BORANG PENGESAHAN STATUS TESIS:

JUDUL: DEVELOPMENT OF A HUMANOID ROBOT

SESIPENGAJIAN: 2007/2008

Saya CHE NOOR HAIZUL BIN CHE ANI

(HURUF BESAR)

mengaku membenarkan tesis (PSM/Sarjana/Doktor Falsafah)* ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hak milik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Silia tandakan ( )
   - SULIT (Mengandungi maklumat yang berdjarah keselamatan atau kepentingan Malaysia seperti yang termakub di dalam AKTA RAHSIA RASMI 1972)
   - TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
   - √ TIDAK TERHAD

Disahkan oleh

(TANDATANGAN PENULIS) (TANDATANGAN PENYELIA)

Alamat-Perkara:

No.89 Taman Sultan Yusof
3300, Kuala Kangsar,
Perak

Tarikh: 5 MAY 2008

CATATAN:
* Potong yang tidak berkenaan.
** Jika tesis ini SULIT atau TERHAD, sila lamparkan surat daripada pihak berkorespondensi/organisasi berkenaan dengan menyatakan sekali sebaik dan tempoh tesis ini perlu dikelaskan sebagai SULIT atau TERHAD.
♦ Tesis dimaksudkan sebagai tesis bagi ijazah Doktor Falsafah dan Sarjana secara penyelidikan, atau disertasi bagi pengajian secara kerja konsus dan penyelidikan, atau Laporan Projekt Sarjana Muda (PSM).
DEVELOPMENT OF A HUMANOID ROBOT (HAMIZDN)

"I hereby declare that I have read this thesis and my opinions are sufficient in terms of scope and quality for the award of Bachelor Degree of Electrical Engineering (Mechatronic)"

Signature: ............................................................

Name of supervisor: Assoc. Prof. Dr. Zaharuddin b Mohamed

Date: 25 APRIL 2008

Che Noor Haji Zil Bin Che Ani

A thesis submitted in partial fulfilment of requirement for the award of the degree of Bachelor of Electrical Engineering (Mechatronics)

Faculty of Electrical Engineering
University of Technology, Malaysia

APRIL 2008
DEVELOPMENT OF A HUMANOID ROBOT
(HAMIZON)

CHE NOOR HAJIZUL BIN CHE ANI

A thesis submitted in partial fulfillment of
requirement for the award of the degree of
Bachelor of Electrical Engineering (Mechatronics)

Faculty of Electrical Engineering
University of Technology, Malaysia

APRIL 2008
"I hereby declare that this thesis entitled "Development of Humanoid Robot" is the result of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any degree"

Signature

Name of supervisor: CHE NOOR HAJIZUL B CHE ANI
Date: 25 APRIL 2008
Specially dedicated from 'Jijon' to
My beloved mother, father, brother, sister, fiancé, and friends who have encouraged, guided
and inspired me throughout my journey of education
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to my project supervisor; Dr Zaharuddin B Mohamed who has persistently and determinedly assisted me during the whole course of this project. It would have been very difficult to complete this project without the enthusiastic support, insight and advice given by him.

My outmost thanks also go to my family who has given me support throughout my academic years. Without them, I might not be the person I am today.

It is to my advantage that I have received help and support from friends and staffs in the Faculty of Electrical Engineering. My appreciation to the all my friends, Khairil Amirul, Mohamad Firdaus Abu Akar, Muhamad Syamsul Fadhilah, Johan Elyas, Hanisah, Zaidi Tunari and Shazlin Elaiza for their cooperation and material aid. It is of my greatest thanks and joy that I have met these people. Thank you.
ABSTRACT

The purpose of the project is to develop a walking robot. Review of previous research has been conducted. A basic operating principle of human model has also been studied to modify and design a suitable structure of the humanoid. In this project a humanoid robot with 22 degrees of freedom has developed. Several experiments have been conducted to investigate the performance of the robot. It has been shown that the robot is capable of walking forward and turns left or right.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TITLE PAGE</td>
<td>i</td>
</tr>
<tr>
<td></td>
<td>DECLARATION</td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td>DEDICATION</td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td>ACKNOWLEDGEMENT</td>
<td>iv</td>
</tr>
<tr>
<td></td>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>ABSTRAK</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>CONTENTS</td>
<td>vii</td>
</tr>
<tr>
<td></td>
<td>LIST OF FIGURE</td>
<td>xi</td>
</tr>
<tr>
<td></td>
<td>LIST OF TABLE</td>
<td>xii</td>
</tr>
<tr>
<td></td>
<td>LIST OF APPENDIX</td>
<td>xii</td>
</tr>
</tbody>
</table>

1      INTRODUCTION       1
1.1 Robot   1
1.2 Project Overview  3
1.3 Objectives  3
1.4 Scope of Works  3
1.5 Problem Statement  4
1.6 Summary  4

2      LITERATURE REVIEW  5
2.1 Project Background  5
2.2 Previous Research  6
3.4.5 Main Board
3.4.5.1 USART
USART Library
Boot Loader
3.5 Summary

4  HAMIZON (RESULTS AND DISCUSSIONS) 47
4.1 Specifications 47
4.2 Walking Gait 54
   4.2.1 Forward Gait 54
   4.2.2 Right Turn Gait 54
   4.2.1 Left Turn Gait 55
4.3 Summary 55

5  CONCLUSIONS AND RECOMMENDATIONS 56
5.1 Accomplishments 56
5.2 Future Development Suggestions 57
   5.2.1 Upgrade Hamizon Locomotion 57
   5.2.2 Incorporate With Sensor 58
5.3 Summary 58
<table>
<thead>
<tr>
<th>CHAPTER TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCE</td>
<td>60</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>61</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Asimo</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Open PINO</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Projection of the centre of mass point</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Static Walking</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>Dynamic Walking</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Design Phase</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Schematic for whole circuit</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Schematic for microcontroller supported circuit</td>
<td>19</td>
</tr>
<tr>
<td>3.4</td>
<td>Schematic of power supply circuit</td>
<td>20</td>
</tr>
<tr>
<td>3.5</td>
<td>Location of D.O.F</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>Circuit Block Diagram</td>
<td>22</td>
</tr>
<tr>
<td>3.7</td>
<td>Example of the Frame</td>
<td>24</td>
</tr>
<tr>
<td>3.8</td>
<td>VRLA Power Cell</td>
<td>25</td>
</tr>
<tr>
<td>3.9</td>
<td>Servo Motor</td>
<td>27</td>
</tr>
<tr>
<td>3.10</td>
<td>Servo motor that being used in this project</td>
<td>27</td>
</tr>
<tr>
<td>3.11</td>
<td>Servo motor that being used in this project</td>
<td>27</td>
</tr>
<tr>
<td>3.12</td>
<td>Servo Controller SSC-32</td>
<td>28</td>
</tr>
<tr>
<td>3.13</td>
<td>Main Board Circuit (PCB)</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Front view</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Right view</td>
<td>49</td>
</tr>
<tr>
<td>4.3</td>
<td>Left view</td>
<td>49</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.4</td>
<td>Bottom view</td>
<td>50</td>
</tr>
<tr>
<td>4.5</td>
<td>Top view</td>
<td>50</td>
</tr>
<tr>
<td>4.6</td>
<td>Servo location</td>
<td>51</td>
</tr>
<tr>
<td>4.7</td>
<td>Forward Gait</td>
<td>54</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Location of D.O.F</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Servo motor Specification</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>Hamizon Specifications</td>
<td>52</td>
</tr>
<tr>
<td>4.2</td>
<td>Budget of Major Component</td>
<td>53</td>
</tr>
</tbody>
</table>
# LIST OF APPENDIX

<table>
<thead>
<tr>
<th>APPENDIX</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Final Set Of programming</td>
<td>60</td>
</tr>
<tr>
<td>B</td>
<td>HAMIZON</td>
<td>68</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Humanoid robot research has produced robot capable of robust locomotion wide range of human like movement. The tasks are to move toward equipping humanoid robot systems with cognitive capabilities such as object recognition, scene segmentation and avoid the obstacle. High performance in mechanical components as well as in the controlling software is required since the robot should be able to operate in a dynamically changing environment.

1.1 Robot

Robots in general have many definitions. Therefore, it is hard to define the definitions of robot accurately. As we know, The Robotic Institute of America on the others hand define robot as:

“Reprogrammable multifunctional manipulator designed to move material, parts, and tools or specialized through variable programmed motion for the performance of variety or tasks’.”
Online dictionary, whatis.com defines robots as:

“A robot is a machine designed to execute one or more tasks repeatedly, with speed and precision.” [www.whatis.com].

