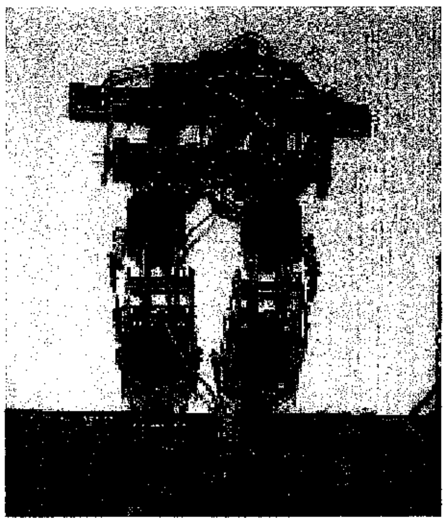


Figure 3 : Biped Project Example[7]

There are also a Malaysian sharing his biped project in this web page. He is a student of UTM. He conducted a biped project as his final year project. Figure 4 below illustrates the UTM's student bipedal.

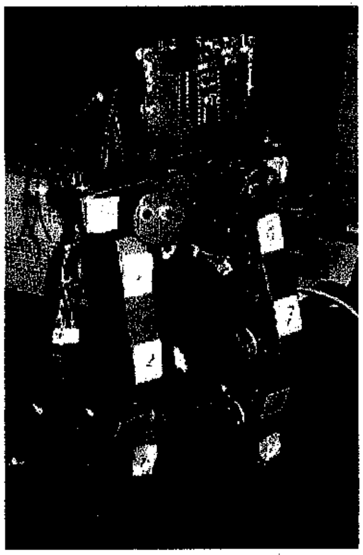


Figure 4 : UTM Student Biped Structure[7]

2.1.2 Project Paper on Biped Walking Algorithm

Walking algorithm is one of the most important issues in biped design. The walking algorithm for the biped can be found easily from the internet from previous and current projects on humanoid robot.

One of the best examples is a project done by a group of academicians from the "Universitat de Girona" of Girona. Figure 5 below illustrates their biped.

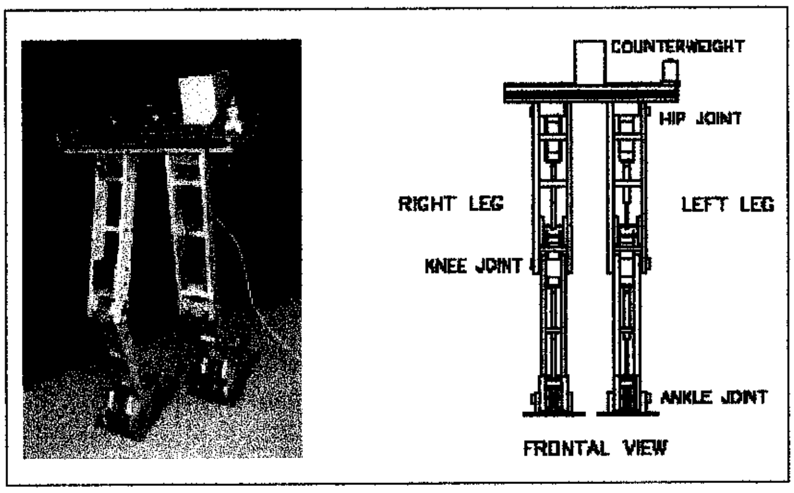


Figure 5 : Project on Walking Biped[5]

From the figure, it is clear that the biped is using 7 degrees of freedom including the top part. The structure also implemented a balancing mechanism on top of it to balance the structure especially when making movement. Together with this report is the walking algorithm for their structure. Figure 6 illustrates the walking algorithm for this structure.

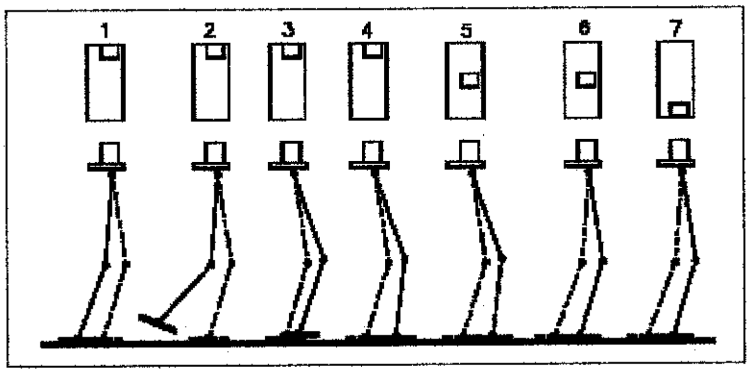


Figure 6 : Walking Algorithm for 7 DOF Biped[5]

The walking algorithm as shown in the figure 6 above includes the movement of what the called the counterweight as depicted above for each of the walking movement. This counterweight or balancing mechanism is very important to project the center of gravity of the structure across the foot base.

2.2 Theory

In designing and implementing the biped, there are a number of useful mechanical theories involved. They are:

- Orientation angle
- Mass supporting area
- Center of mass (COM)

2.2.1 Center of Mass (COM)

The COM or center of gravity (COG) is defined by the gravity of the body. The total gravity of the body is equal to the sum of the gravity of its mass-element. The total gravity of the body is equivalent to the center of mass (COM).

2.2.2 Orientation Angle

In common, the aircraft orientation angles can be expressed in three forms, which are the pitch, rolls and yaw. This orientation angles are used in the aviation and navigation of the aircraft and their clearness makes them feasible for biped also. In fact, the human body system also makes use of these three orientation angles in every moving joins. Figure 7 illustrates the definition of these angles.

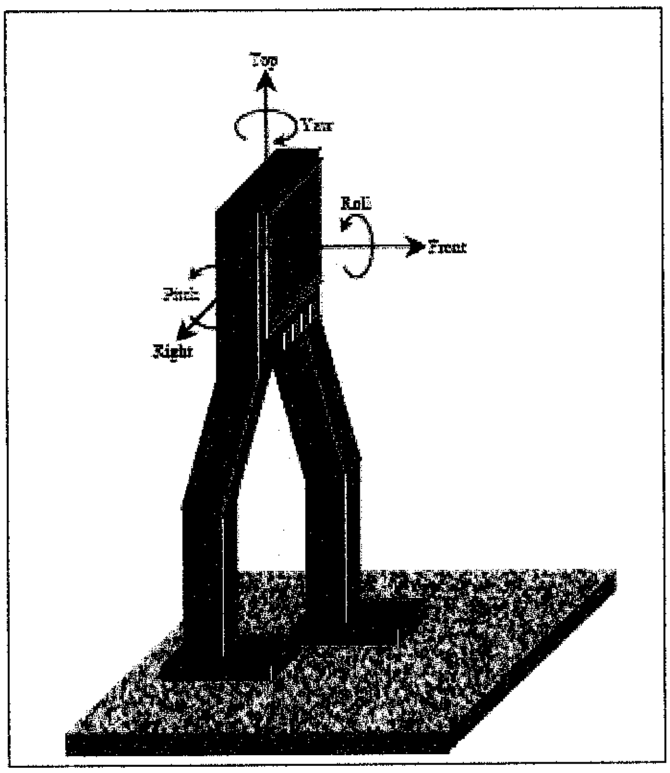


Figure 7 : Orientation Angles[4]

For the deflection of the center of the mass to the front and to the back, the orientation angle is the pitch. The roll angle is for the body to move either to the left or to the right. The function of pitch angle is for the forward-backward movement. On the other hand, the yaw is used to turn the body around. This functions when the whole body wanted to make turning movement. Finally, combining all the orientation angles will produce a smooth walking movement.

2.2.3 Mass Supporting Area

The mass supporting area is defined as the area surrounded by the corners of the feet. This area is for the purpose of stability. Figure 8 illustrated the definition of the supported area.

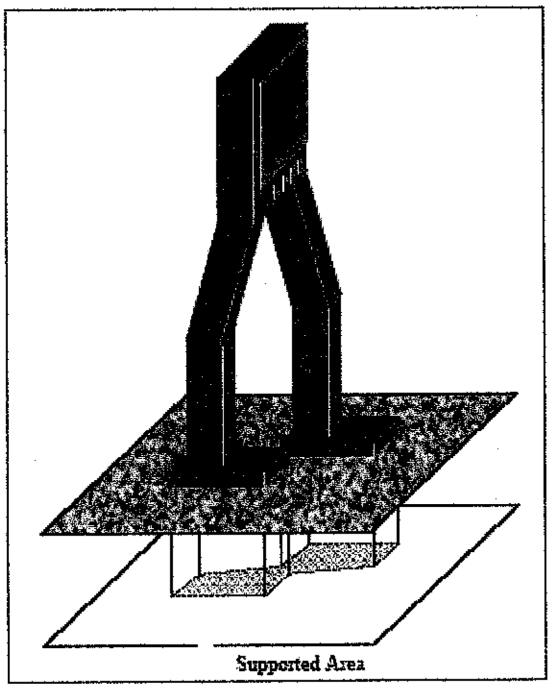


Figure 8 : Supporting Area[4]

CHAPTER 3
METHODOLOGY

3.1 Procedures

The procedure to complete the project is as depicted in the figure 9. The procedure has been planned properly in the early stage of this project to avoid redundant steps and to optimize the time allocated.

The whole design and implementation procedures are subdivided into two parts, each for first and second semester. The first semester is on the mechanical part, while the second semester is on the electronics and programming part.

