# [DESIGN AND IMPLEMENTATION OF A WALKING ROBOT]

By

[MOHD ZURIX BIN MOHAMED]

FINAL DESIGN PROJECT

Submitted to the Electrical & Electronics Engineering Programme
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 AND IMPLEMENTATION OF A WALKING ROBOT**

by

[Mohd Zurix Mohamed]

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:

[Mr. Mohd Haris Md Khir]
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2003

# 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.

[Mohd Zurix Mohamed]

# ABSTRACT

The origin of robotics can be traced back to the ancient Greeks, who created movable statues. Around 270 BC, Ctesibus (a Greek engineer) made organs and water clocks with movable figures. The word robot was first used in 1921 play titiled R.U.R. Tossum's Universal Robots by Czechoslovakian writer Karek Capek. The word robot is a Czech word for workers. The play described mechanical servants, the "robots'. When the robots were endowed with emotion, they turned on their masters and destroyed them. Walking robots are essential especially when mobile robots can't reach a certain place. A good example would be climbing stairs or go on rough terrain. Each leg of the walking robot can analyze the balancing of the robot, thus de-counters it so that the robot would stay in position and would not fall. Volcanoes and well as for other rough terrain area uses legged robot to pursue. With the help of sensors to locate its position, walking robots can easily conquer an area. This report contains a project on design and implementation of a six legged walking robot. The robot is power using DC source and servo motors. With a chip, PIC16F84 as its brain, the robot is capable of manoeuvring itself and avoid any obstacles along its way. Analysis and research on the material to be used, electrical components as well as the constructions is also in this report. All coding for the microcontroller were done in C language and compiled using the PIC C Compiler.

# ACKNOWLEDGEMENT

Alhamdulillah, thank you to Allah s.w.t for giving me the strength to complete this project.

First of all, I would like to thank you Mr. Mohd Haris Bin Md Khir, my project supervisor for leading me and showing me the correct ways in pursuing this project. Without him I would be lost and may not be able to complete this project on time.
I would also like to thank the technicians for giving me a chance to use the equipments in the lab. They are most appreciated.

Secondly, I would like to thank you my parents as well as my friends who had indirectly supported in completing this project. Their eagerness to see my project complete raises my spirit and strengthens my motivation to complete this project. Thank you.

## TABLE OF CONTENTS

# LIST OF TABLE

# LIST OF FIGURES

# CHAPTER 1
# INTRODUCTION

## 1.1 Background

Historically, we have sought to endow inanimate objects that resemble the human form with human abilities and attributes. From this is derived the word anthrobots, robots in human form. Robot have been useful in industrials, hazardous duty, maintenance work, fire – fighting, medical, space explorations as well as wars.

Robots are indispensable in many manufacturing industries. For instance, robot welders are commonly used in automobile manufacturing. Other robots are equipped with spray painters and paint components. The semiconductor industry uses robots to solder micro wires to semiconductor chips. Other robots insert integrated circuits onto printed circuit boards, a process known as "stuffing the board".

## 1.2 Objective

i) To built a fully functional six legged walking robot with the capability to avoid obstacles along its way.

ii) To equip the walking robot with sensors as eye function.

iii) To program a microcontroller as the robot's brain for movement purposes.

## 1.3 Problem Statement

The ability to produce a perfect walking robot with good stability takes high technology as well as research. Robots are closely related to AI (artificial intelligence) where scientists are still on research to produce a robot which is capable of thinking and making its own decision (un-programmed). The main objective of this project is to design a six legged walking robot, which is capable of walking as well as avoid obstacles along its way. Both mechanical and electrical components have to be mastered to achieve this objective.

It's been said that balance and controls for walking robots are closely related. Less leg requires less control but lose out in balance while more legs will give more balance but, to control the movement or pace of the legs are more complex. According to books and websites, the best number of legs for walking robot would be six. More legs would create more complex controller.

The significance of this project is that a six legs walking robot will be made. The mechanism will be simple, yet durable, and is design for spy purposes. The robot will use a microcontroller as its main processor or 'brain', where it will interpret its surroundings. Complex sensors such as eye sight will be studied. Implementation of an ultrasonic sensor is considered.

## 1.4 Scope of Study

The walking robot consists of six legs, which is connected to the servo motors. The building of the robot will be divided into two different sections, which are the mechanical parts and electrical parts.

Mechanically, the robot must be able to stand and walk with good balance, as well as walk in forward, reverse and turn. The scope of study for mechanical parts will be the based on the current design of what is called, a spider robot. The backbone of the structure for the robot will be aluminum, as it is lights as well as durable

Electrical wise, the robot brain of the robot will be a microcontroller, which will control the movement of the robot – the DC motor running the gears. The microcontroller (PIC16F84A) output's can control direct current motor drives, using pulse width modulation, servo motor positioning, stepper motor and more.