So, robot will be simplify as the art, knowledge base, and the know-how of designing, applying, and using robots in human endeavors. Robotic system not consist just robots, but also other devices and systems that are used together with the robots to perform the necessary tasks or intelligent of robots.

In any capacity, robots can be useful, but need to be programmed and controlled. Robotics is an interdisciplinary subject that benefits from mechanical engineering, electrical and electronic engineering, computers implementation and environmental observation.

There are as many different types of robots as there are tasks for them to perform. A human operator, sometimes from a great distance, can control a robot. But most robots are controlled by computer, and fall into either of two categories:

Autonomous robots and insect robots. An autonomous robot acts as a stand-alone system, complete with its own computer (called the controller). Insect robots work in fleets ranging in number from a few to thousands, with all fleet members under the supervision of a single controller. The term insect arises from the similarity of the system to a colony of insects, where the individuals are simple but the fleet as a whole can be sophisticated.
1.2 Project Overview

This project involves the design and fabrication of a humanoid robot. The frame of this robot will be design based on the Almir’s Robot specification. However the main control circuit and programming will be different from the Almir’s Robot. The robot will have 22 degrees of freedom (D.O.F) in total maneuvering system. A vast number of servo motors will be used in the robot to provide sufficient D.O.F.

1.3 Objectives

The main objective of this project is to design and build a small size humanoid robot that has the ability to walk forward, turn both side and sit down (humanlike geometry and motion).

1.4 Scope of Work

In order to achieve the objective of the project, several scopes have been outlined. The first one is to built and programs a robot that can move forward and can turn both sides. Then design circuit and develop software to control its movement. And for an addition is to built hand that can hold light object.
1.5 Problem Statement

Problem that I have found while preparing this project are; How to achieve stability during walking; How to mimic human pattern of walking; How to program the movement.

1.6 Summary

This thesis discusses the construction of a Humanoid Robot. First chapter of this thesis discusses on the introduction of the project. Second chapter discusses on the literature review and chapter three focuses on the robot design. Chapter four present the result.
CHAPTER 2

LITERATURE REVIEW

This chapter focuses on the related field and knowledge pertaining to the accomplishment of the thesis. Reference books, papers, journal articles, websites, conference articles and documentations regarding applications and research works have been obtained and studied.

2.1 Project Background

As the world advances and technology becomes more complicated, the need to create an artificial human is complex than ever. Humanoid robots are robot with they overall appearance based on that of the human body. In general humanoid robots have a torso with head, a pair of arms and a pair of legs. On the other hand, some forms of humanoid robot may model only in certain part of the body, for example, bipedal robot. Some human robot may also have ‘face’, with ‘eyes’ and ‘mouth’. Android are humanoid robots built to resemble male human, and Ganoids are humanoid built to resemble female humans.
Just like a human, a humanoid has component and parts to replace the function of human organ. For example the microcontroller in the main controller circuit will serve to function as the brain and the servo motor will serve to function as a muscle, while the cable and wires which connect all the components will serve to function as the nervous system. Understanding the human body structure and behaviors (biomechanics) are a very important process before one starts building a humanoid robot. On the other hand, the attempt to simulate the human body leads to a better understanding of it and helps to create better tools or equipment for human use.

2.2 Previous Research

2.2.1 WASEDA

The Humanoid Robotics Institute of the Waseda University is the first group to build hardware humanoid robot. Their current Wabian series stands in the long tradition of the Wabot Project that started in 1968 and produced the first humanoid in 1970. A multitude of different versions of the Wabot and Wabian series have been released since the examining different kinds of actuators, especially various type of artificial muscles, and novel control method. The humanoid research has since the beginning been accompanied by research intelligence prosthesis. By joining these related research fields, synergies have been exploited by exchanging biological and robotics know-how.

The latest model WABIAN-RV with 43 actuated and 8 passive joints is currently among the most complex humanoid. Its disposes of advance visual and auditory sense to mimic the capabilities of the human sensory system. Based on the sensing information, the whole body-motion of WABIAN-RV is generated online. For this purpose the dynamics of the robot are reduced by just describing the knee
trajectory. This method slows robot gait with stretched knee which is not possible with most others methods due to the configuration singularity. Stable walking is realized by stiffness/compliance control.

For memorizing and performing complex walking patterns systematically and effectively, a method for online tech-in of low-level and high-level has been developed using speech recognition system. The robot can emotionally interact with humans by showing feelings like happiness, sadness and anger.

2.2.2 HONDA ASIMO

![Asimo](image)

**Figure 2.1:** Asimo

Started in 1986, the success of the Honda humanoid series P2, P3 and Asimo aroused a widespread interest both in public and in the research community. Figure 2.1
shows the Asimo developed by Honda. Conceived as a human size universal helper, the latest model Asimo shows one of the most advanced walking technologies; unfortunately little information on the technology is published. Asimo disposes of set of recalculated trajectories. If these trajectories do not match the requirement, new motions are generated online by interpolating between two closely matching pattern. The resulting motion is further refined adapting the dynamic properties base on estimating future behavior. The walking controller shows very fast and dynamic walking performance.

Asimo also sets standards in human-robot-interaction. It can recognize individual people by their face and thus react specifically to the person. Hence, the Honda robot has successfully performed reception or guiding task on trade fair or representation.

2.2.3 SONY QRio

Another robot developed in industries is the Sony QRio. As opposed to Asimo, the concept of this small size robot does not follow the idea of the universal helper, but is rather designed as an entertainment platform. With this perspective, development focus on side on creating an inexpensive platform suitable for the mass market and on the other side on implementing entertaining performances like gymnastic shows, dancing synchronized between several robots or singing. QRio has a highly advanced equilibrium system allowing it to balance on moving surfboard or skating on roller skates.

A special tool has been created that allows people with little technical background knowledge to graphically tech-in new motion patterns. Based on the
Dynamics Filter method, balanced motion patterns are generated from the teach in specifications that can be transferred to the robot. This system even account for external reaction forces, thus QRio can be taught to grab a ball and throw it away.

### 2.2.4 Open PINO

![Open PINO](image)

**Figure 2.2:** Open PINO

Figure 2.2 shows the Open PINO. Open PINO was started in November 1999. The concepts of the PINO project were to develop a platform for perception and behavior research using multiple perception channels and high degree of freedom. To investigate robot design that is well received by general public. To develop an affordable humanoid platform using off-the-shelf component and low-precision material
Open INO or PINO class platform is an attempt to create Linux-like open source development community by disclosing its technical information based on PINO. PINO project software is disclosed under GPL (GNU General Public License) and its electric circuits and mechanical design diagram are released under GNU Free Document License. However PINO Exterior aesthetic design is proprietary property and do not serve as a subject of open source release.

PINO is a simple design and has many opportunities for advancements. It is only intended to be a minimum platform that serves as a starting point of the collective efforts. Just like when Linux disclosed its initial Linux kernel, it was a collective effort of many people who were interested, that contributed to the formation of current Linux system. PINO is the first attempt in robotics that tries to evolve through open source movement and it is the creator’s wish that this initiative contributes to the endorsement of scientific research and triggers faster escalation of the industry.

2.3 Theory

Servo modules or servo motors are the higher basic architecture of this bipedal robot and each motor will generate one D.O.F for the robot. In order to build a robot to perform various kinds of behaviors like a human, high number of D.O.F is prerequisite.

For a robot to function properly it needs different kind of sensors to send feedback to the main system. In a servo module, DC motor, potential meter and a set of gear are included. The function of the potential meter is to send position feedback to the servo controller and receive. Once the feedback is process the servo controller will send new instruction back to the motor until the module reach its desired position.
When all the servo modules group together to form a set of system, it can perform various task and behaviors. These behaviors including swinging the leg, balancing and walking. According to the Almir’s research, a good design of leg needs six degree of freedom to move the swing leg to its arbitrary location. Therefore, the design of my Humanoid leg will have six degree of freedom for each leg as well. By doing this robot can achieve better and smoother walking pattern.

Forces sensor is added to the system in order to detect if the leg has make proper contact with the ground or it is hanging in the air. This is very important to ensure that the robot will not lift the other leg before the first legs reach the ground. If the robot fails to create this condition, the robots will loss control from the center of gravity and fall.

Furthermore, adding a posture or gyro sensor will enable the system to detect the robot’s body balance. The robot will be able to detect if it is standing on an even surface or leaning surface and make necessary alternation to maintain balance.