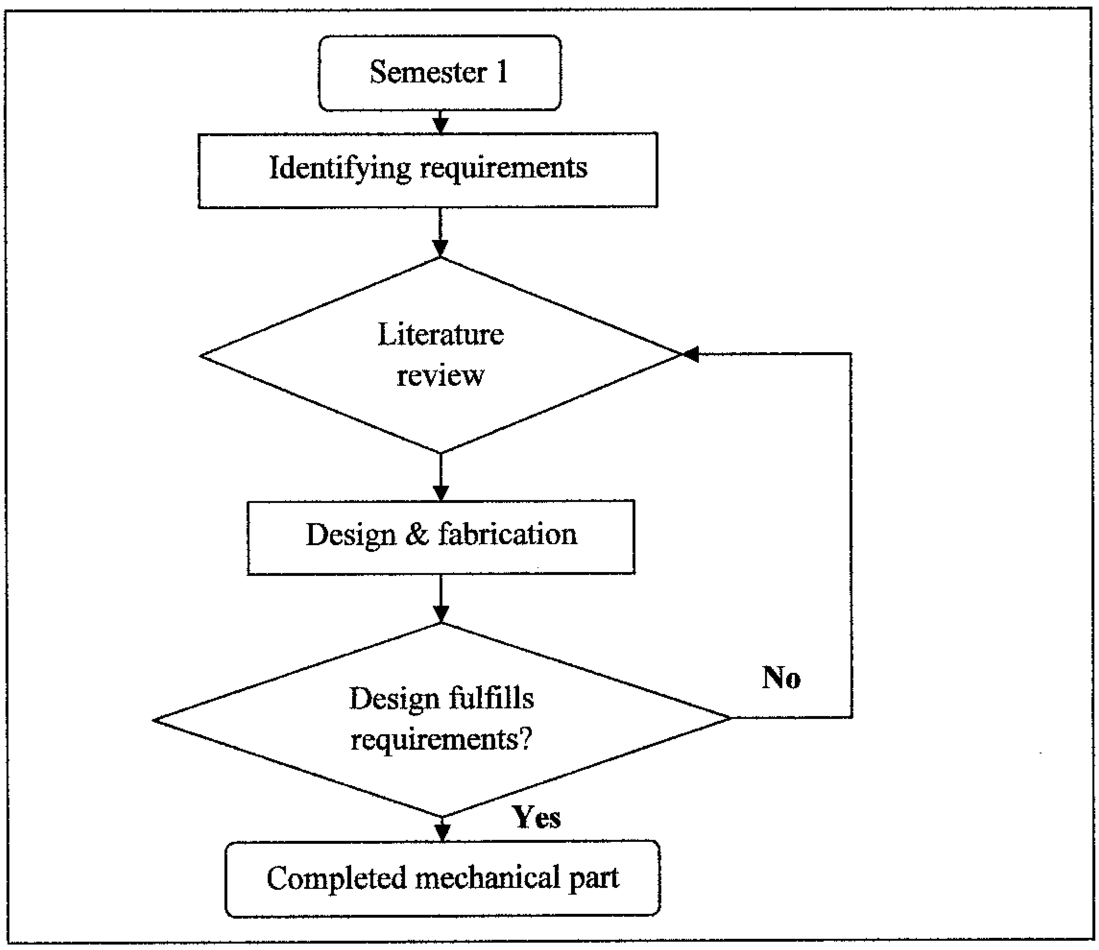


Figure 9 : Project Methodology for Semester I

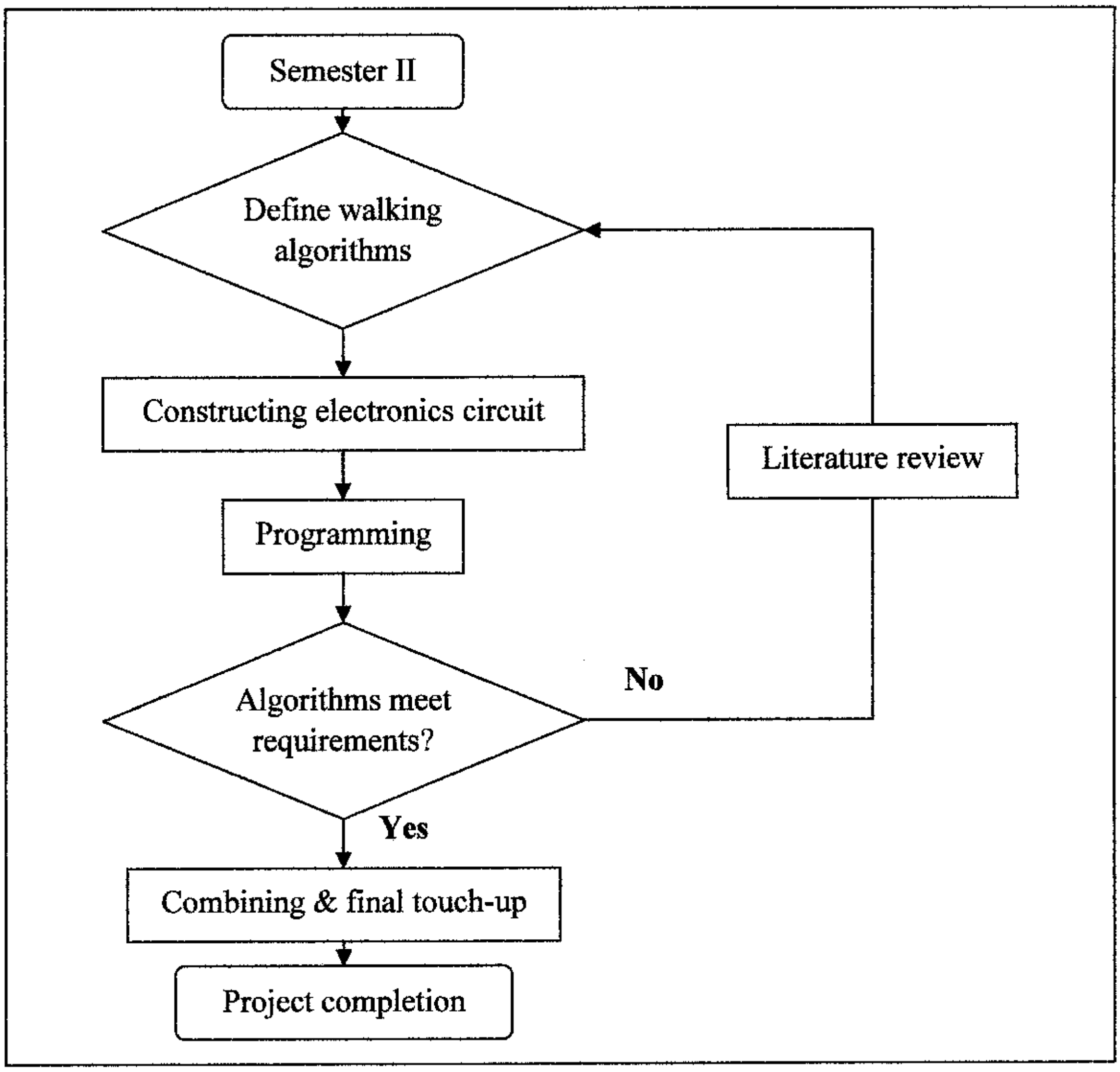

Semester II
Define walking algorithms
Constructing electronics circuit
Literature review
Programming
No
Algorithms meet requirements?
Yes
Combining & final touch-up
Project completion

Figure 10 : Project Methodology for Semester II

3.1.1 Identifying Requirements

The need to identify all the design requirements is to ensure the rest of the design process runs smoothly. In the case of the biped robot, the very important design requirements are the balancing and the fabrication materials.

3.1.2 Literature Review

Literature review is where all the ideas of the robot structure are generated. Literature review is done through internet, journal, books, thesis and previous project (FYP). All possible ideas are gathered and evaluated to choose the best design for fabricating purposes.

3.1.3 Design & Fabrication

The best idea is put into drawing. The drawing is important for documentation purposes and to simplify the fabrication. The design will be modified from time to time for improvement purposes.

3.1.4 Design Evaluation

The evaluation stage is to review the design whether it has met the requirement or not. If the design has not meet the requirement especially in terms of stability, then the redesign step is done and the structure will be modified. If necessary, the redesigning step will also involve literature review until the requirement is fulfilled. When the requirements have been fulfilled, the mechanical part is completed.

3.1.5 Defined Walking Algorithm

The second part of the project methodology is begun with defining the desired walking algorithm. The walking algorithm is for the purpose of programming the controller. The algorithm is obtained from literature review.

3.1.6 Constructing Electronics Circuit

When the algorithm has been finalized, the controller circuit is constructed. The circuit will be first constructed on the bread board before it is finalized on the printed circuit board (PCB). The PCB is design using software and fabricated in the department's PCB design laboratory (PCB lab).

3.1.7 Programming

The programming part is to control the servomotors using the PIC microchip to achieve the desired walking algorithm. The programming is performed using C-language and it is modified from time to time for optimizing purpose.

3.1.8 Walking Algorithm Evaluation

The initially planned walking algorithm needed evaluation to see whether it was working or otherwise. If the algorithm is not working, then the new algorithm needed to be developed. This is done through literature review. This process will continue until the most suitable algorithm fitted the structure design.

3.1.9 Combining & Final Touch-up

When everything has been completed (both the mechanical and electrical part), they are combined together to form one well-functioning biped that can walk forward and backward.

3.2 Tools & Software

The list of tools and software to complete the project can be separated into two parts, the mechanical and electronics. Table 1 and 2 describes briefly all the tools and software used in this project.

Table 1 : List of Tools for Designing and Fabricating

Mechanical Tools	*Electronics Tools*
L-shape aluminum	PIC16F877
Screws	PIC16F84A
Nuts	Resistors
Perspex	Crystal clock oscillator
Metal glue	Toggle switch
Drill	Servomotor
Saw	Bread board
Grinder	PCB board
Screw driver	Programmer board
	Soldering equipments
	Multimeter
	Oscilloscope

Table 2 : List of Software for Designing and Programming

Electronics Software
PIC C-compiler
WARP 13
Eagle

3.2.1 Tools

The mechanical and electronics tools are very common and frequently used. The only uncommon tools in the list are the programmer board and the target board.

The programmer board is an electronics board used to upload the program into the microcontroller. After designing the program in C-language, the C file will be converted into ".hex" file to be uploaded using this programmer board into the microcontroller.

3.2.2 Software

For programming purposes, the software used are PIC C-compiler and WARP 13. The PIC C-compiler is the programming software and the one that is used to produce the ".hex" file. After that, the ".hex" file will be uploaded into the microcontroller using the WARP 13 software. For the design of the printed circuit board (PCB), Eagle is used. This software produced "gerber" file that is used to print the electronic circuit.

CHAPTER 4
RESULTS & DISCUSSION

The designing of the biped robot is separated into two parts which are the mechanical fabrication and the electronics and programming part. The mechanical fabrication involves modifying the structure to obtain an optimized design while the electronics part involved doing the programming in various stages and designing the electronics circuit.

4.1 Mechanical Fabrication of the Structure

The mechanical fabrication was performed stage by stage with each step in the modification of the previous structure. The fabrication began with the very first rough sketch until the final optimized structure.

4.1.1 Rough Sketch

The first rough sketch took into account the total degree of freedom that should be implemented on the structure. The degree of freedom is 10 in total which is very similar to a human leg in general. Those 10 degrees of freedom comprises 5 degrees of freedom being implemented in each leg as illustrated in the following figure. Three degrees of freedom on pelvis-thigh joins, one on the knee and one on the ankle. Figure 11 in the following page illustrates the rough design of the structure.

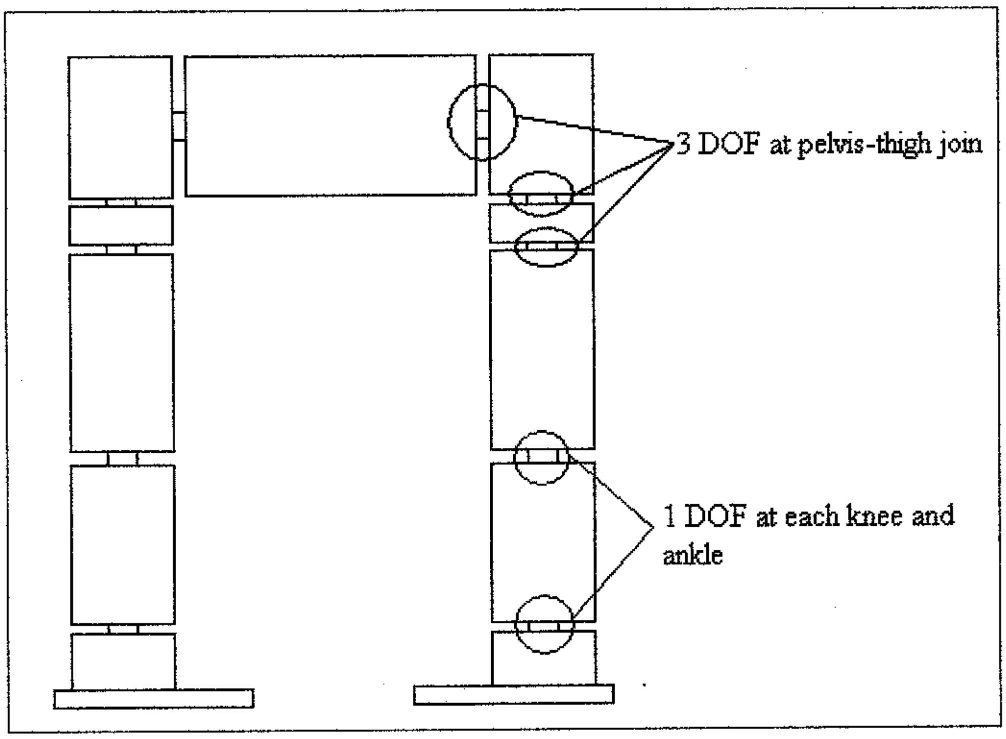


Figure 11 : Rough Idea of the Structure

With this 10 degrees of freedom the biped seem to be copying each human muscle and is capable of walking like a human. However, due to limited number of servos available and due to the complexity of this 10 degrees of freedom, only 5 degrees of freedom is implemented on the structure. This will need only five servos to be used, two servos in each leg, one as a knee and the other one as the pelvis-thigh join. The fifth servo will be used as the balancing mechanism.

4.1.2 First Structure Design

Figure 12 illustrates the first structure of this biped. This first structure seems to be unstable and rigid due to its height and its overall weight. The total height of this structure is 38 cm when both legs are in straight position. Its overall width measured from left foot to right foot is 23 cm. Due to these two factors, the structure tends to collapse even when it is in an equilibrium position.

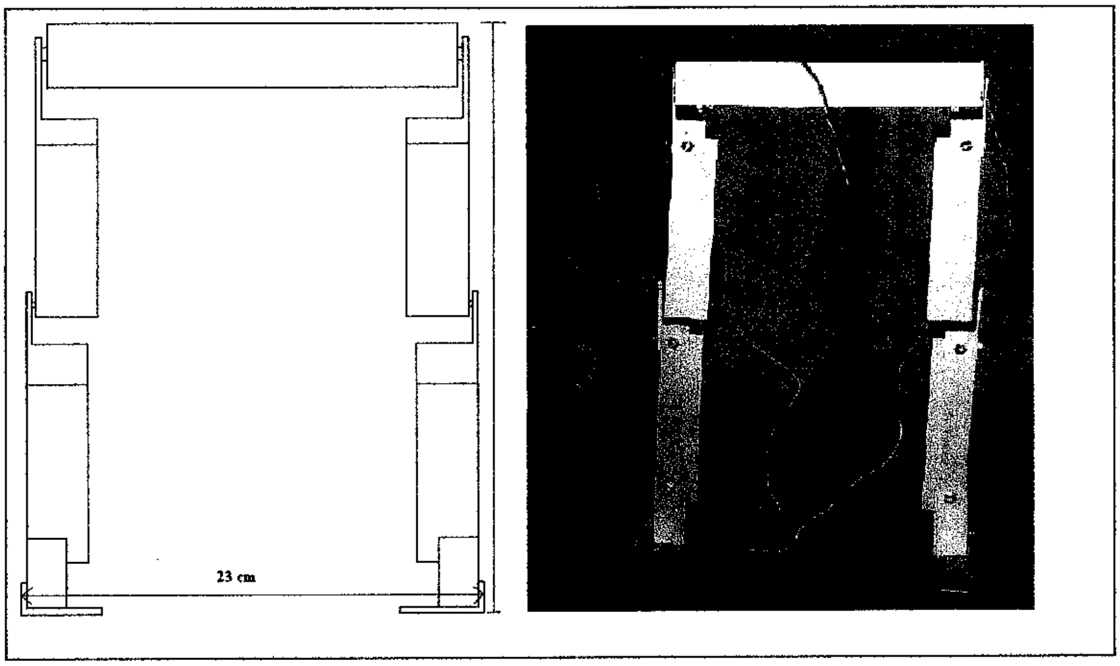


Figure 12 : First Structure Design

4.1.3 Second Structure

The unstable structure needs to be redesigned to make it become more stable. This leads to the second design, which seems to be more rigid and stable and in addition, this second design is fabricating with balancing mechanism at the back of its pelvis. In this design, the height is reduced to 20 cm and its pelvis is also reduced to 13 cm x 7cm. The following figure 13 illustrates this second design.