Sensors will be place onto the robot, which will send signals to the microcontroller. Its function is as eyes, where it detects obstacles along the robots way. The most suitable sensors being considered now is the ultrasonic sensor, as it is more reliable compared to infrared.

# CHAPTER 2
# LITERATURE REVIEW

In the beginning, personal robots will focus on a single function (job task) or purpose. For instance, today there small, mobile robots that can autonomously maintain a lawn by cutting the grass. These robot are solar powered and don't require any training. Underground wires are placed around the lawn perimeter. The robots sense the wire, remain within the defined perimeter, and don't wander off.

NASA routinely sends unmanned robotic explorers where it is impossible to send human explorers. Why send robots instead of humans? Economics. It's much cheaper to send an expendable toot than a human. Humans require an enormous support system to function: breathable atmosphere \, food, heat, living quarters, and, quite frankly, most want to live through the experience and return to Earth in their lifetime.

Explorer spacecraft travel through the solar system, where their electronic eyes transmit back to Earth fascinating pictures of the planets and their moons. The Viking probes sent to Mars looked for life and sent back pictures of the Martian landscape. NASA is developing planetary rovers, space probes, spider-legged walking explorers and underwater rovers. NASA has the most advanced telerobotic program in the world, operating under the Office of Space Access and Technology (OSAT).

## 2.1 Type of robots

a. Industrial robots – going to work

Robots are indispensable in many manufacturing industries. For instance, robot welders are commonly used in automobile manufacturing. Other robots are equipped with spray painters and paint components. The semiconductor industry uses robots to solder (spot weld) micro-wires to semiconductor chips. Other robots (called pick and place) insert integrated circuits (ICs) onto printed circuit boards, a process known as stuffing the board.

Robots are ideally suited for performing repetitive tasks. Robots are faster and cheaper than human laborers. This is one reason that manufacturing goods are available at low cost. Robots improve the profit and competitiveness of manufacturing companies. Without robots, many companies would no longer be able to compete in their industries.

b. Design and prototyping

Some robots are useful for more than repetitive work. Manufacturing companies commonly use computer aid design (CAD), computer aided manufacturing (CAM), and computer numerical control (CNC) machines to produce designs, manufacture components, and assemble machines. These technologies allow an engineer to design a component using CAD and manufacture the design of the board using computer controlled equipment quickly. Computers assist in the entire process, from design to production.

c. Maintenance

Maintenance robots specially designed to travel through pipes, sewers, air conditioning ducts, and other systems can assist in assessment and repair. A video camera mounted on the robot can transmit video pictures back to an inspection technician. Where there is damage, the technician can use the robot to facilitate small repairs quickly and efficiently.

d. Fire-fighting robots

Better than a home fire extinguisher, how about a home fire fighting robot? This robot will detect a fire anywhere in the house, travel to the location, and put out the fire. Fire fighting robots are so attractive that this type of robot has an annual national competition open to all robotists, who try to develop the best robot.

e. Medical robots

Medical robots fall into four general categories. The first category relates to diagnostic testing. In the spring of 1992, Neuromedical Systems Inc. of Suffern, New York, released a product called Papnet. Papnet is a neural network tool that helps cytologists detects cervical cancer quickly and more accurately. Papnet uses an advanced image recognition system and neural network. The network selects 128 of the most abnormal cells found on a pap smear for later review by a cytologist.

The second medical category relates to telepresence surgery. Using a specially developed medical robot, a surgeon is able to operate on a patient remotely. The robot has unique force-feedback sensors that relate to the surgeon the feed of the surgeon's movements. This makes it possible for the surgeon to perform microsurgical procedures that can only be dreamed about today. For instance, let's suppose the surgeon moves his or her hand 1 inch. The computer would translate that to 1/10" or 1/100" travel. It is now possible for the surgeon to perform delicate, microscopic surgery that would normally be impossible.