2.4 Humanoid Robot Locomotion

This section will mainly rely on [Vukobratovi´c et al., 2006] and [Azevedo et al., 2005] who have made a contribution toward a unification in the area of humanoid robots.

Walk- In the area of humanoid locomotion walk is defined as: Movement by putting forward each foot in turn, not having both feet off the ground at once [Vukobratovi´c et al., 2006].

Gait -The term gait and walk is not the same, gait refers to the manner of walking.
Hence when a humanoids walk, the can have different gaits. If a vector, $\theta(t)$ is defined to contain all the joint angles, then a time history of $\theta(t)$ represents the specific gait.

**Step**-A step is defined as: *in the direction of motion, during the contact with the ground, the leg from the front position with respect to the trunk comes to the rear position, then it is deployed from the ground and in the transfer phase moves to the front position, to make again contact with the ground, and the cycle is repeated* [Vukobratović et al., 2006]. Note that the duration of a step runs from the foot have a certain position until it reaches that position again.

**Support Phases**- A step can be divided into a large number of phases but must at least contain the following two: a single support phase (SSP) where only one foot is in contact with ground, and a double support phase (DSP) where both feet are in contact with ground. The SSP can be divided into two different phases, namely the left single support phase (SSP-L), where the left leg is the supporting leg, and the right single support phase, where the right leg is the supporting leg. SSP-L and SSP-R can be further divided into the lift-off phase and the impact phase. The impact phase starts when the heel of the rear foot leaves the ground and ends when the toe leaves the ground, after which the SSP starts. The impact phase is when the heel of the front foot hits the ground.

Providing a bipedal robot with a reliable gait is far from trivial. The robot should be able to walk fast and not consume too much energy. At the same time the amount of computations needed to provide a stable gait should be minimized since the limited onboard processing power is needed for the interaction between the robot and its environment.

Even though it is more difficult to implement algorithms for reliable locomotion in bipedal robots, compared to e.g. wheeled robots or perhaps multi-legged ones, there are still good reasons for developing walking robots. First, bipedal robots have the advantage of being able to move in areas that are normally inaccessible to wheeled
robots, such as stairs and areas with plenty of obstacles. Second, if the bipedal robot is a full-scale humanoid, it can easily function in areas designed for humans such as houses, factories etc, since its humanlike shape would allow it to reach shelves etc.

The bipedal locomotion consists of two phases; the single support phase and the double support phase (see figure 3.3). During the single support phase only one foot is on the ground while the other is being transferred from back to front position. The double support phase occurs when the swing foot reaches the ground.

![Figure 2.3: Projection of the centre of mass point above the supporting polygon in single support phase and double support phase](image)

2.5 Bipedal Walking Robots

Research into bipedal walking robots can be split into two categories: *active* and *passive*. The passive or un-powered category (for example, McGeer’s passive dynamic walker [McGeer, 1990]) is of interest as it illustrates that walking is fundamentally a dynamic problem. Passive walkers do not require actuators, sensors, or computers in
order to make them move, but walk down gentle slopes generating motion by the
hardware geometry. The passive walkers also illustrate the walking can be performed
with very little power input.

Active walkers can further be split into two categories; those that employ the
natural dynamics of specialized actuators, and those that are fully power operated.
Raibert [Raibert, 1986] and later Pratt [Pratt, 1998] have shown some impressive feats
of walking and gymnastic ability in robots that have the capacity for energy storage in
the actuator. These robots have been shown to have robust and stable performance from
relatively simple control mechanisms.

2.5.1 Static Walking

Static walking assumes that the robot is statically stable. This mean that, at any
time, if all the motion is stop the robot will stay indefinitely in stable position. It is
necessary that the projection of the center of gravity of the robot on the ground must be
contained within the foot support area. The support area is either the foot surface in case
of one supporting leg or the minimum convex area containing both foot surfaces in case
both feet on the ground. These are referred to as single and double support phases,
respectively. Also, walking speed must be low so that the inertial forces are negligible.
This kind of walking requires large feet; strong ankle joints and can achieve only slow
walking speeds. It has been abandoned by most researchers for dynamic walking, which
provides more realistic and agile movements.
2.5.2 Dynamic Walking

Biped dynamic walking allows the center of gravity to be outside the support region for limited amount of time. There is no absolute criterion that determines whether the dynamic walking is stable or not. Indeed a walker can be designed to recover from different kinds of instabilities. However, if the robot has active ankle joints and always keeps at least one foot at on the ground then the Zero Moment Point (ZMP) can be used as stability criterion. The ZMP is the point where the robot’s total moment at the ground is zero. As long the ZMP is inside the support region the walking is considered dynamically stable because is the only case where the foot can control the robot’ posture. It is clear that for robots that do not continuously keep at least one foot on the ground or that do not have active ankle joints (walking on stilts), the notion of support are do not exist, therefore the ZMP criterion cannot applied. Dynamic walking achieved by ensuring that the robot is always rotating around a point in the support region. If the robot rotates around a point outside the support region then this means that the supporting foot will tend to get off the ground or get presses against the ground. Both cases lead to instability. To draw an analogy with the static a walking, if all motion is stopped the robot will tend to rotate around the ZMP.

Figure 2.4:  Static Walking
2.6 Summary

Succinctly, this chapter explains the background of the project where a humanoid is presented. Then, it continues on previous research. Next, the theory behind humanoid was explained. The robot builds by the several servo modules and has a main controller integrating others subsystem such as sensors in the robot. This chapter explains on others humanoids such as ASIMO, SONY QRio, WASEDA and OPEN PINO. There are two different type of bipedal walking; static walking and dynamic walking and dynamic has lot of advantage compare to the static walking.
CHAPTER 3

ROBOT DESIGN

This chapter discusses about the design process of Humanoid robot. Generally, the design processes are divided into two phases: the Electronic Design, and Mechanical Design. Each phase is explained accordingly to method and material selection factors. Figure 3.1 shows the design phase.

![Figure 3.1: Design Phase](image_url)
3.1 Electronic Design

The electronic design for this humanoid robot consists of microcontroller circuit, power supply circuit and interfacing circuit. Circuit Design is a process of planning, attachment of components and implementation of circuit to gather output based on given input. Planning a circuit among others to identify the circuit’s specification, components involved, and ensure the effectiveness the implementation and troubleshooting.

The best components selection is required to ensure the success of this project. Lastly, the implementation stage is the most crucial step in circuit design.

The fabrication of circuit board must be suitable for the body of this robot. The body of this robot is small, compact and light weight. Figure 3.2 shows the schematic circuit.

![Figure 3.2: Schematic for whole circuit](image-url)
3.1.1 The Microcontroller Circuit Design

The microcontroller is a circuit that controls every activities of the robot. The function of this circuit is to activate, protect and stabilize the I/O PIC18F452 signals. This supported circuit does not require oscillator circuit because PIC18F452 is equipped with built in oscillator.

Therefore, smaller circuit in dimension can be attached to the robot structure. The I/O signals are received and sent to microcontroller through this circuit. Without this supported circuit the microcontroller is dysfunctional. A 10 k Ohm resistor is used to limit the incoming current into PIC18F452 while a capacitor is used to stabilize the voltage that flows into PIC18F452. A port B is used as a pin of I/O microcontroller. Figure 3.3 shows the Schematic for microcontroller supported circuit.

Figure 3.3: Schematic for microcontroller supported circuit
3.1.2 Power Supply Circuit

Power Supply Circuit is a circuit that supplying electric power to all circuits including servo motor and sensor. The main function of this circuit is to arrange the input of 9V voltage battery to two level of output voltage which is the 5V and 6V. The 5V output is supplying power to sensor circuit and the 6V output is supplying power to motor servo circuit. The voltage arranger type L7805 and L7806 were used to produce output voltage of 5V and 6V power supply respectively. The capacitor 0.1uF is used to stabilize the input and output voltage. Two LEDs are used as indicators to show both input and output. The LED will be lighted if the output needed drain out from resource. As big current flows through the LEDs, a 330 Ohm transistor is used to limit the current. Figure 3.4 shows the Schematic of power supply circuit.

![Schematic of power supply circuit](image)

**Figure 3.4**: Schematic of power supply circuit
3.2 Mechanical Design

This Humanoid robot consist 22 degree of freedom. Each degree of freedom will be representing by the servo motor to control the revolute movement. Each servo receives a Pulse-width modulated signal from the servo controller. Figure 3.5 shows the location of the degree of freedom located in the robot. There will be three degree of freedom at the hip at each leg, one degree of freedom located at the knee for each leg and two degree of freedom located at the ankle of each leg. All this twelve degree of freedom will enable the robot to walk and turn both sides.