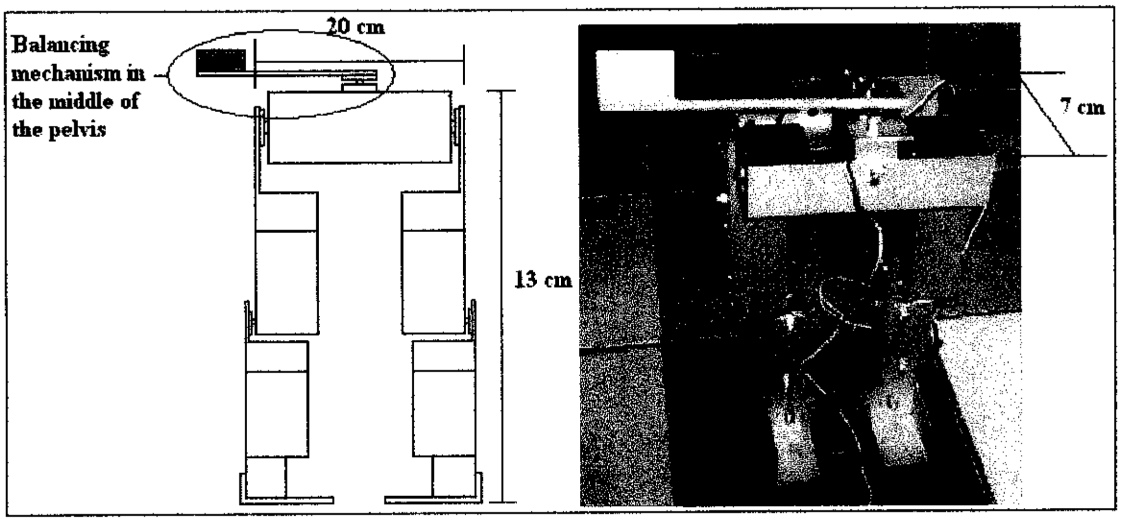


Figure 13 : Second Structure Design

4.1.4 Problem in the Second Design

The second design also seems to be unstable when it is tested to stand with one leg. The structure leg tends to bend to the other side when it is let to stand with only one leg. One experiment has been conducted to determine the strength of the leg when it stands with one leg. This is to simulate the condition of the structure when it pulls one of its legs up. The result is the current structure cannot sustain the load. The following figure 14 illustrated the forces acting on the leg that may cause the structure to bend and cannot stand straight.

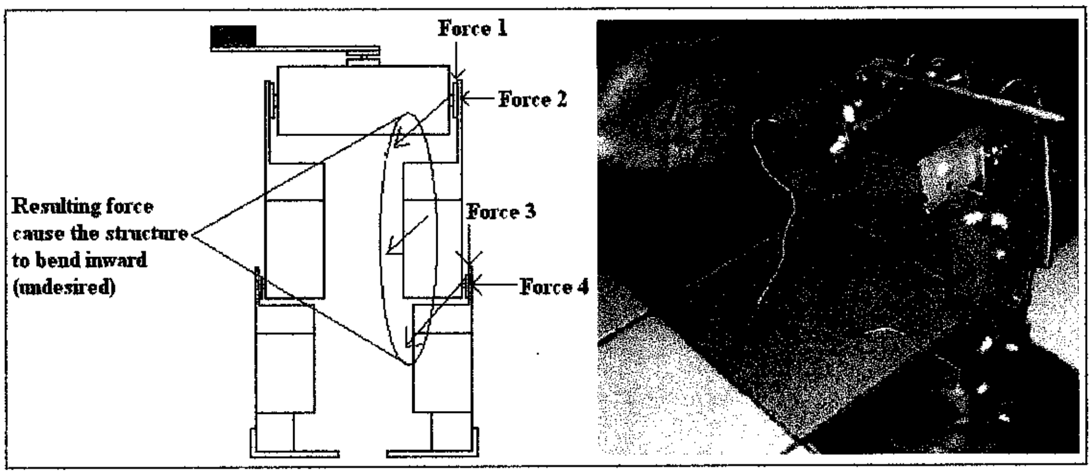

Force 1
Force 2
Force 3
Force 4
Resulting force cause the structure to bend inward (undesired)

Figure 14 : Unbalance Force Acting on the Structure

4.1.5 Third Structure Design

The unbalanced forces acting on the structure need the structure to be redesigned. The solution to this is changing the position of the 'knee-servo' to face inwards. As illustrated in figure 15 on the following page, the unbalanced force is solved by changing the position of the knee servomotor. Besides, the length of the rod to place the load is also reduced. In this third design, the area of the pelvis is further reduced to 11 cm x 7 cm. both of this modification is to optimize the stability of the structure.

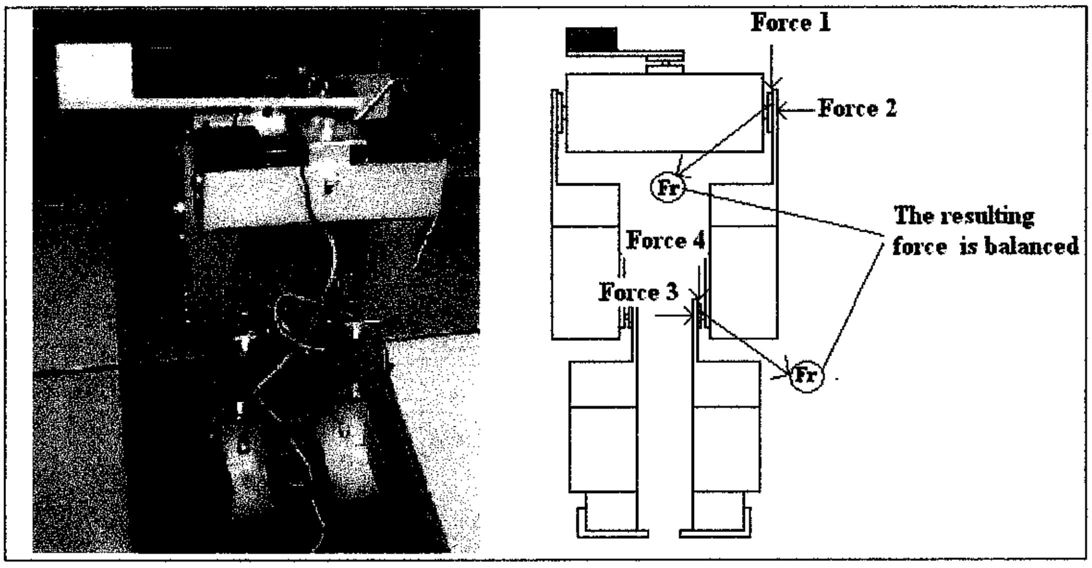


Figure 15 : Third Structure Design Balanced the Force

4.1.6 Final Structure Design

The fourth structure design is further improved to optimize its stability. The improvements are:

- The mechanism of counter weight
- The foot
- The pelvis-thigh join

On the counterweight modification, the sliding path is designed on top of the balancing servo. This path enables the balancing weight to slide from left to right. This method of transferring the balancing weight is better than the previous one in which the balancing weight is attached directly to the servomotor. With this method, the stability of the structure is improved. The movement of the weight from left to the right is performed by one servomotor. The servomotor moves the counterweight by pulling the string attached to the counterweight. This mechanism also includes a pulley on each side of the pelvis and a small displacement disc used to slide the weight to the left and right when the servomotor rotates.

On the other hand, the mass supporting area (foot) is enlarged. On each foot, one rectangular perspex is attached to it. This modification also improves the stability of the structure through enlarging the supporting base.

The existing pelvis-thigh join is not strong and not capable of supporting the pelvis weight. When standing with one leg, this join tends to bend. Thus, modification is made on this structure to improve its strength. Figure 16 illustrates the final biped structure.

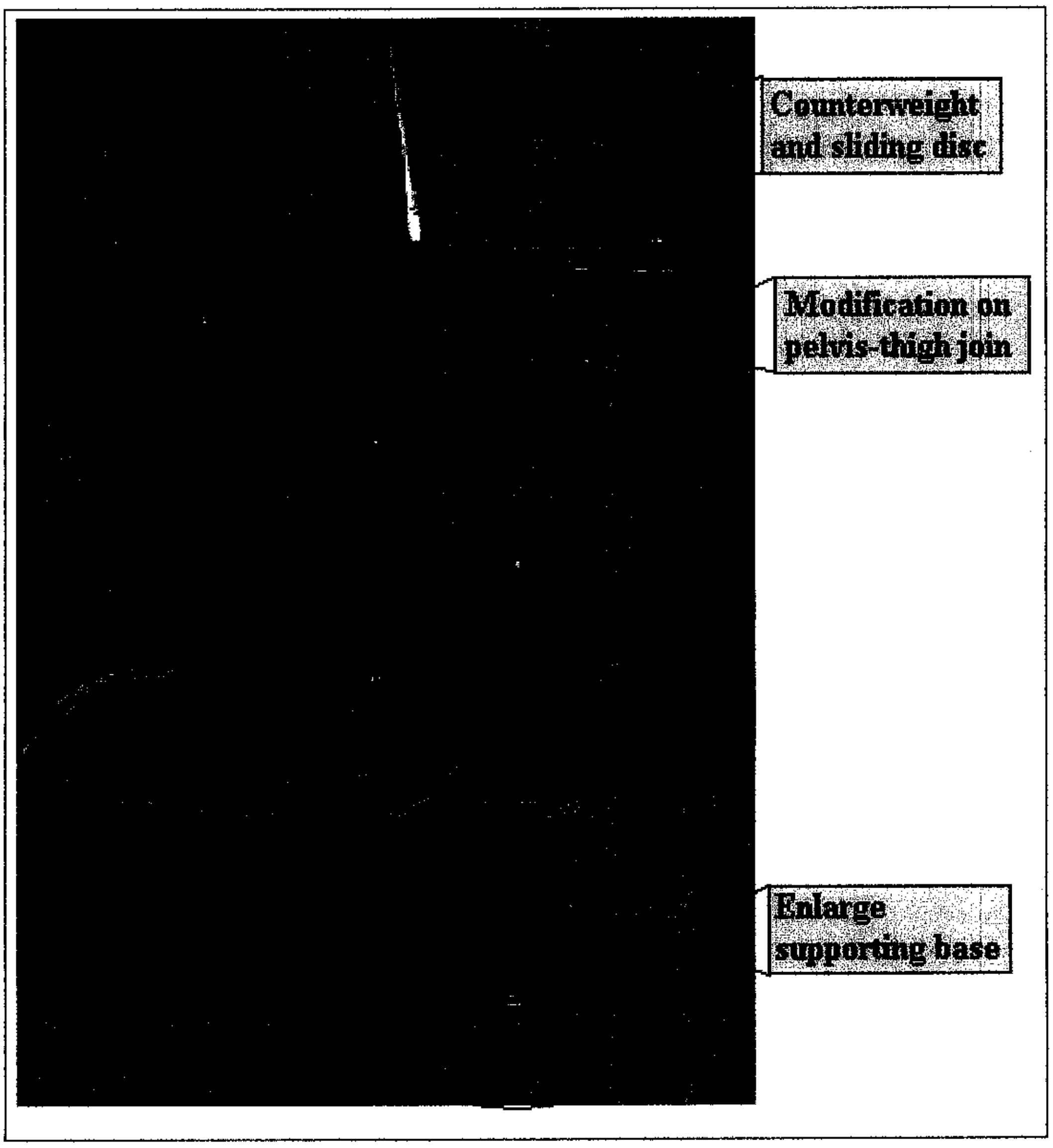

Counterweight and sliding disc
Modification on pelvis-thigh join
Enlarge supporting base

Figure 16 : Final Structure Design and Improvement Done on it

4.1.7 The Overall Features of the Final Structure

The features for final structure can be summarized by the following table 3:

Table 3 : The Overall Features of the Final Structure

Features	**Measures**
Side-to-side length*	0.17 m
Pelvis width	0.09 m
Total height**	0.25 m
Feet length	0.10 m
Foot width	0.06 m
Feet supporting area***	0.10 x 0.06 m^2
Total weight	322.0 g
Total degrees of freedom	5
Weight for dummy load	200 g

* *Measurement taken from left-end of sliding path to the right end of sliding path.*

** *Measurement taken when both legs are in a straight condition including the height of balancing weight.*

*** *Supporting area for one foot where both feet are having similar supporting area.*

4.1.8 The Desired Walking Algorithm

Finished with the walking stability, the next step is to plan for the biped walking algorithm. Because only 5 degree of freedom is used on the structure, it is impossible to copy exactly how humans walk. It has been determined that the constraint is at the ankle side.