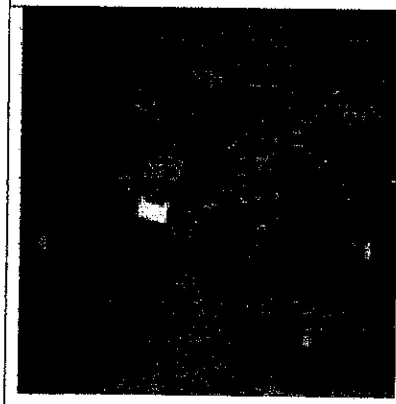
f. Nanotechnology medical bots

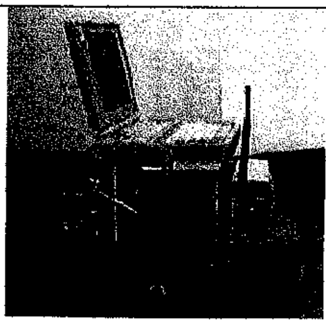
Nanotechnoloy is used to create small, microscopic robots – image robots so small that they can be injected into a patients's bloodstream. The robots travel to the heart and begin removing the fatty deposits, restoring circulation. Or the robots travel to a tumor, where they selectively destroy all cancerous cells. Another hope of nanomedical robots is to stop and reverse the aging process in humans. Tiny virus-sized nano-bots enter each cell resetting the cell clock back to 1.

g. War robots

One of the first applications of robots is war. If forced into a war, robots can help us win, and win fast. Robots are becoming increasingly more important in modern warfare. Drone aircraft can track enemy movements and keep the enemy under surveillance. Smart bombs and cruise missiles are other examples of "smart" weaponry.

## 2.2 Other type of robots

| | |
|---|---|
|  Mobile Robots | Mobile robots are able to move, usually they perform task such as search areas. A prime example is the Mars Explorer, specifically designed to roam the mars surface. Mobile robots are a great help to such collapsed building for survivors Mobile robots are used for task where people cannot go. Either because it is too dangerous of because people cannot reach the area that needs to be searched. |
|  Rolling robots | Rolling robots have wheels to move around. These are the type of robots that can quickly and easily search move around. However they are only useful in flat areas, rocky terrains give them a hard time. Flat terrains are their territory. |
|  Walking robot | ***Walking Robots:*** Robots on legs are usually brought in when the terrain is rocky and difficult to enter with wheels. Robots have a hard time shifting balance and keep them from tumbling. That's why most robots with have at least 4 of them, usually they have 6 legs or more. Even when they lift one or more legs they still keep their balance. Development of legged robots is often modeled after insects or crawfish. |
|  Stationary Robots | Robots are not only used to explore areas or imitate a human being. Most robots perform repeating tasks without ever moving an inch. Most robots are 'working' in industry settings. Especially dull and repeating tasks are suitable for robots. A robot never grows tired, it will perform its duty day and night without ever complaining. |

| | |
|---|---|
| | In case the tasks at hand are done, the robots will be reprogrammed to perform other tasks. |
| <br>Autonomous robots | Autonomous robots are self supporting or in other words self contained. In a way they rely on their own 'brains'.<br><br>Autonomous robots run a program that gives them the opportunity to decide on the action to perform depending on their surroundings. At times these robots even learn new behavior. They start out with a short routine and adapt this routine to be more successful at the task they perform. The most successful routine will be repeated as such their behavior is shaped. Autonomous robots can learn to walk or avoid obstacles they find in their way. Think about a six legged robot, at first the legs move ad random, after a little while the robot adjust its program and performs a pattern which enables it to move in a direction. |
| <br>Autonomous robot | An autonomous robot is despite its autonomous not a very clever or intelligent unit. The memory and brain capacity is usually limited, an autonomous robot can be compared to an insect in that respect.<br><br>In case a robot needs to perform more complicated yet undetermined tasks an autonomous robot is not the right choice.<br><br>Complicated tasks are still best performed by human beings with real brainpower. A person can guide a robot by remote control. A person can perform difficult and usually dangerous tasks without being at the spot where the tasks are performed. To detonate a bomb it is safer to send the robot to the danger area. |

# CHAPTER 3

# THEORY

In this section, the author would like to divide the theory into three major parts, which is the mechanical design, circuit and components and microcontroller. The first section which is the mechanical design consists of the movement concept, materials and the alternative designs to suite the movement. The movement concept is based on a technology called tripod gait where only three legs will be set on the ground during walking mode. The circuit and components section will consists of the electrical components that are used mainly by robots for its movements and coordination, such as sensors and motor. The circuit and components section will also elaborate the types of servo motors. The microcontroller section elaborates on the chip that will be used for the robot brain, and how it will be implemented to the robot.

## 3.1 Mechanical Design

The design of the robot is usually taken of the shape of an insect or crawfish. Legged robot must be capable of standing in balance if either one or more legs are lifted from the ground. The legs of the robot must be strong enough to support the body mass during walking mode as well as still mode. The technology which will be used for the movement purposes is called the tripod gait.

### *3.1.1 Tripod Gait*

Conceptually, the six legged robot main driver is the servo motor. Three servo motors will be used for walking and other movements. Figure 2.1 shows the location for the servos.