There are three degree of freedom at each shoulder and one for the each elbow. For the neck it will have two degree of freedom. Table 3.1 shows the location of D.O.F

<table>
<thead>
<tr>
<th>Location</th>
<th>Center</th>
<th>Left</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neck</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hand</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shoulder</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Elbow</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Leg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hip</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Knee</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ankle</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 3.1: Location of D.O.F**
Figure 3.5: Location of D.O.F

3.3 Operation

Figure 3.6 shows a block diagram representing the electronic circuit and its relation to the humanoid robot.
From the block diagram it is clearly illustrate that all the servo module are control by a servo controller and the servo controller receive instruction from the main controller. When the signal is receive from the servo controller the DC motor in the servo module is responsible to generate torque to spin the gear and create motion for the robot to move. On the other hand, the potentiometer sends it position feedback to the servo controller where it will tell the servo controller if it has reaches the desire angle.

If a set of force sensor array is added in the robot system, it will be responsible to send a sign feedback to the main controller if it has detected any force had applied on it. On the other hand by adding a posture sensor in the sensory system will enable the
robot to detect if its structure is not in the desire alignment and send feedback to the main controller.

3.4 Component

3.4.1 Frame

Figure 3.7: Example of the Frame

Figure 3.7 shows the example of the frame. The frame of the body will be manufactured using aluminum sheet and aluminum bar. The function of the frame is to provide protection and mounting for the robot circuitry and servo module. Another function of the frame is to serve as the backbone for the robot and act like human bones.
3.4.2 Power Supply

This robot will use 6V/4.5Ah VRLA power cell and 9V power cell to provide sufficient power to controller circuit and servo module.

3.4.2.1 Valve Regulated Lead Acid Power Cell

![VRLA Power Cell](image)

**Figure 3.8**: VRLA Power Cell

Figure 3.8 shows the example of the VRLA Power Cell. Valve Regulated Lead Acid rechargeable batteries are designed to provide outstanding performance in withstanding overcharge, over discharge, and resisting vibration and shock. Compact, these batteries save installation space, while providing full and reliable power. The use of special sealing epoxies, tongue and groove case and cover construction, and long-sealing path for posts and connectors assures that the Valve Regulated Lead Acid (VRLA) battery will offer exceptional leak resistance, and allows them to be used in any position.
The sealed VRLA battery is closed system in which the quantity of electrolyte is limited, but the gases resulting from energy production are recombined to be reused instead of being released to atmosphere. Because the electrolyte is recycled, there is no need to add water to the batteries. Excess gas pressure that may build up inside the battery is released through a simple regulating valve to ensure that structural integrity is not compromised. Each cell contains (in the charged state) electrodes of lead metal (Pb) and lead (IV) oxide (PbO₂) in an electrolyte of about 37% (5.99 Molar) w/w sulfuric acid (H₂SO₄). In the discharged state both electrodes turn into lead (II) sulfate (PbSO₄) and the electrolyte loses its dissolved sulfuric acid and becomes primarily water. Due to the freezing-point depression of water, as the battery discharges and the concentration of sulfuric acid decreases, the electrolyte is more likely to freeze.

3.4.3 Servo Motor

There are actually two types of servomotors available. The DC servomotor illustrated at the right in Figure 3.8 is for industrial use and are much heavier, powerful and expensive. It is sometime used in building bigger robot. The other type of servomotor calls pager type because it looks like a pager. Pager type servomotor, or simply called a servomotor are used widely in building robots.

The nature of the servo motor, wide selection of torque, low weight, cartridge type casing, precise positioning of the servo and robust, make it an obvious choice for the biped robot. The limitations of servomotors are its price which is a bit expensive and that most of the model can’t be obtained in Malaysia.

Basically the physical characteristic of the digital servo is pretty much the same as conventional analogue servo. The difference is that the digital servo has a faster
response time and consumes more power. It is not feasible to use a digital servomotor in this project because of the fast response, high power needed and it is expensive. The model and specifications of the servomotor used in this project will be discussed in detail in the following chapter. Figure 3.9 shows the example of Servo Motor while Figure 3.10 and Figure 3.11 shows the Servo motor that being used in this project. Table 3.2 shows the servo motor specification.

Figure 3.9: Servo Motor

Figure 3.10: Servo motor that being used in this project (RBS5802)

Figure 3.11: Servo motor that being used in this project (RBS5802)
### Table 3.2: Servo motor Specification

<table>
<thead>
<tr>
<th>TYPE</th>
<th>RBS 5802</th>
<th>RBS 582</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION</td>
<td>Provide rotation function in specific range</td>
<td>Provide rotation function in specific range</td>
</tr>
<tr>
<td></td>
<td>(for the lower body)</td>
<td>(for the upper body)</td>
</tr>
<tr>
<td>FEATURES</td>
<td>Speed - 0.13s/60°</td>
<td>Speed - 0.11s/60°</td>
</tr>
<tr>
<td></td>
<td>Torque- 17kg · cm</td>
<td>Torque- 9.2kg · cm</td>
</tr>
</tbody>
</table>

#### 3.4.4 Servo Controller

The bigger robot, the more servos and I/O are required to enable them to fulfill their duties. The Lynxmotion Servo Controller SSC-32, control up to 32 servos using a single I/O line. The SSC-32 manages all the servo pulse so the host controller does not have to.

#### 3.4.4.1 Servo Controller SSC-32

![Figure 3.12: Servo Controller SSC-32](image)
Figure 3.12 shows the example of Servo Controller SSC-32. Pulse-proportional servos are designed for use in radio-controlled (R/C) cars, boats and planes. They provide precise control for steering, throttle, rudder, etc. using a signal that is easy to transmit and receive. The signal consists of positive going pulses ranging from 0.9 to 2.1mS (milliseconds) long, repeated 50 times a second (every 20mS). The servo positions its output shaft in proportion to the width of the pulse, as shown below.

In radio-control applications, a servo needs no more than a 90° range of motion, since it is usually driving a crank mechanism that can't move more than 90°. So when you send pulses within the manufacturer-specified range of 0.9 to 2.1mS, you get around 90° range of motion.

Most servos have more than 90° of mechanical range. In fact, most servos can move up to 180° of rotation. However, some servos can be damaged when commanded past their mechanical limitations. The SSC-32 lets you use this extra range. A position value of 500 corresponds to 0.50mS pulse, and a position value of 2500 corresponds to a 2.50mS pulse. A one unit change in position value produces a 1uS (microsecond) change in pulse width. The positioning resolution is 0.09°/unit (180°/2000). From here on, the term pulse width and position are the same.
3.4.4.2 Command Formatting for the SSC-32

Command Types and Groups

1) Servo Movement.  7) Read Analog Inputs.
2) Discrete Output.  8) 12 Servo Hexapod Gait Sequencer.
3) Byte Output.  9) Query Hex Sequencer.
4) Query Movement Status. 10) Get Version.
5) Query Pulse Width. 11) Go To Boot.

With the exception of MiniSSC-II mode, all SSC-32 commands must end with a carriage return character (ASCII 13). Multiple commands of the same type can be issued simultaneously in a Command Group. All of the commands in a command group
will be executed after the final carriage return is received. Commands of different types cannot be mixed in the same command group. In addition, numeric arguments to all SSC-32 commands must be ASCII strings of decimal numbers, e.g. "1234". Some commands accept negative numbers, e.g. "-5678". Programming examples will be provided. ASCII format is not case sensitive. Use as many bytes as required. Spaces, tabs, and line feeds are ignored.

**Servo Move or Group Move:**

```
# <ch> P <pw> S <spd>... # <ch> P <pw> S <spd> T <time><cr>
```

<ch> =Channel number in decimal, 0 - 31.
<pw> =Pulse width in microseconds, 500 - 2500.
<spd> =Movement speed in uS per second for one channel. (Optional)
<time> =Time in mS for the entire move, affects all channels, 65535 max. (Optional)
<cr> =Carriage return character, ASCII 13. (Required to initiate action)
<esc> =Cancel the current command, ASCII 27.