To follow the human motion exactly, the ankle should also have at least one degree of freedom. This is to correspond to the leg movement when walking. To be detailed, when the thigh starts to bend forward or backwards, the ankle should also follow the leg bending just as the knee. This is best illustrated in figure 4 that is shown in the theoretical and literature review chapter.

When there is no degree of freedom at the ankle, the way this structure moves is a little bit different compared to human-beings. The desired movement is best described in the following figure. The difference is at the ankle and knee. The ankle is not bent and because of this, the knee has to bend outward, which is in the opposing direction of the real human's knee as shown in the following figure 17.

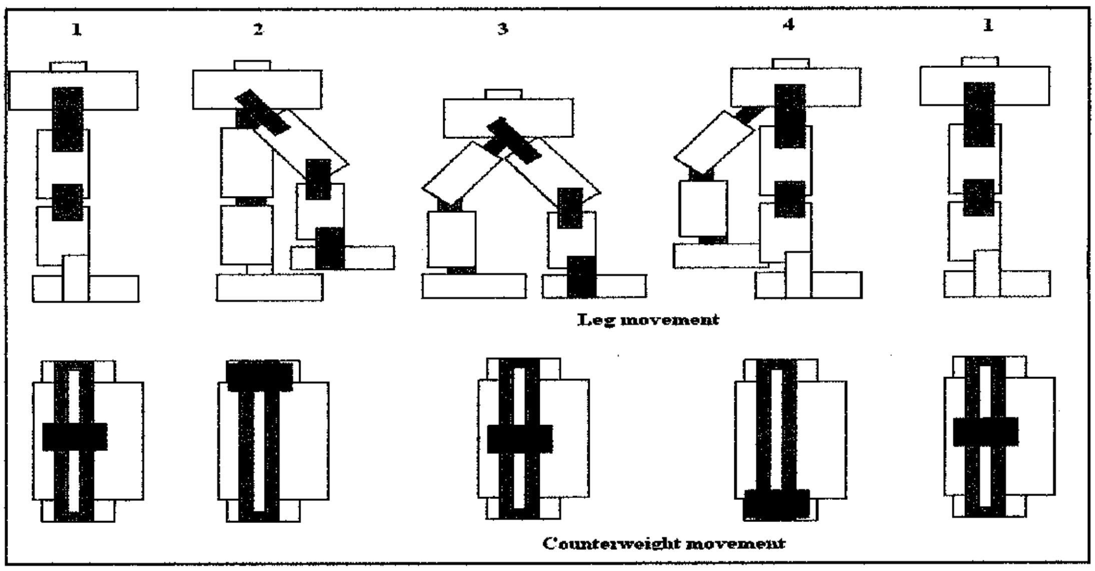


Figure 17 : Walking Algorithm for the Biped

The walking algorithm of the robot has been divided into 6 different leg movements. The figure is also included with the movement of balancing weight. This figure is looking from the upper side. The following is a brief explanation on each of the movements.

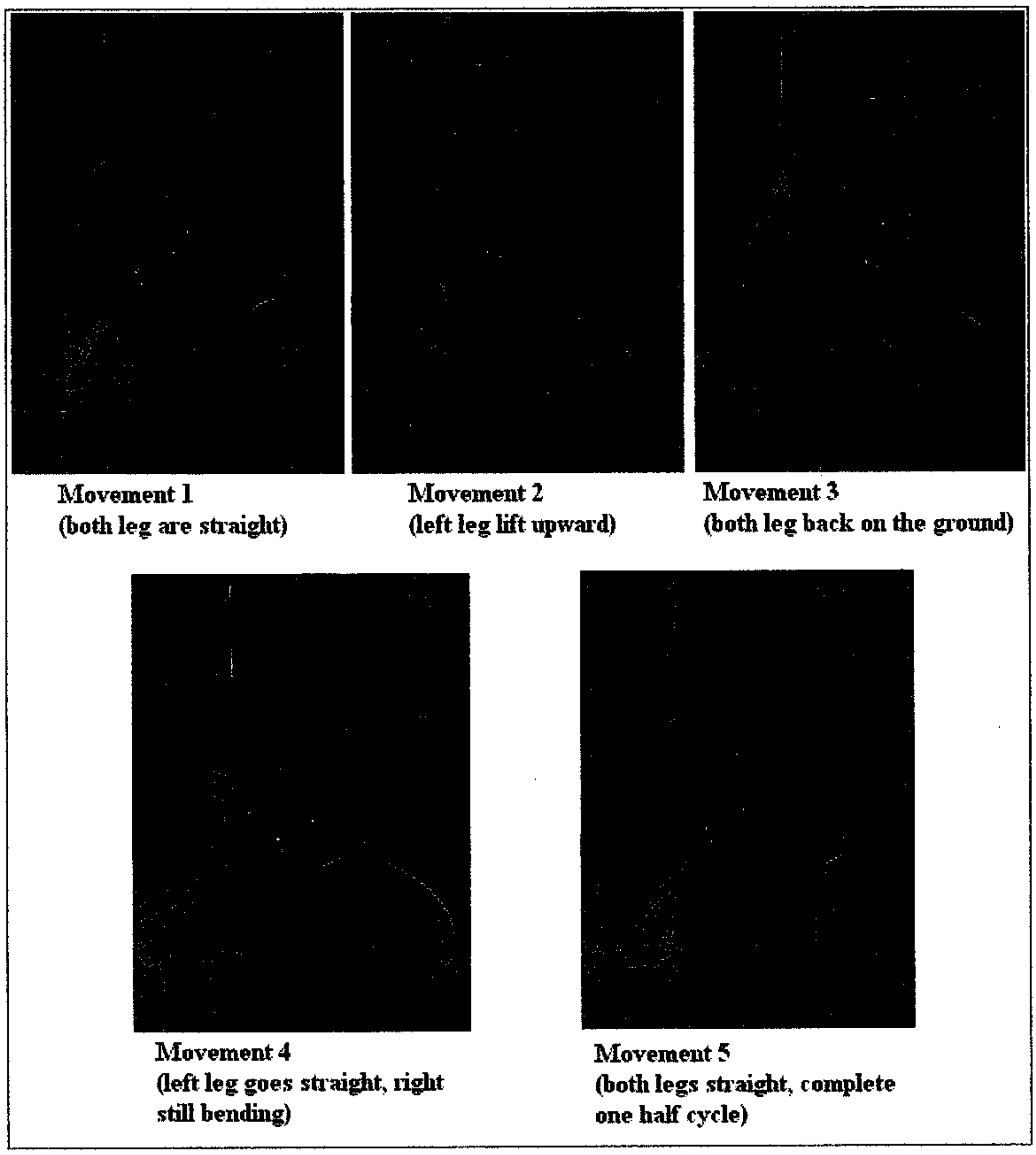
- Movement 1 – The biped begin from its equilibrium position. This means that it is standing straight with both legs. At this equilibrium position, the balancing weight position is at the middle of the structure, as shown in the figure.
- Movement 2 – The movement begins with the biped pulling up its left leg. At this time, it is standing with a single leg. At this condition, the balancing weight will turn to the opposing side, which is to the right side as indicated in the corresponding balancing figure.
- Movement 3 – The movement continues with the right leg to complete its servo turn. This will bring the left leg totally to the ground and after this movement, the feet of the biped is already on the ground. The position of weight balance is back to its equilibrium position.

- Movement 4 – Then, the left leg continues its movement making it as the standing leg of the structure. The left leg servo will turn until the leg stands straight. This will bring the right leg up with its current position (leg still bending). The movement of the right leg will only start after the left leg is fully straight.
- Movement 1 – This movement is just the completion of the one step walking cycle. At this condition, both legs are already standing straight. For the balancing weight, its position is back at the center.

To complete one full walking cycle, similar algorithm (movement 1 – movement 4) is repeated with right leg begin the motion. Thus, for one full walking cycle, a total of 8 movements are implemented. For backward movement algorithm, the same algorithm is applied in reverse order.

4.1.9 Experiment on the Desired Walking Algorithm

The desired walking algorithm is further simulated using the final structure to investigate whether it suits the structure or otherwise, especially on the stability of the structure. This simulation is done manually by performing each walking movement without the controller circuit. Figure 18 illustrates the experiment performed on the structure for half walking cycle.



Movement 1
(both leg are straight)

Movement 2
(left leg lift upward)

Movement 3
(both leg back on the ground)

Movement 4
(left leg goes straight, right still bending)

Movement 5
(both legs straight, complete one half cycle)

Figure 18 : Experiment for Half Walking Cycle

4.1.10 Servomotor Turning Degree

The turning degrees of the servos determine how wide the legs will swing. This will affect the walking algorithm for the biped. The following figure 19 illustrates the maximum swing for each leg component. This is determined manually.

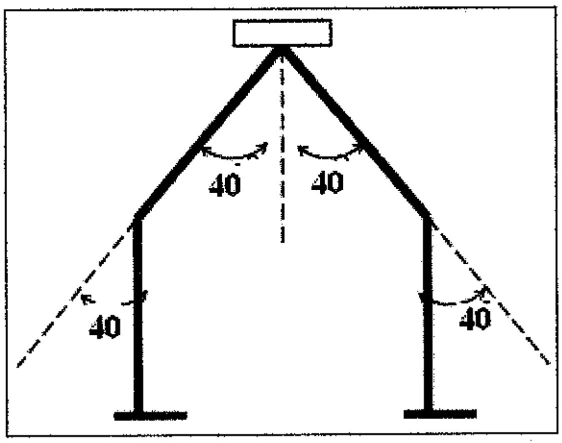


Figure 19 : Turning Degree of the Servomotor

4.2 Programming & Electronics Part

Programming and electronics part are like the energy and the brain of the robot. The existence of electronics circuits enable the servo to move. On the other side, the programming functioned as brain to synchronous the servomotors movement.

With a total of five DOF, the whole structure seems to be complex thus making the programming part also becoming more complex. To simplify the programming part, a concept called master-slave controller is introduced. This method is widely used especially in the project that involves very complex programming structures.

4.2.1 Master-Slave Controller Concept

The master-slave concept is very popular in microcontroller programming especially for those that involve a very complex task. The concept is by dividing those complex tasks into small portion of task and this small portion of task will be assigned to several slave microcontrollers. To control the operation of these entire slave microcontrollers, one master microcontroller is introduced. This master controller controls the entire operation by sending and receiving controlled signal to the slave controller.

To be detailed, the master will send control signal to both slaves at a time and wait for the signal from slave (to indicate that those slaves have finished its part) before proceeding to the next step. The movement of the left leg and the movement of the right leg is controlled by two different microcontroller. These two slaves will fetch the pulse width modulation signal (PWM) to the servomotor according to the control signal supplied by the master.

4.2.2 The Master Microcontroller

For controlling purposes the state machine is implemented in programming the master microcontroller. The state machine is said to be the representation of the walking algorithm.

As briefly discussed earlier, the walking algorithm has a total of 8 leg movements. In this state machine, this 8 leg movement is represented by 8 states. The principle is, the execution of the next state can only be performed when the current state has been completed. This methodology is followed for the next movement.

Firstly, state 8 will be executed and when it has finished, state 1 will take over followed by state 2 until state 7. For the backward movement, the reverse of forward movement (movement 8, movement 7……, and movement 1) is applied to the structure because the coding and algorithm is similar. More details on the coding is included in the appendix.

For controlling purposes, 3 bits binary signal is implemented in each of these 8 states. 3 bits is chosen because it is an ideal representation for 8 walking movements/states. These 8 different bits will be fetch into both of the slave and both slaves will fetch PWM signal according to this bits. The following table 4 indicated these 8 bits and its different walking states.

Table 4 : Binary Representation for Walking States

State	Control signal (Binary bits)	Walking Movement
0	000	Movement 8
1	001	Movement 1
2	010	Movement 2
3	011	Movement 3
4	100	Movement 4
5	101	Movement 5
6	110	Movement 6
7	111	Movement 7

4.2.3 The Slave Microcontroller

The function of slave microcontroller is to supply the PWM signal to the servomotor. The signal supplied is according to the control signal fetched by the master microcontroller. The following table 5 and 6 illustrated the duty cycle of the PWM signal for different turning degrees.