Servo Move Example: "#5 P1600 S750 <cr>"

The example will move the servo on channel 5 to position 1600. It will move from its current position at a rate of 750uS per second until it reaches its commanded destination. For a better understanding of the speed argument consider that 1000uS of travel will result in around 90° of rotation. A speed value of 100uS per second means the servo will take 10 seconds to move 90°. Alternatively a speed value of 2000uS per second equates to 500mS (half a second) to move 90°.

Servo Move Example: "#5 P1600 T1000 <cr>"

The example will move servo 5 to position 1600. It will take 1 second to complete the move regardless of how far the servo has to travel to reach the destination.
Servo Group Move Example: "#5 P1600 #10 P750 T2500 <cr>"

The example will move servo 5 to position 1600 and servo 10 to position 750. It will take 2.5 seconds to complete the move, even if one servo has farther to travel than another. The servos will both start and stop moving at the same time. This is a very powerful command. By commanding all of the legs in a walking robot with the Group Move it is easy to synchronize complex gaits. The same synchronized motion can benefit the control of a robotic arm well.

You can combine the speed and time commands if desired. The speed for each servo will be calculated according to the following rules:

1. All channels will start and end the move simultaneously.
2. If a speed is specified for a servo, it will not move any faster than the speed specified (But it might move slower if the time command requires).
3. If a time is specified for the move, then the move will take at least the amount of time specified (but might take longer if the speed command requires).

Servo Move Example: "#5 P1600 #17 P750 S500 #2 P2250 T2000 <cr>"

The example provides 1600uS on ch5, 750uS on ch17, and 2250uS on ch2. The entire move will take at least 2 seconds, but ch17 will not move faster than 500uS per second. The actual time for the move will depend on the initial pulse width for ch17. Suppose ch17 starts at position 2000. Then it has to move 1250uS. Since it is limited to 500uS per second, it will require at least 2.5 seconds, so the entire move will take 2.5 seconds. On the other hand, if ch17 starts at position 1000, it only needs to move 250uS, which it can do in 0.5 seconds, so the entire move will take 2 seconds.
Important! Don't issue a speed or time command to the servo controller as the first instruction. It will assume it needs to start at 500\textmu S and will zip there as quickly as possible. The first positioning command should be a normal "# <ch> P <pw>" command. Because the controller doesn't know where the servo is positioned on power up it has to be this way.

**Pulse Offset:**

# <ch>PO <offset value> … # <ch> PO <offset value> <cr>

<ch>=Channel number in decimal, 0 - 31.
<offset value> = 100 to -100 in uSeconds.
<cr>=Carriage return character, ASCII 13.

The servo channel will be offset by the amount indicated in offset value. This makes it easy to setup legs in a robot that do not allow mechanical calibration.

**Discrete Output:**

# <ch><lvl> ... # <ch> <lvl><cr>

<ch>=Channel number in decimal, 0 - 31.
<lvl>=Logic level for the channel, either 'H' for High or 'L' for Low.
<cr>=Carriage return character, ASCII 13.

The channel will go to the level indicated within 20mS of receiving the carriage return.

Discrete Output Example: "#3H #4L <cr>"

This example will output a High (+5v) on channel 3 and a Low (0v) on channel 4.
Byte Output:

# <bank> : <value><cr>

<Bank>=(0 = Pins 0-7, 1 = Pins 8-15, 2 = Pins 16-23, 3 = Pins 24-31.)
<Value>=Decimal value to output to the selected bank (0-255). Bit 0 = LSB of bank.

This command allows 8 bits of binary data to be written at once. All pins of the bank are updated simultaneously. The banks will be updated within 20mS of receiving the CR.

Bank Output Example: "#3:123 <cr>"

This example will output the value 123 (decimal) to bank 3. 123 (dec) = 01111011 (bin), and bank 3 is pins 24-31. So this command will output a "0" to pins 26 and 31, and will output a "1" to all other pins.

Query Movement Status:

Q <cr>

This will return a "." if the previous move is complete, or a "+" if it is still in progress. There will be a delay of 50uS to 5mS before the response is sent.

Query Pulse Width:

QP <arg><cr>

This will return a single byte (in binary format) indicating the pulse width of the selected servo with a resolution of 10uS. For example, if the pulse width is 1500uS, the returned byte would be 150 (binary).
Multiple servos may be queried in the same command. The return value will be one byte per servo. There will be a delay of least 50uS to 5mS before the response is sent. Typically the response will be started within 100uS.

**Read Digital Inputs:**

A B C D AL BL CL DL <cr>

A, B, C, or D reads the value on the input as a binary value. It returns ASCII "0" if the input is a low (0v) or an ASCII "1" if the input is a high (+5v).

AL, BL, CL, or DL returns the value on the input as an ASCII "0" if the input is a low (0v) or if it has been low since the last *L command. It returns a high (+5v) if the input is a high and never went low since the last *L command. Simply stated it will return a low if the input ever goes low. Reading the status automatically resets the latch.

The ABCD inputs have a weak pull-up (~50k) that is enabled when used as inputs. They are checked approximately every 1mS, and are denounced for approximately 15mS. The logic value for the read commands will not be changed until the input has been at the new logic level continuously for 15mS. The Read Digital Input Commands can be grouped in a single read, up to 8 values per read. They will return a string with one character per input with no spaces.

Read Digital Input Example: "A B C DL <cr>"

This example returns 4 characters with the values of A, B, C, and D-Latch. If A=0, B=1, C=1, and DL=0, the return value will be "0110".

**Read Analog Inputs:**

VA VB VC VD <cr>
VA, VB, VC, VD reads the value on the input as analog. It returns a single byte with the 8-bit (binary) value for the voltage on the pin.

When the ABCD inputs are used as analog inputs the internal pullup is disabled. The inputs are digitally filtered to reduce the effect of noise. The filtered values will settle to their final values within 8mS of a change. A return value of 0 represents 0vdc. A return value of 255 represents +4.98vdc. To convert the return values to a voltage multiply by 5/256. At power-up the ABCD inputs are configured for digital input with pullup. The first time a V* command is used; the pin will be converted to analog without pullup. The result of this first read will not return valid data.

Read Analog Input Example: "VA VB <cr>"

This example will return 2 bytes with the analog values of A and B. For example if the voltage on Pin A is 2vdc and Pin B is 3.5vdc, the return value will be the bytes 102 (binary) and 179 (binary).

12 Servo Hexapod Sequencer Commands:

LH <arg>, LM<arg>, LL <arg>
Set the value for the vertical servos on the left side of the hexapod. LH sets the high value, i.e. the pulse width to raise the leg to its maximum height; LM sets the mid value; and LL sets the low value. The valid range for the arguments is 500 to 2500uS.

RH <arg>, RM<arg>, RL <arg>
Set the value for the vertical servos on the right side of the hexapod. RH sets the high value, i.e. the pulse width to raise the leg to its maximum height; RM sets the mid value; and RL sets the low value. The valid range for the arguments is 500 to 2500uS.

VS <arg>
Sets the speed for movement of vertical servos. All vertical servo moves use this speed.
Valid range is 0 to 65535\mu S/Sec.

LF <arg>, LR <arg>
Set the value for the horizontal servos on the left side of the robot. LF sets the front value, i.e. the pulse width to move the leg to the maximum forward position; LR sets the rear value. The valid range for the arguments is 500 to 2500\mu S.

RF <arg>, RR <arg>
Set the value for the horizontal servos on the right side of the robot. RF sets the front value, i.e. the pulse width to move the leg to the maximum forward position; RR sets the rear value. The valid range for the arguments is 500 to 2500\mu S.

HT <arg>
Sets the time to move between horizontal front and rear positions. The valid range for the argument is 1 to 65535\mu S.

XL <arg>, XR <arg>
Set the travel percentage for left and right legs. The valid range is -100\% to 100\%. Negative values cause the legs on the side to move in reverse. With a value of 100\%, the legs will move between the front and rear positions. Lower values cause the travel to be proportionally less, but always centered. The speed for horizontal moves is adjusted based on the XL and XR commands, so the move time remains the same.

XS <arg>
Set the horizontal speed percentage for all legs. The valid range is 0\% to 200\%. With a value of 100\%, the horizontal travel time will be the value programmed using the HT command. Higher values proportionally reduce the travel time; lower values increase it. A value of 0 will stop the robot in place. The hex sequencer will not be started until the XS command is received.

XSTOP
Stop the hex sequencer. Return all servos to normal operation.