Table 5 : High and Low Time (Duty Cycle) of Pelvis and Knee Servomotor

Servo Position	Straight (us)	Forward 45 (us)	Backward 45 (us)
Left pelvis	1205	880	1530
Right pelvis	1205	1530	880
Left knee	1410	1740	1110
Right knee	1410	1110	1740
Low time (ms)	20	20	20

Table 6 : High and Low Time (Duty Cycle) of Counterweight Servomotor.

Servo Position	Neutral (us)	Left 90 (us)	Right 90 (us)
Balancing	1480	500	2400
Low time (ms)	20	20	20

4.2.4 Microcontroller PIC16F877 & PIC16F84A

The servomotor movement is controlled by the PIC16F877 (Master controller) and PIC16F84A (Slave controller). Figure 20 illustrated this microcontroller. The PIC16F877 is also provided in the Appendix H.

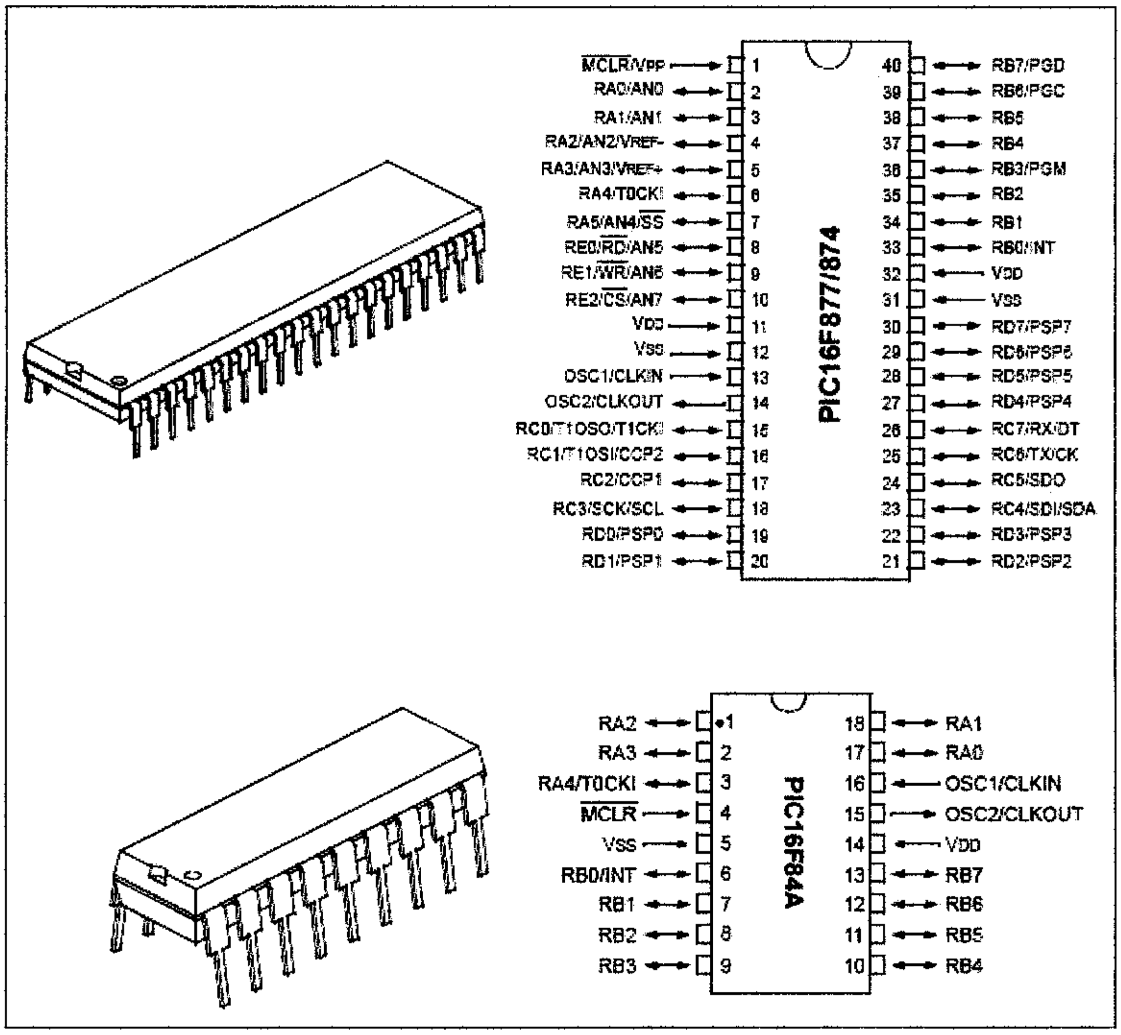


Figure 20 : The Structure of PIC16F877 and PIC16F84A (Courtesy of Microchip)

The microcontroller is programmed using C-language rather than using the assembly language. The use of C-language simplifies the programming code compared to the assembly language. Upon successfully compiling the program, the next step is to upload the program into the microcontroller. This is done using the "downloader" board.

4.2.5 Servomotor Controller Circuit

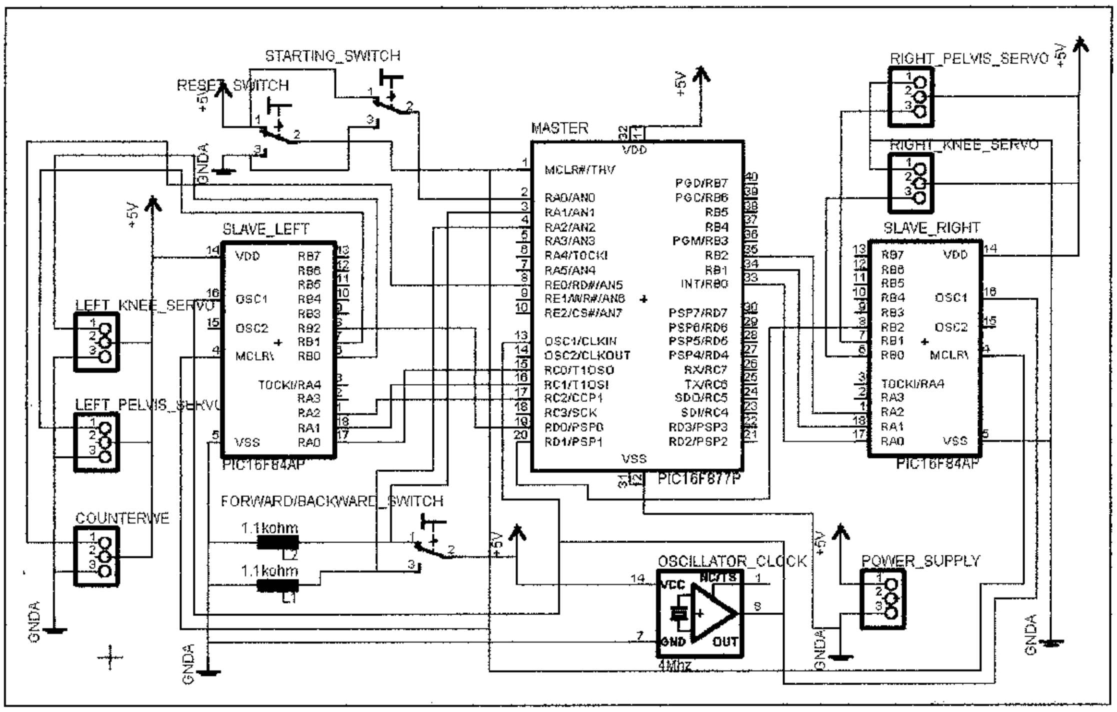


Figure 21 : Servomotor Controller Circuit Using PIC16F877 and PIC16F84A

Figure 21 above illustrates the servomotor controller circuit. The circuit is very simple and requires less electronics components. The only component that is needed in this circuit is the PIC itself, oscillator clock, resistor, switch and the servomotor connector. A clear view of this circuit is provided in Appendix C and D.

The input/output (I/O) pin module of the PIC16F877 and PIC16F84A is divided into input pin and output pin. The summary of the PIC16F877 and PIC16F84A pins usage is tabulated in Appendix E for references.

4.2.6 Pulse Width Modulation (PWM)

The pulse width modulation or PWM is the best way to control the turning degree of the servomotor. This method makes use of the average DC voltage that is sent to the servomotor.

The average DC voltage is varied by varying the high time of the PWM signal. This variation of high time varies proportionally to the average DC voltage and to the turning degree of the servomotor. The following figure 22 and 23 illustrates the PWM signal applied to the left pelvis and counterweight servomotors respectively during walking movement. The other three servomotor's PWM waveform is similar to the left pelvis waveform because the values of high and low time are the same.

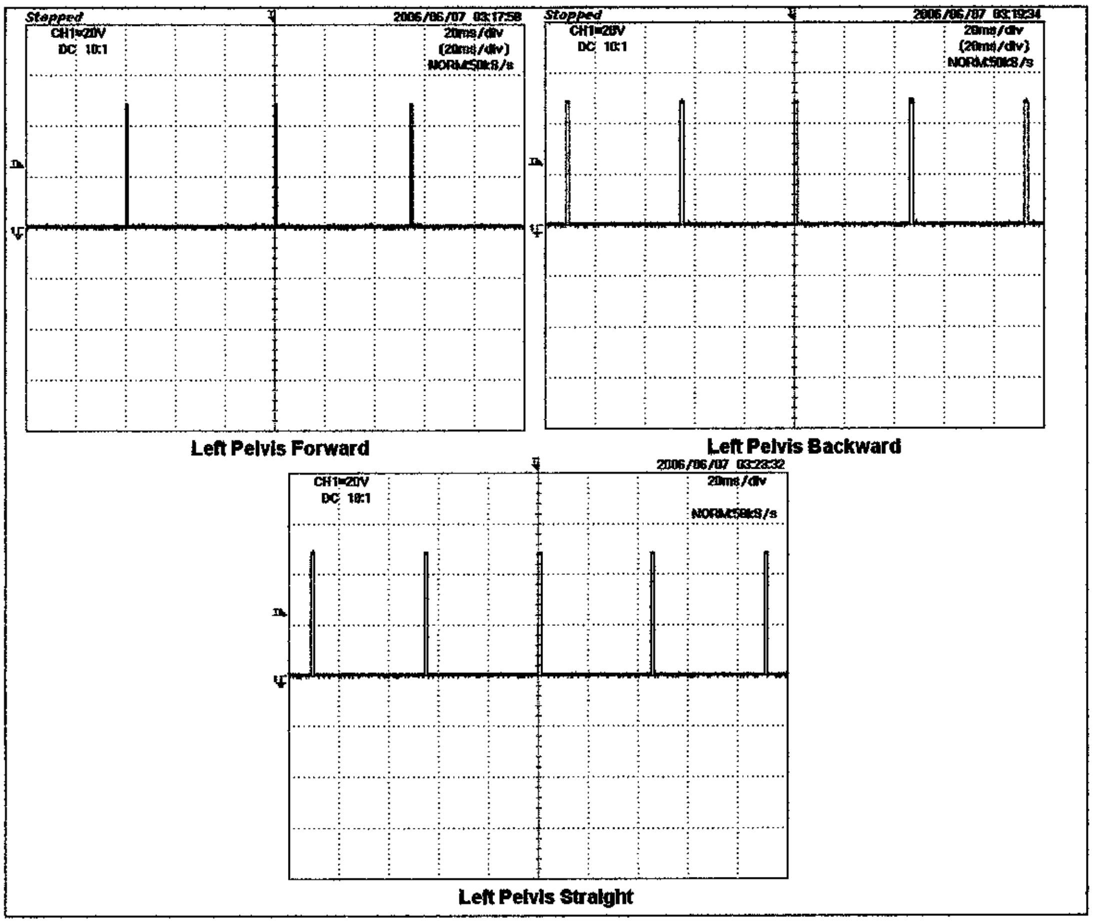


Figure 22 : PWM Signal for Left Pelvis Servomotor

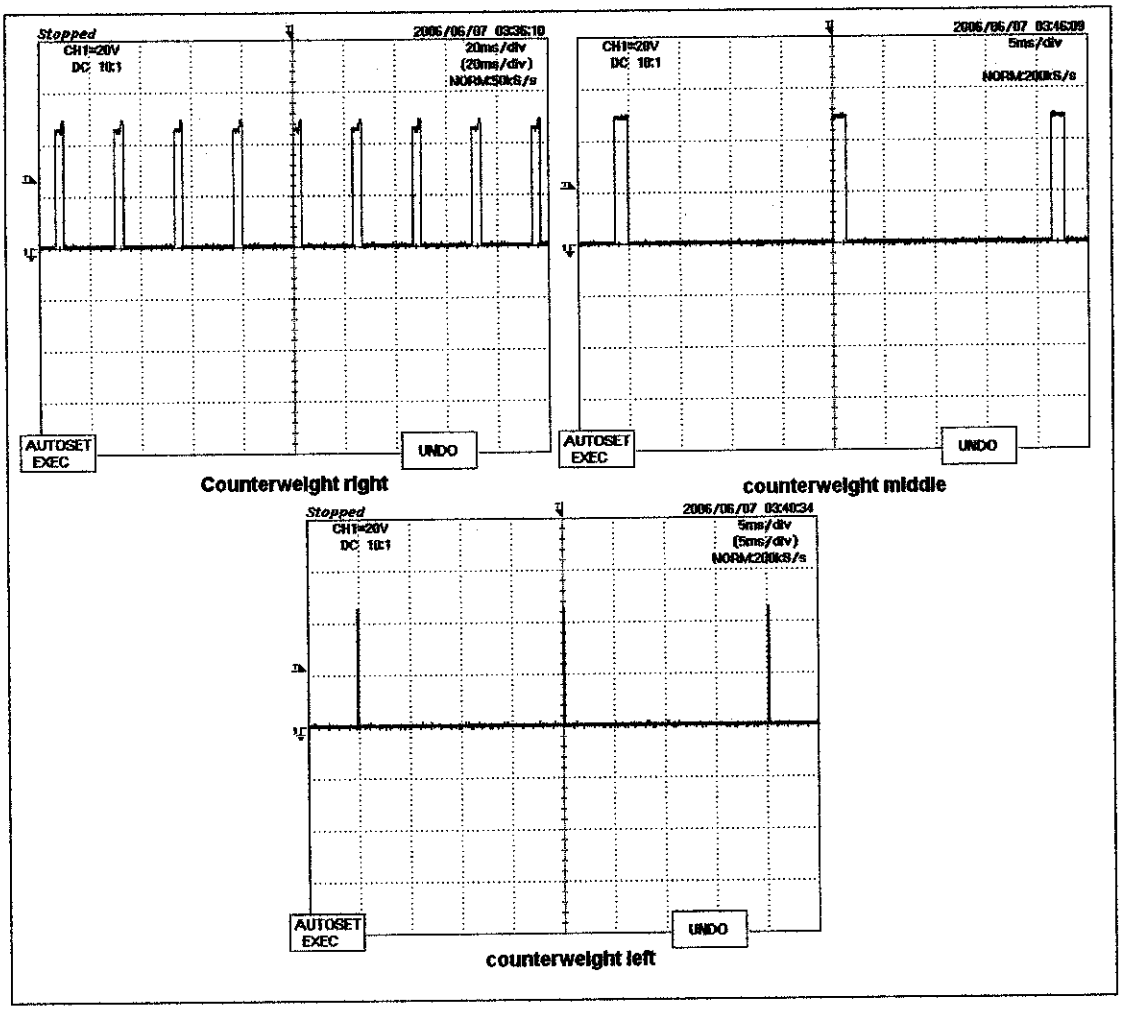


Figure 23 : PWM Signal for Counterweight Servomotor

The high time of turning degree for each servomotor is different for different servomotors. The value of "high time and low time" is varied for different servomotors. These values are obtained through experiments of trial and error.

CHAPTER 5
CONCLUSION

As a conclusion, the author has managed to achieve the first objective, which is obtaining the stable and rigid biped structure. The final structure as briefly discussed in the result is the optimized structure. This structure's stability has been achieved and simulated manually without the controller. This is simulated by pulling up one of the structure's leg and let it stand only with one leg as simulated in the result (part experiment on the desire walking algorithm).

On the controller circuit part, the author has successfully constructed it using printed circuit board (PCB) as illustrated in the result and Appendix C and D. This controller circuit is working very well.

The author could not manage to fully achieve the walking algorithm for the structure. To be detailed, the desired movement of the servo (turning to desired degree) has been obtained. However, this desired movement can only be simulated by hanging the structure freely in the air. When the structure is placed on the ground, the desired movement is not working.

CHAPTER 6
RECOMMENDATION

The main cause of failure for this project is due to its overall weight. Thus, this constraint needs to be overcome if this project is to be proceed within the near future. The following recommendations can be considered to overcome this weight constraint.

6.1 Fabrication Material

One of the main contributors to the overall structure weight of the structure is the fabrication material. In this project, the fabrication material (aluminum L-shape) used is not light enough to reduce the overall weight. Thus, to further reduce the overall weight, other lighter materials such as plastic or perspex can be chosen. Both of these two materials is light and they are easy to fabricate. If the aluminum is to be used again, it should be fabricated in such a way that its weight is minimized. This can be done by drilling holes on the aluminum.

6.2 Servomotor Selection

Another main reason that led to the failure of making the structure walk smoothly is the servomotor problem. In this project, the servomotor used is a low torque type of servomotor, the Futaba S-148. The torque is insufficient to lift the leg. Thus, it is very important to choose the more powerful servomotor if this project is going to be proceed with next time. The most suitable servomotor is the Futaba S9303, which has more powerful torque, compared the existing one.

6.3 Degree of Freedom

The degree of freedom can also be taken into consideration. In this project, the method of tackling the structure's stability is through the use of the counterweight. The use of counterweight increases the overall weight. The counterweight methods of balancing the structure can be eliminated by introducing one extra DOF on each the pelvis. This extra DOF will shift the center of mass of the structure to the middle of the supporting base thus balancing the structure. Besides, by adding more DOF, it is easier to perform other walking motions such as making the structure to turn left and right.

The drawback of adding more DOF is in the controller programming part. In other words, the more DOF the structure has the more complex the programming will become. However, this problem can be solved through proper programming architecture by introducing the master-slave concept.

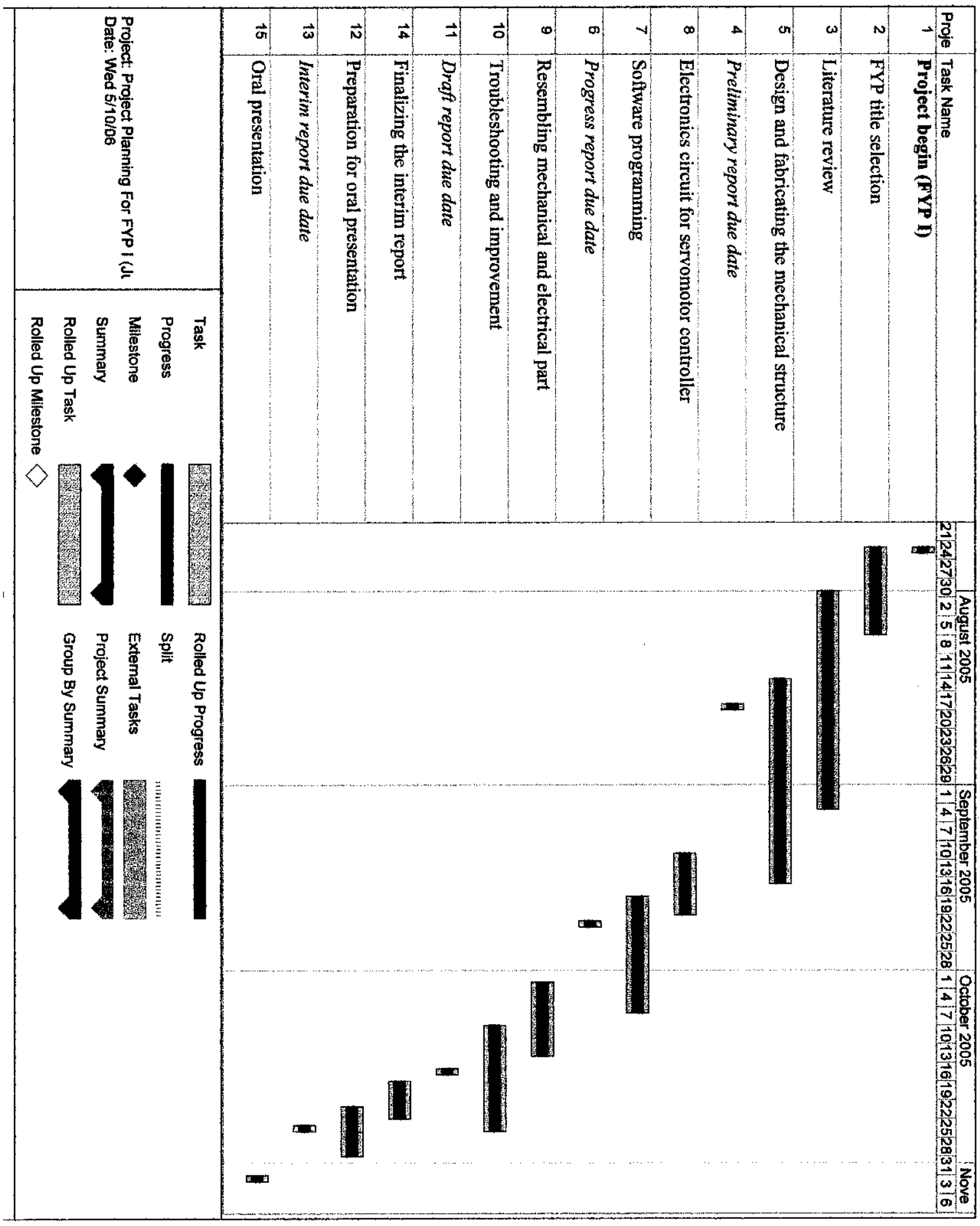
REFERENCES

1. Machines & Mechanisms Applied Kinematics Analysis, 2nd Edition, David H. Myszka, Prentice Hall.
2. Robot Androids and Animatronics, 2nd Edition, John Iovine, McGraw Hill.
3. Build a Remote Control Robot, David R. Shircliff, McGraw Hill.
4. Jochen Zimmermann, "Balancing of Biped Robot Using Force Feedback", University of Western Australia, 2003.
5. Joan Battle, Enric Hospital, Jeroni Sallelas & Marc Carreras, "Development of a Biped Robot", Universitat de Girona.
6. Mohd Shariman B. M Sharif, "Mobile Robots: Obstacle Avoidance and Maneuvering", Electrical & Electronic Engineering, Universiti Teknologi PETRONAS.
7. http://www.androidworld.com
8. http://www.galileo.org/robotics/design.html
9. http://www.picant.com/robot/robot.html