Notes on Hex Sequencer:

1) The following servo channels are used for the Hex Sequencer

<table>
<thead>
<tr>
<th>Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Right Rear Vertical</td>
</tr>
<tr>
<td>1</td>
<td>Right Rear Horizontal</td>
</tr>
<tr>
<td>2</td>
<td>Right Center Vertical</td>
</tr>
<tr>
<td>3</td>
<td>Right Center Horizontal</td>
</tr>
<tr>
<td>4</td>
<td>Right Front Vertical</td>
</tr>
<tr>
<td>5</td>
<td>Right Front Horizontal</td>
</tr>
<tr>
<td>6</td>
<td>Left Rear Vertical</td>
</tr>
<tr>
<td>7</td>
<td>Left Rear Horizontal</td>
</tr>
<tr>
<td>8</td>
<td>Left Center Vertical</td>
</tr>
<tr>
<td>9</td>
<td>Left Center Horizontal</td>
</tr>
<tr>
<td>10</td>
<td>Left Front Vertical</td>
</tr>
<tr>
<td>11</td>
<td>Left Front Horizontal</td>
</tr>
</tbody>
</table>

2) The Hexapod walking gait is an alternating tripod. The tripods are labeled Tripod A and Tripod B. Tripod A consists of {Left Front, Left Rear, Right Center}, and Tripod B consists of {Left Center, Right Front, Right Rear}.

3) While walking, the legs pass through 6 points: (Low Front), (Low Center), (Low Rear), (Mid Rear), (High Center), and (Mid Front). “Center” refers to the mid-point between the Front and Rear pulse widths.

4) The walking sequence consists of 8 states, numbered 0 –7. They are defined below:

<table>
<thead>
<tr>
<th>State</th>
<th>Tripod A</th>
<th>Tripod B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vertical</td>
<td>Horizontal</td>
</tr>
<tr>
<td>0</td>
<td>Low</td>
<td>Front to Center</td>
</tr>
<tr>
<td>1</td>
<td>Low</td>
<td>Center to Rear</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>Rear</td>
</tr>
<tr>
<td>3</td>
<td>Low to Mid</td>
<td>Rear</td>
</tr>
<tr>
<td>4</td>
<td>Mid to High</td>
<td>Rear to Center</td>
</tr>
<tr>
<td>5</td>
<td>High to Mid</td>
<td>Center to Front</td>
</tr>
<tr>
<td>6</td>
<td>Mid to Low</td>
<td>Front</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>Front</td>
</tr>
</tbody>
</table>

In this table, “Front” and “Rear” are modified by the XL and XR commands. A value of 100% results in the movement in the table. Between 0 and 100%, the Front/Rear positions are moved closer to Center. For negative values, Front and Rear are
exchanged. For example, with an XL of -100%, in State 0, Tripod A on the left side would be moving Rear to Center, and Tripod B would be moving Front to Center.

5) When a horizontal servo is moving, its speed will be adjusted based on the Front/Rear pulse widths, the XL/XR percentage, and the XS percentage. Regardless of the travel distance from front to rear (adjusted by XL/XR), the total move time will be the HT divided by the XS percentage.

6) When a vertical servo is moving from Low to Mid or from Mid to Low, it will move at the speed specified by the VS command. When a vertical servo is moving from Mid to High or High to Mid, the vertical speed will be adjusted so that the horizontal and vertical movements end at the same time.

7) Any of the Hex Sequencer commands can be issued while the sequencer is operating. They will take effect immediately.

Hex Sequencer Examples:
“LH1000 LM1400 LL1800 RH2000 RM1600 RL1200 VS3000 <cr>”
Sets the vertical servo parameters.
“LF1700 LR1300 RF1300 RR1700 HT1500 <cr>”
Sets the horizontal servo parameters.
“XL50 XR100 XS100 <cr>”
Causes the gradual left turn at 100% speed (and starts the sequencer if it is not already started).

“XL -100 XR 100 XS 50 <cr>”
Causes a left rotate in place at 50% speed.

“XSTOP <cr>”
Stops the sequencer and allows servo channels 0-5, 16-21 to be controlled using the normal servo movement commands.

**Query Hex Sequencer State:**

XQ <vr>
Returns 1 digit representing the state of the hex sequencer, and the approximate percentage of movement in the state. The high nibble will be ‘0’ to ‘7’, and the low nibble will be ‘0’ to ‘9’. For example, if the sequencer is 80% of the way through state 5, it will return the value 58 hex.

**Get Software Version:**

VER <cr>
Returns the software version number as an ASCII string.

**Transfer to Boot:**

GOBOOT <cr>
Starts the bootloader running for software updates. To exit the bootloader and start running the application, power cycles the controller or enter (case sensitive, no spaces).

g0000<cr>

**SSC Emulation:** Binary format, 3-bytes.
Byte 1: 255, the sync byte
Byte 2: 0 - 31, the servo number
Byte 3: 0 - 250, the pulse width, 0=500uS, 125=1500uS, 250=2500uS
3.4.5 Main Board

The main board or the main controller in Figure 3.13 is the one build in form of printed circuit board. The circuit was design and developed using Proteus Professional ISIS 6. The main controller board was designed to work with a wide range of Microchip PIC Microcontroller and compatible with most of the PIC18 and PIC16 series. There is a reset button on top of the microcontroller, 2 pin to connect with the servo controller, 4 pin to connect with the vision circuit and 5 pin to connect to the USB ICSP PIC Programmer.

Figure 3.13: Main Board Circuit (PCB)
The 2 pin connect to the servo controller is actually form the PIC’s RX and TX pin. This two pin will transmit the data from the main controller to the servo controller. The 5 pin is being used to program and erase the microcontroller using boot loader protocol.

3.4.5.1 USART

PIC 18F452 has built in USART and it is one of the most commonly used serial interface peripherals. It also known as the Serial Communications Interface (SCI). The most common use of the usart is to communicate to a PC serial port using the RS-232 protocol. Usart stands for Universal Synchronous Asynchronous Receiver Transmitter and its main function is to transmit or receive serial data. Its operation can be divided into two categories: synchronous and asynchronous. Synchronous operation uses a clock and data line while asynchronous operation has no separate clock accompanying the data.

USART Library

USART hardware module is available with a number of PICmicros. MikroC USART Library provides comfortable work with the Asynchronous (full duplex) mode.

You can easily communicate with other devices via RS232 protocol (for example with PC, see the figure at the end of the topic – RS232 HW connection). You need a PIC MCU with hardware integrated USART, for example PIC16F877. Then, simply use the functions listed below.
Note: USART library functions support module on PORTB, PORTC, or PORTG, and will not work with modules on other ports. Examples for PICmicros with module on other ports can be found in “Examples” in mikroC installation folder.

Library Routines

- **Usart_Init**
- **Usart_Data_Ready**
- **Usart_Read**
- **Usart_Write**

Certain PICmicros with two USART modules, such as P18F8520, require you to specify the module you want to use. Simply append the number 1 or 2 to a function name. For example, **Usart_Write2();**

**Usart_Init**

**Prototype**  
`void Usart_Init(const unsigned long baud_rate);`

**Returns**  
Nothing.

**Description**  
Initializes hardware USART module with the desired baud rate. Refer to the device data sheet for baud rates allowed for specific Fosc. If you specify the unsupported baud rate, compiler will report an error.

**Requires**  
You need PIC MCU with hardware USART.

**Usart_Init** needs to be called before using other functions from USART Library.

**Example**  
This will initialize hardware USART and establish the communication at 2400 bps:
Usart_Init(2400);

Usart_Data_Ready

Prototype unsigned short Usart_Data_Ready(void);
Returns Function returns 1 if data is ready or 0 if there is no data.
Description Use the function to test if data in receive buffer is ready for reading.
Requires USART HW module must be initialized and communication established before using this function. See Usart_Init.
Example If data is ready, read it:

    int receive;
    ...
    if (Usart_Data_Ready()) receive = Usart_Read;

Usart_Read

Prototype unsigned short Usart_Read(void);
Returns Returns the received byte. If byte is not received, returns 0.
Description Function receives a byte via USART. Use the function Usart_Data_Ready to test if data is ready first.
Requires USART HW module must be initialized and communication established before using this function. See Usart_Init.
Example If data is ready, read it:

    int receive;
    ...
    if (Usart_Data_Ready()) receive = Usart_Read();

Usart_Write

Prototype void Usart_Write(unsigned short data);
Returns  Nothing.

Description  Function transmits a byte (data) via USART.

Requires  USART HW module must be initialized and communication established before using this function. See Usart_Init.

Example  int chunk = 0x1E;
          Usart_Write(chunk);  /* send chunk via USART */

Library Example

The example demonstrates simple data exchange via USART. When PIC MCU receives data, it immediately sends the same data back. If PIC is connected to the PC (see the figure below), you can test the example from mikroC terminal for RS232 communication, menu choice Tools › Terminal.

unsigned short  i;

void  main() {

   // Initialize USART module (8 bit, 2400 baud rate, no parity bit..)
   Usart_Init(2400);

   do {
      if (Usart_Data_Ready()) {  // If data is received
          i = Usart_Read();       // Read the received data
          Usart_Write(i);        // Send data via USART
      }
   } while (1);
}

}//~!
### 3.4.5.2 Boot Loader

Among the many features built into Microchip’s Enhanced FLASH Microcontroller devices is the capability of the program memory to self-program. This very useful feature has been deliberately included to give the user the ability to perform boot loading operations. Device like PIC18F452 are designed with a designated “boot block”, a small section of protectable program memory allocated specifically for boot load firmware. The coding for the device families is slightly different however; the functionality is essentially the same. The goals of this implementation stress a maximum performance and functionality, while requiring the minimum of code space.

### 3.5 Summary

The robot is controlled by a few subsystems which serve under a main controller. There will be a servo motor controller which controls the servo motor angular position by providing a PWM signal to the servo motor. Main controller will get power supply from the 9V power cell while the servo motors get power from the VRLA power cell. Main controller will communicate with the servo controller using serial communication protocol.
CHAPTER 4

HAMIZON
(RESULTS AND DISCUSSIONS)

This chapter discussed about Hamizon specifications.

4.1 Specifications

Hamizon has 22 degrees of freedom (D.O.F) where each D.O.F has a servo motor to control its revolute movement. Each servo motor receives a Pulse-Width Modulated signal from the servo controller. Figure 4.1 till Figure 4.5 illustrate the physical measurement of Hamizon. The location of the 22 servo motors were shown in Figure 4.6 and Table 4.1 shown the Hamizon Specifications while Table 4.2 shows the budget of the major component.
Figure 4.1: Front view
Figure 4.2: Right view

Figure 4.3: Left view
Figure 4.4: Bottom view

Figure 4.5: Top view
Figure 4.6: Servo location
Table 4.1: Hamizon Specifications

<table>
<thead>
<tr>
<th>Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Height</td>
<td>35 cm</td>
</tr>
<tr>
<td>Head Height</td>
<td>4 cm</td>
</tr>
<tr>
<td>Body Height</td>
<td>12 cm</td>
</tr>
<tr>
<td>Leg Height</td>
<td>21 cm</td>
</tr>
<tr>
<td>Head Width</td>
<td>4 cm</td>
</tr>
<tr>
<td>Head Long</td>
<td>5 cm</td>
</tr>
<tr>
<td>Body Long</td>
<td>7.5 cm</td>
</tr>
<tr>
<td>Body Long</td>
<td>12.5 cm</td>
</tr>
<tr>
<td>Chest Width</td>
<td>6.5 cm</td>
</tr>
<tr>
<td>Foot Width</td>
<td>11 cm</td>
</tr>
<tr>
<td>Foot Long</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree of Freedom (D.O.F) :</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>2 DOF</td>
</tr>
<tr>
<td>Body (hands)</td>
<td>8 DOF</td>
</tr>
<tr>
<td>Leg</td>
<td>12 DOF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Servo</th>
<th>JR Robotic Servo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Body</td>
<td>RBS582</td>
</tr>
<tr>
<td>Lower Body</td>
<td>RBS5802</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Supply</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Board</td>
<td>9V</td>
</tr>
<tr>
<td>Servo Controller</td>
<td>6.0 v/4.5Ah, VRLA Power Cell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controller</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Board</td>
<td>Microchip PIC18F452</td>
</tr>
<tr>
<td>Servo Controller</td>
<td>SSC32(Lynxmotion)</td>
</tr>
</tbody>
</table>

| Chassis                | Aluminum         |

Table 4.1: Hamizon Specifications
<table>
<thead>
<tr>
<th>No</th>
<th>Item</th>
<th>Qty</th>
<th>Price(RM)</th>
<th>Total(RM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SERVO MOTOR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1) RBS 582</td>
<td>10</td>
<td>150.10</td>
<td>1501.00</td>
</tr>
<tr>
<td></td>
<td>2) RBS 5802</td>
<td>12</td>
<td>303.03</td>
<td>3636.36</td>
</tr>
<tr>
<td>2</td>
<td>SERVO CONTROLLER(SSC-32)</td>
<td>1</td>
<td>137.83</td>
<td>137.83</td>
</tr>
<tr>
<td>3</td>
<td>CAMERA (CMU2)</td>
<td>1</td>
<td>617.55</td>
<td>617.55</td>
</tr>
<tr>
<td>4</td>
<td>ALUMINUM</td>
<td></td>
<td>1000.00</td>
<td>1000.00</td>
</tr>
<tr>
<td>5</td>
<td>POWER SUPPLY</td>
<td>2</td>
<td>100.00</td>
<td>200.00</td>
</tr>
<tr>
<td>6</td>
<td>PIC</td>
<td>3</td>
<td>30.00</td>
<td>90.00</td>
</tr>
<tr>
<td>7</td>
<td>MISC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>217.26</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td></td>
<td></td>
<td>7500.00</td>
</tr>
</tbody>
</table>

**Table 4.2:** Budget of Major Component
4.2 Walking Gait

4.2.1 Forward Gait

When the forward subroutine is called, first Hamizon will active the walking formation where it will bend it knee a little to lower its center of gravity. Then it will shift its weight to the left where the whole robot will be supported by the left foot and lift up the right leg. Then Hamizon will continue by stepping it first step. As the right leg swing to the front, its left hand will swing to the front and the right will swing to the back. This to make sure Hamizon stay in stable condition. After that Hamizon will shift its weight to the right and then lift up its left leg. Finally it will step out its right leg and same as before the hands will swing but now in opposite direction. The cycle will repeat for continuous walking pattern. Figure 4.7 shows the forward gait.

Figure 4.7: Forward Gait
4.2.2 Right Turn Gait

First Hamizon will align its body and leg into turning formation. Then it will shift its weight to the left and lift up its right leg. After that Hamizon will make its first step. While the right leg reaches the floor it will spin clockwise by 30 degree. Next Hamizon will shift its weight to right and lift up its left leg. Then it will align its left leg to the right leg and the process will repeat until the Hamizon makes a 90 degree right turn.

4.2.1 Left Turn Gait

First Hamizon will align its body and leg into turning formation. Then it will shift its weight to the right and lift up its left leg. After that Hamizon will make its first step. While the left leg reaches the floor it will spin anti clockwise by 30 degree. Next Hamizon will shift its weight to left and lift up its right leg. Then it will align its right leg to the left leg and the process will repeat until the Hamizon makes a 90 degree left turn.

4.3 Summary

Hamizon have 22 DOF which is control by 22 servo motors. The total height of the robot is 35 cm. Static walking locomotion was chosen as the walking gait for Hamizon and it can travel forward and turn left or right.
CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

This chapter will summarize the result, end product and possible future developments of the project.

5.1 Accomplishments

A functional humanoid robot called Hamizon was fabricated in this project. Hamizon has the ability to walk forward and make left or right turns. Hamizon has 22 DOF installed in its structure and it was constructed to resemble closely the flexibility of the human body and it capable of many types of walking gait. Static gait was developed and integrated into Hamizon maneuvering system. Taken as a whole, Hamizon is a very successful project since the main objective to develop and fabricate a functional humanoid robot and integrate with the vision system (from others project) was archived. However, there are still much room for improvement for Hamizon in term of imitates human capability.
5.2 Future Development Suggestions

The future of the project relies on the continuity of the project. Therefore, students from FKE especially Mechatronic student are urged to continue this project and develop it into a better extra ordinary humanoid. I have faith in Hamizon that it could be one day evolve into a humanoid that has capability of decisions making and have the appearance of a human. The ultimate goal of Hamizon is towards perfection and the benefit is a wonder. However it is not a job of a one year period but a lifetime. It is not only the contribution of an individual but it is the contribution of many other people to make it reality.