APPENDICES

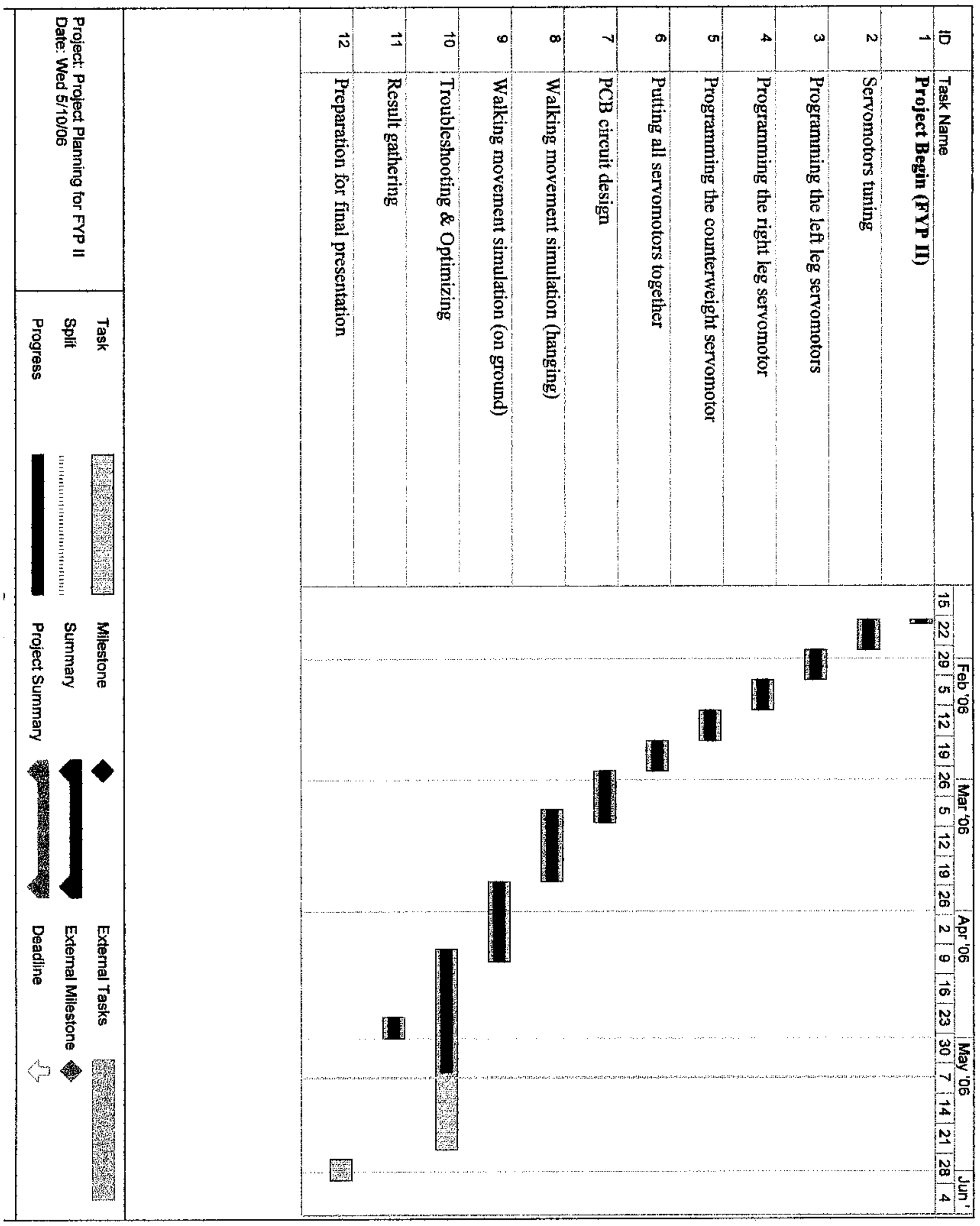
APPENDIX A

PROJECT PLANNING FOR FYP I



APPENDIX B

PROJECT PLANNING FOR FYP II



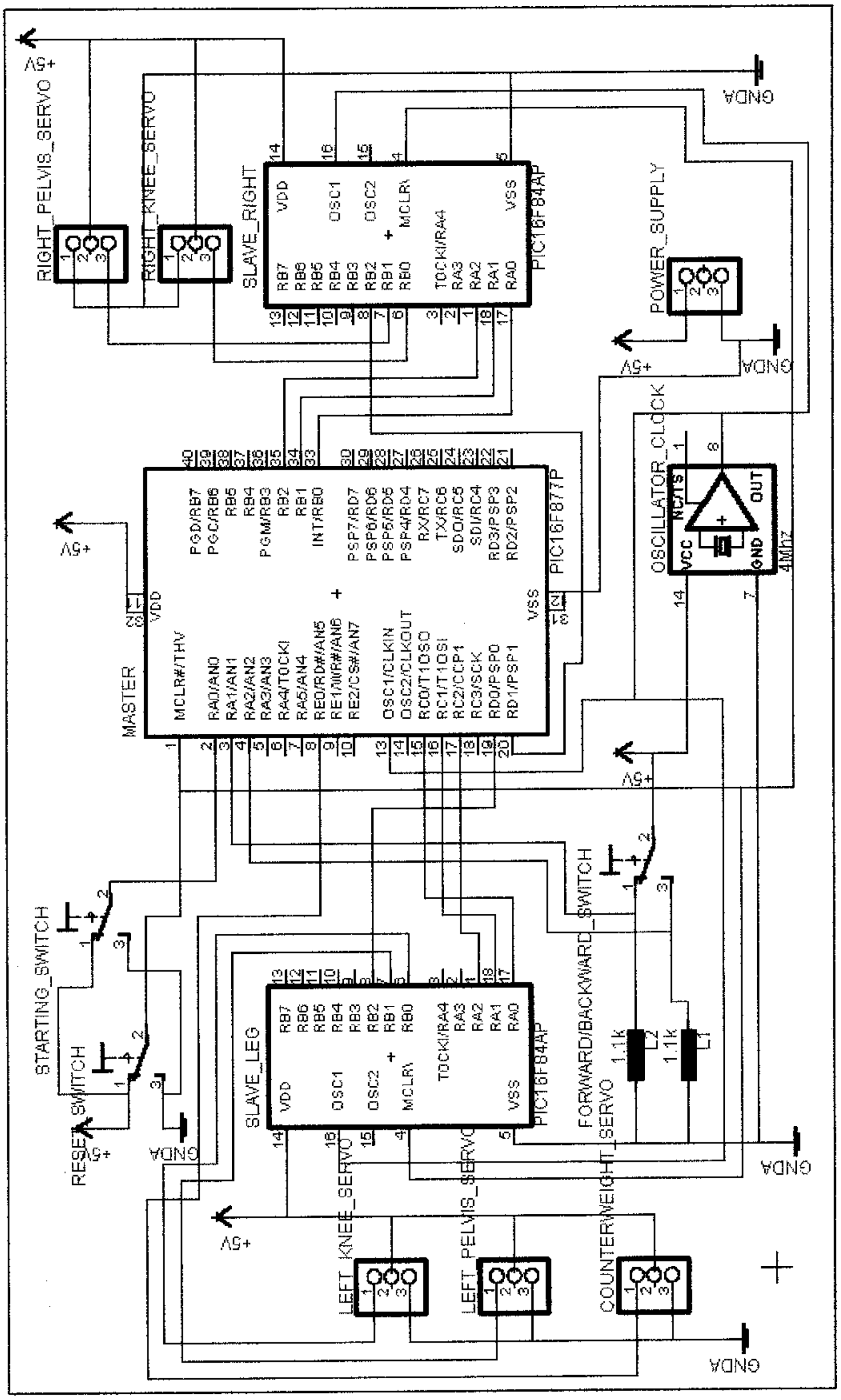
ID	Task Name
1	**Project Begin (FYP II)**
2	Servomotors tuning
3	Programming the left leg servomotors
4	Programming the right leg servomotor
5	Programming the counterweight servomotor
6	Putting all servomotors together
7	PCB circuit design
8	Walking movement simulation (hanging)
9	Walking movement simulation (on ground)
10	Troubleshooting & Optimizing
11	Result gathering
12	Preparation for final presentation

Project: Project Planning for FYP II
Date: Wed 5/10/06

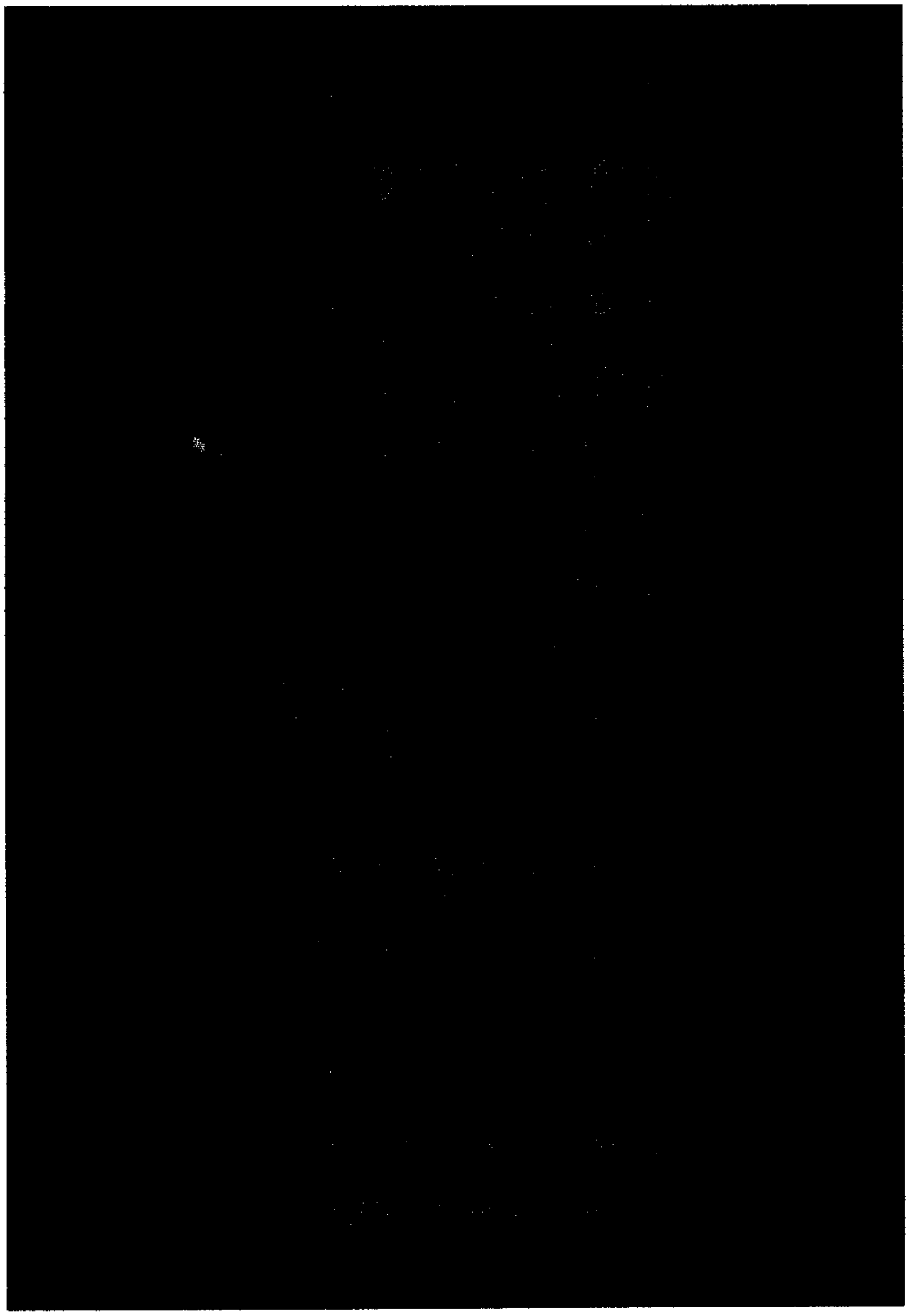
Task	▨	Milestone	◆	External Tasks	▨
Split	┈	Summary	▬	External Milestone	◆
Progress	■	Project Summary	▬	Deadline	⇩

APPENDIX C

SERVOMOTOR CONTROLLER CIRCUIT (SCHEMATIC)



APPENDIX D
SERVOMOTOR CONTROLLER CIRCUIT (VERO-BOARD DIAGRAM)



APPENDIX E

SUMMARY OF PIC16F877 & PIC16F84A PINS USAGE

Pin No	Pin Description	Usage	Connected to
1	MCLR	Program reset	Switch 3(SW3)
2	Pin A0	Input(starting switch)	Switch 1(SW1)
3	Pin A1	Input (walking forward switch)	Switch 2(SW2)
4	Pin A2	Input (walking backward switch)	Switch 2(SW2)
11&32	Vdd	+5V power supply	+5V
12&31	Vss	0V power supply	0V
13	ClkIn	Clock signal	Oscillator clock
33	Pin B0	Output (left servomotor)	Pin A0(Slave left)
34	Pin B1	Output (left servomotor)	Pin A1 (Slave left)
35	Pin B2	Output (left servomotor)	Pin A2 (Slave left)
15	Pin C0	Output (right servomotor)	Pin A0(Slave left)
16	Pin C1	Output (right servomotor)	Pin A1 (Slave left)
17	Pin C2	Output (left servomotor)	Pin A2 (Slave left)
8	Pin E0	Output PWM(counterweight servo)	Pin header

Pin No	Pin Description	Usage	Connected to
1	Pin A2	Control signal	Pin B2/Pin C2 (master)
4	MCLR	Reset	Switch 3
5	Vss	0V Power supply	0V
7	Pin B0	PWM signal	Knee servomotor
8	Pin B1	PWM signal	Pelvis servomotor
14	Vdd	+5V power supply	+5V
16	ClkIn	Clock signal	Oscillator clock
17	Pin A0	Control signal	Pin B0/Pin C0 (Master)
18	Pin A1	Control signal	Pin B1/Pin C1 (Master)

APPENDIX F

PROGRAMMING CODE (MASTER)

```
/*******************************************************************/
/*                                                                 */
/* Tittle      :Design and implementation bipedal robot            */
/* Programmer  :Saiful bin A'ee                                    */
/* ID          :3119                                               */
/* Supervisor  :Mohd Haris bin Md Khir                             */
/*                                                                 */
/*******************************************************************/

////////////////////////////////////////////////////////////////////
//                                                                //
// -The program is design to control the movement of 5 sevomotor //
// -The architecture of the program is based on state machine     //
// -The main state machine is movement_1() - movement_8(),        //
//  the rest is supporting function.                              //
//                                                                //
////////////////////////////////////////////////////////////////////

/**********This portion is program header file**************/
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,NOPUT,NOBROWNOUT
#use delay(clock=10000000)
#use standard_io(A,B,C,D,E)
#include <LCD.C>
#include <string.h>
/**********************************************************/

/*****This portion is for variable declaration in memory***/

#define kiri 100
#define tgh 1380
#define kanan 3380

#define low 20
#define lo 20
#define time 1000
#define time2 20
/*********************************************************/

/*******Variable declaration (global declaration)*********/
int a,b,c,d,e,f,g,h,i,j,k,l,m;
/*********************************************************/

/*******Function declaration*****************************/
void movement_1(void);     //---->  |\
void movement_2(void);     //---->  /\
void movement_3(void);      //---->  /|
void movement_4(void);      //---->  ||
void movement_5(void);      //---->  |\
void movement_6(void);      //---->  /\
void movement_7(void);      //---->  /|
void movement_8(void);      //---->  ||

void weight_0(void);
void weight_90(void);
void weight_180(void);

void initial(void);
void walk_forward(void);
void walk_backward(void);

void wait(void);
/**********************************************************/


/*******The beginning of function definition**************/

/** Movement 8 - both legs back to straight position ******************/

void movement_8()
{
   output_B(0x00);
   output_C(0x00);
   wait();
}
/**************************************************************************/

/****** Movement 1 is for pulling left leg upward ********/
/****** while leaving the right leg stand straight ******/
void movement_1()     // |\
{
   output_C(0x01);
   output_B(0x01);
   wait();
}
/********************************************************/


/**** Movement 2 is for leg spread forward *************/

void movement_2()     // /\
{
   output_C(0x02);
   output_B(0x02);
   wait();
}
/*********************************************************/
```

```
/*Movement 3 - left leg goes straight, right leg bending*/

void movement_3()     //    /|
{
   output_B(0x03);
   output_C(0x03);
   wait();
}
/****************************************************/


/****** Movement 4 - both leg are straight *************/

void movement_4()     //    ||
{
   output_B(0x04);
   output_C(0x04);
   wait();
}

/************************************************************/


/** Movement 5 - right leg bend forward, left leg straight ******/

void movement_5()     // |\
{
   output_C(0x05);
   output_B(0x05);
   wait();
}
/*******************************************************************/


/***** Similar to movement 2 ******************************************/
void movement_6()     // /\
{
   output_C(0x06);
   output_B(0x06);
   wait();
}
/*******************************************************************/


/***Movement 7 - right leg stright, left leg bending backward **********/

void movement_7()     //    /|
{
   output_B(0x07);
   output_C(0x07);
   wait();
}
/**********************************************************************/


/*** Balancing weight move to the right side ***************************/

void weight_rite()
{
   for(i=0;i<=time2;i++)
   {
      output_high(PIN_E0);
      delay_us(kanan);
      output_low(PIN_E0);
      delay_ms(low);
   }
}
/**********************************************************************/


/*** Balancing weight in the middle ************************************/

void weight_mid()
{
   for(j=0;j<=time2;j++)
   {
      output_high(PIN_E0);
      delay_us(tgh);
      output_low(PIN_E0);
      delay_ms(lo);
   }
}
/**********************************************************************/


/**** Balancing weight move to the left side ***************************/

void weight_left()
{
   for(k=0;k<=time2;k++)
   {
      output_high(PIN_E0);
      delay_us(kiri);
      output_low(PIN_E0);
      delay_ms(lo);
   }
}
/**********************************************************************/


/**** This is the main function ****************************************/

main()
```

```
{

   loop:

   initial();

   walk_forward();

   walk_backward();

   goto loop;

}
/*************************************************************************/

/**** Forward movement algorithm *****************************************/
/*    -This movement is activated when switch 1 is triggered             */
void walk_forward()
{
   while(!input(PIN_A1))
   {
      weight_mid();

      movement_8();
      delay_ms(time);

      weight_rite();

      movement_1();
      delay_ms(time);

      movement_2();
      delay_ms(time);

      weight_mid();

      weight_left();

      movement_3();
      delay_ms(time);

      movement_4();
      delay_ms(time);

      weight_mid();

      weight_left();

      movement_5();
      delay_ms(time);

      movement_6();
      delay_ms(time);

      weight_mid();

      weight_rite();

      movement_7();
      delay_ms(time);
   }
}
/*************************************************************************/


/**** Backward movement algorithm *************************************/
/*    -This movement is activated when switch 2 is triggered          */
void walk_backward()
{
   while(!input(PIN_A2))
   {
      weight_mid();

      movement_8();
      delay_ms(time);

      weight_rite();

      movement_7();
      delay_ms(time);

      movement_6();
      delay_ms(time);

      weight_mid();

      weight_left();

      movement_5();
      delay_ms(time);

      movement_4();
      delay_ms(time);

      weight_mid();

      weight_left();

      movement_3();
      delay_ms(time);

      movement_2();
      delay_ms(time);
      weight_left();

      weight_rite();

      movement_1();
      delay_ms(time);
   }
}
/***********************************************************************/


/**** Starting movement ***********************************************/
/*   - During this movement, both leg are straight until switch 1 is    */
/*     triggered to initiate forward movement                           */
void initial()
{
   while(!input(PIN_A0))
   {

      movement_8();

      weight_mid();
   }
}
/***********************************************************************/

void wait()
{
   do
   {
      a = input_D() & 0x03;
   }
   while(a!=3);
}
```

APPENDIX G

PROGRAMMING CODE (SLAVE LEFT LEG)

```
#include <16F84A.h>
#fuses HS,NOWDT
#use delay(clock=10000000)

#define plvs_L0   1205
#define plvs_Lb45 1530     //initial value is 1630
#define plvs_Lf45 880      //initial value is 780

#define knee_L0   1410
#define knee_Lb45 1110     //initial value is 1010
#define knee_Lf45 1740     //initial value is 1840

#define plvs_R0   1205
#define plvs_Rf45 1530     //initial value is 1630
#define plvs_Rb45 880      //initial value is 780

#define knee_R0   1410
#define knee_Rf45 1110     //initial value is 1010
#define knee_Rb45 1740     //initial value is 1840

#define cw0 500
#define cw90 1480
#define cw180 2400

#define low 20
#define lo 20
#define time 14


int a,b,c;
int z;

void main()
{
   loop:

   a = input_A() & 0x07;

   switch(a)
   {
      case 0:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_L0);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_L0);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }

    case 1:
    {
        for(z=0;z<=time;z++)
        {
            output_high(PIN_B0);
            delay_us(knee_Lb45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Lf45);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }

    case 2:
    {
        for(z=0;z<=time;z++)
        {
            output_high(PIN_B0);
            delay_us(knee_Lb45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Lf45);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }

    case 3:
    {
        for(z=0;z<=time;z++)
        {
```

```
        output_high(PIN_B0);
        delay_us(knee_L0);
        output_low(PIN_B0);
        delay_ms(low);

        output_high(PIN_B1);
        delay_us(plvs_L0);
        output_low(PIN_B1);
        delay_ms(low);
    }
    output_high(PIN_B2);
    goto loop;
}

case 4:
{
    for(z=0;z<=time;z++)
    {
        output_high(PIN_B0);
        delay_us(knee_L0);
        output_low(PIN_B0);
        delay_ms(low);

        output_high(PIN_B1);
        delay_us(plvs_L0);
        output_low(PIN_B1);
        delay_ms(low);
    }
    output_high(PIN_B2);
    goto loop;
}

case 5:
{
    for(z=0;z<=time;z++)
    {
        output_high(PIN_B0);
        delay_us(knee_L0);
        output_low(PIN_B0);
        delay_ms(low);

        output_high(PIN_B1);
        delay_us(plvs_L0);
        output_low(PIN_B1);
        delay_ms(low);
    }
    output_high(PIN_B2);
    goto loop;
}

case 6:
{
        for(z=0;z<=time;z++)
        {
            output_high(PIN_B0);
            delay_us(knee_Lf45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Lb45);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }

    case 7:
    {
        for(z=0;z<=time;z++)
        {
            output_high(PIN_B0);
            delay_us(knee_Lf45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Lb45);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }

    default:
    {
        for(z=0;z<=time;z++)
        {
            output_high(PIN_B0);
            delay_us(knee_L0);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_L0);
            output_low(PIN_B1);
            delay_ms(low);
        }
        output_high(PIN_B2);
        goto loop;
    }
}
```

```
    goto loop;
}
```

APPENDIX H

PROGRAMMING CODE (SLAVE RIGHT LEG)

```
#include <16F84A.h>
#fuses HS,NOWDT
#use delay(clock=10000000)

#define plvs_L0   1205
#define plvs_Lb45 1530     //initial value is 1630
#define plvs_Lf45 880      //initial value is 780

#define knee_L0   1410
#define knee_Lb45 1110     //initial value is 1010
#define knee_Lf45 1740     //initial value is 1840

#define plvs_R0   1205
#define plvs_Rf45 1530     //initial value is 1630
#define plvs_Rb45 880      //initial value is 780

#define knee_R0   1410
#define knee_Rf45 1110     //initial value is 1010
#define knee_Rb45 1740     //initial value is 1840

#define cw0 500
#define cw90 1480
#define cw180 2400

#define low 20
#define lo 20
#define time 14

int a;
int z;

void main()
{
   loop:

   a = input_A() & 0x07;

   switch(a)
   {
      case 0:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_R0);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_R0);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 1:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_R0);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_R0);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 2:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_Rf45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Rb45);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 3:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_Rf45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Rb45);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 4:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_R0);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_R0);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 5:
      {
         for(z=0;z<=time;z++)
         {
            output_high(PIN_B0);
            delay_us(knee_Rb45);
            output_low(PIN_B0);
            delay_ms(low);

            output_high(PIN_B1);
            delay_us(plvs_Rf45);
            output_low(PIN_B1);
            delay_ms(low);
         }
         output_high(PIN_B2);
         goto loop;
      }

      case 6:
      {
         for(z=0;z<=time;z++)
      {
         output_high(PIN_B0);
         delay_us(knee_Rb45);
         output_low(PIN_B0);
         delay_ms(low);

         output_high(PIN_B1);
         delay_us(plvs_Rf45);
         output_low(PIN_B1);
         delay_ms(low);
      }
      output_high(PIN_B2);
      goto loop;
   }

   case 7:
   {
      for(z=0;z<=time;z++)
      {
         output_high(PIN_B0);
         delay_us(knee_R0);
         output_low(PIN_B0);
         delay_ms(low);

         output_high(PIN_B1);
         delay_us(plvs_R0);
         output_low(PIN_B1);
         delay_ms(low);
      }
      output_high(PIN_B2);
      goto loop;
   }

   default:
   {
      for(z=0;z<=time;z++)
      {
         output_high(PIN_B0);
         delay_us(knee_R0);
         output_low(PIN_B0);
         delay_ms(low);

         output_high(PIN_B1);
         delay_us(plvs_R0);
         output_low(PIN_B1);
         delay_ms(low);
      }
      output_high(PIN_B2);
      goto loop;
   }
}
```

```
   goto loop;
}
```

APPENDIX I

PIC16F877 DATASHEET

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

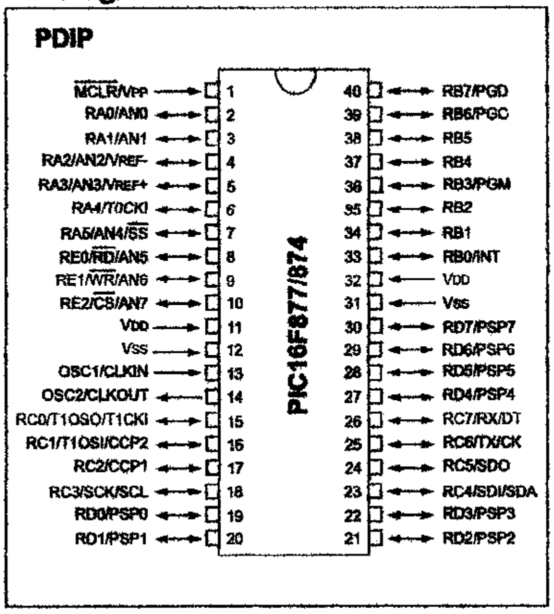
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- *Commercial, Industrial and Extended temperature* ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

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

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS[4]	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/$\overline{\text{SS}}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST[1]	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST[2]	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST[2]	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

APPENDIX J

PIC16F84A DATASHEET

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

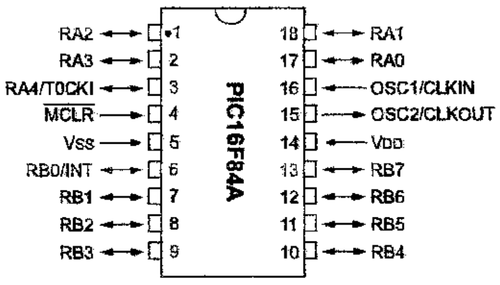
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - *25 mA sink max. per pin*
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

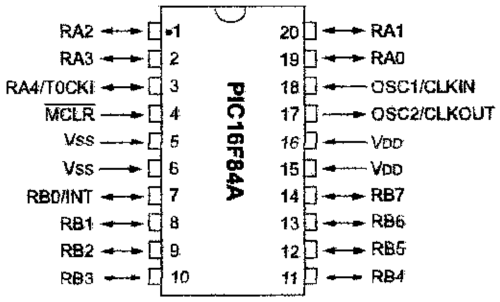
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- *Power-on Reset (POR), Power-up Timer (PWRT),* Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- *Fully static design*
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- *Low power consumption:*
  - < 2 mA typical @ 5V, 4 MHz
  - 15 μA typical @ 2V, 32 kHz
  - < 0.5 μA typical standby current @ 2V

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS[(3)]	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST[(1)]	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST[(2)]	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST[(2)]	Interrupt-on-change pin. Serial programming data.
V_{SS}	5	5	5,6	P	—	Ground reference for logic and I/O pins.
V_{DD}	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.