5.2.1 Upgrade Hamizon Locomotion

Hamizon uses a static walking locomotion and it is not the best approach. With some modification to Hamizon such as adding waist to it structure, dynamic walking which is a better locomotion can be implementing into Hamizon’s system. When I designed the physical structure of Hamizon I had already put this under consideration. Therefore the integration will not be a problem and the only issue that needs to be resolved is the calculation of the exact torque needed and the replacement of the appropriate servo motors. By applying dynamic walking into Hamizon’s system, not only the speed of the robot can be improve to a more human-like walking gait but also others character such as running can be developed into the system.
5.2.2 Incorporate With Sensor

Like others life in the universe, sensors are needed for humanoid to communicate with the environment. Hamizon will not be different as it requires many sensors. As example human use their skin to touch or sense something and for a humanoid maybe the touch sensor will be a great deal. Human can balance their self and for humanoid maybe accelerometer sensor system will give a great benefit. It will not be the same as a human but we are providing it with the ability to adapt to the change of the environment.

5.3 Summary

The end product of this project is a humanoid robot that able to walk forward, turn to its both side and integrate with the vision system. This project is very successful and Hamizon is working within normal parameters. Improvement can be done by adding sensors into Hamizon system.
REFERENCE

8. Almir Heralic-Design and Control of the Prototype Humanoid Robot HR2
10. Developing Adjustable Walking Patterns for Natural Walking in Humanoid Robots James M. Jeanne, Princeton Université
APENDIX A

**Final Set Of Programming**

```c
unsigned long  ser_tmp;

void ser_tx(char c);
void ser_putstring(const char* text);

void main()
{
    int i;
    //LATD =0x00;
    TRISD =0x00;

    Usart_Init(9600);
    //Soft_Uart_Init(&PORTC, 7, 6, 115200, 0);
    Delay_ms(2000);
    PORTD = 0x01;

    // 1-berdiri statik
    ser_putstring("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700#24p1500#25p1500t2000\r");
    Delay_ms(3000);
    ser_putstring("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700#24p1600#25p1500t2000\r");
    Delay_ms(3000);
```

set_putstring("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700#24p1500#25p1500t2000\r");
Delay_ms(3000);

// pusing kepala
for(i=0;i<6;i++)
{
    set_putstring("#24p1700#25p1350t500\r");
    Delay_ms(500);
    set_putstring("#24p1300#25p1500t500\r");
    Delay_ms(500);
    set_putstring("#24p1700#25p1650t500\r");
    Delay_ms(500);
    set_putstring("#24p2000#25p1500t500\r");
    Delay_ms(500);
}
set_putstring("#24p1500#25p1500t1000\r");
Delay_ms(1000);
// pusing kepala tamat

// depa tangan
// depa tangan-1
set_putstring("#12p1500#13p780#14p2000#28p1500#29p1500#30p1500t1000\r");
Delay_ms(1000);
for(i=0;i<6;i++)
{
    // depa tangan-2
    set_putstring("#12p1000#13p780#28p2000#29p1500t500\r");
    Delay_ms(1000);
// depan tangan-3
ser_putchar("#12p1000#13p1380#28p2000#29p900t500\r");
Delay_ms(1000);
}

// depan tangan-2
ser_putchar("#12p1000#13p780#28p2000#29p1500t500\r");
Delay_ms(1000);

// depan tangan-1
ser_putchar("#12p1500#13p780#14p2000#28p1500#29p1500#30p1500t1000\r"");
Delay_ms(1000);

// depan tangan tamat

ser_putchar("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700#24p1500#25p1500t2000\r"");
Delay_ms(3000);

// berjalan kedepan
// 2-anjak ke kiri
ser_putchar("#0p1550#1p1300#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t2000\r"");
Delay_ms(3000);

// 3-berdiri sebelah kaki-kiri
ser_putchar("#0p1550#1p1300#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1480#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t2000\r"");
Delay_ms(3000);

for(i=0;i<6;i++)
{
   // 4- lutut kaki kanan ke depan
   ser_putstring("#0p1550#1p1200#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1550#19p1750#20p1480#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t2000\r");
   Delay_ms(3000);

   // 5-kaki kanan sentuh lantai
   ser_putstring("#0p1550#1p1200#2p1600#3p1200#4p1650#5p1450#16p1450#17p1850#18p1550#19p1750#20p1300#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t3000\r");
   Delay_ms(3000);

   // 6-senget ke kanan
   ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1650#5p1450#16p1450#17p1850#18p1450#19p1750#20p1300#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t2500\r");
   Delay_ms(3000);

   // 7-berdiri sebelah kaki-kanan
   ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1550#5p1450#16p1450#17p1850#18p1450#19p1750#20p1350#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t2000\r");
   Delay_ms(3000);

   // 8-angkat lutut kiri
   ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1550#5p1450#16p1450#17p2000#18p1450#19p1750#20p1350#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t2000\r");}
Delay_ms(4000);

// 9-kaki kiri sentuh lantai
ser_putstring("#0p1550#1p1400#2p1470#3p1200#4p1550#5p1600#16p1450#17p1900#18p1400#19p1750#20p1250#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t3000\r");
Delay_ms(4000);

// 10-senget ke kiri2
ser_putstring("#0p1650#1p1400#2p1600#3p1200#4p1680#5p1600#16p1550#17p1900#18p1550#19p1750#20p1200#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t3000\r");
Delay_ms(4000);

// 11-Kaki kanan terangkat
ser_putstring("#0p1650#1p1400#2p1600#3p1200#4p1750#5p1530#16p1550#17p1900#18p1550#19p1750#20p1450#21p1190#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t3000\r");
Delay_ms(3000);
}

// 2-anjak ke kiri
ser_putstring("#0p1550#1p1300#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1550#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t2000\r");
Delay_ms(3000);

// 1-berdiri statik
ser_putstring("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t2000\r");
Delay_ms(3000);

//berjalan kedepan
// jalan kebelakang
// 11-Kaki kanan terangkat
ser_putstring("#0p1650#1p1400#2p1600#3p1200#4p1750#5p1530#16p1550#17p1900#18p1550#19p1750#20p1450#21p1190#13p1100#14p1800#29p1200#30p1700t3000\r");
Delay_ms(3000);
// 10-senget ke kiri2
ser_putstring("#0p1650#1p1400#2p1600#3p1200#4p1680#5p1600#16p1550#17p1900#18p1550#19p1750#20p1200#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 9-kaki kiri sentuh lantai
ser_putstring("#0p1550#1p1400#2p1470#3p1200#4p1550#5p1600#16p1450#17p1900#18p1400#19p1750#20p1250#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t3000\r");
Delay_ms(3000);
// 8-angkat lutut kiri
ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1550#5p1450#16p1450#17p2000#18p1450#19p1750#20p1350#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 7-berdiri sebelah kaki-kanan
ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1550#5p1450#16p1450#17p1850#18p1450#19p1750#20p1350#21p1200#12p1100#13p1100#14p1800#28p1200#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 6-senget ke kanan
ser_putstring("#0p1550#1p1200#2p1470#3p1200#4p1650#5p1450#16p1450#17p1850#18p1450#19p1750#20p1300#21p1200#12p1800#13p1420#14p1800#28p1900#29p1200#30p1700t2500\r");
Delay_ms(3000);
// 5-kaki kanan sentuh lantai
ser_putstring("#0p1550#1p1200#2p1600#3p1200#4p1650#5p1450#16p1450#17p1850#18p1550#19p1750#20p1300#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t3000\r");
Delay_ms(3000);
// 4- lutut kaki kanan ke depan
ser_putstring("#0p1550#1p1200#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1550#19p1750#20p1480#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 3-berdiri sebelah kaki-kiri
ser_putstring("#0p1550#1p1300#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1550#19p1750#20p1480#21p1200#12p1800#13p1100#14p1800#28p1900#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 2-anjak ke kiri
ser_putstring("#0p1550#1p1300#2p1600#3p1200#4p1650#5p1530#16p1450#17p1850#18p1550#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t2000\r");
Delay_ms(3000);
// 1-berdiri statik
ser_putstring("#0p1550#1p1300#2p1500#3p1200#4p1650#5p1530#16p1450#17p1850#18p1500#19p1750#20p1300#21p1200#12p1500#13p1100#14p1800#28p1500#29p1200#30p1700t2000\r");
Delay_ms(3000);
// jalan kebelakang
void ser_putchar(const char* text) {
    char i = 0;
    // set_bit( pie1, TXIE );
    while( text[i] != 0 )
        ser_tx( text[i++] );
    // clear_bit(pie1,TXIE);
}

void ser_putchar(char c) {
    //set_bit( pie1, TXIE );
    //wait for txif to go hi
    while (!(PIR1 & 16)); //while (!TXIF);
    TXREG = c;
    //enable_interrupt(GIE); //?
}
APENDIX B

HAMIZON