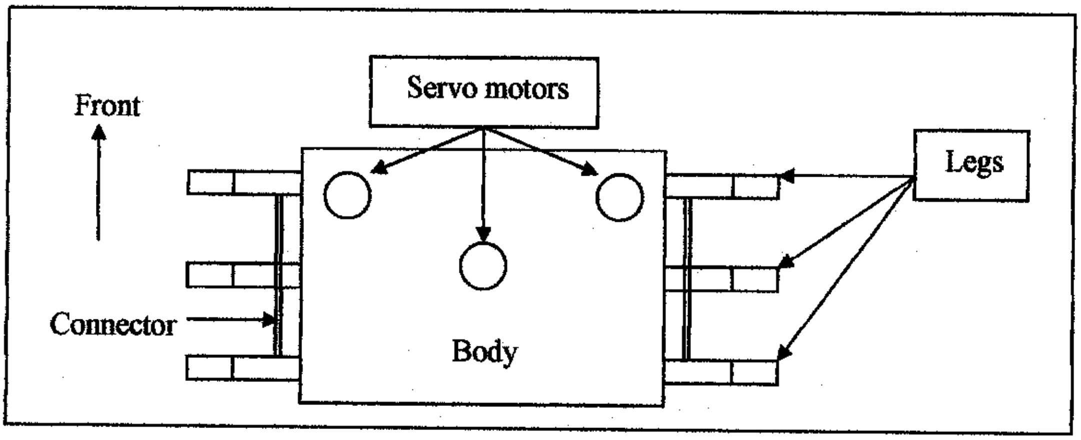



Figure 1 Top view of design

Figure 1 shows the top view of the initial sketch of the design. Basically, there are three servos placed on the body of the robot, one on each side of the body and the third on the center. As seen in Figure 1, the connector connects the front most leg with the back most leg. This shows that both front and back legs are moving together via the servo motor.

The design is adopted from a technology of robots called tripod gait, in which the word 'tripod' refers to stand on three legs at a time. The robot will move its legs alternately in which three legs will be on the ground at a time. The tripod technology provides simple, yet balanced theory design.

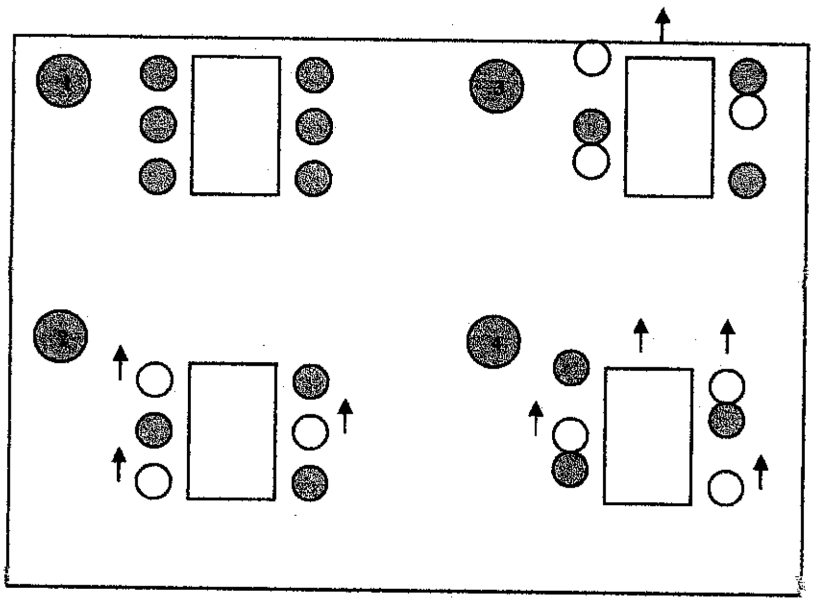


Figure 2 Tripod Gait

Figure 2 is the basic function how a tripod gait works. Each leg has its own servo motor for movement purposes. However, the design will not be as the above. It will be simpler in terms of the movement and maneuvering. Only three servos will be used for the movement.

Referring to Figure 2, figure 1 shows the robot at rest. All feet are on the ground. From the resting position the robot decides to move forward. To step forward, it lifts three of its legs (figure 2, white circles), leaving its weight on the remaining three legs (dark circles). Notice that the legs supporting the weight (dark circles) are in the shape of a tripod. This is a stable weight- supporting position. Our creature is unlikely to fall over. The three lifted legs (white circles) are free to move, and they move forward.

Figure 3 illustrates where the three lifted legs move. At this point, the creature's weight shifts from the stationary legs to the movable legs (figure 4). Notice that the creature's weight is still supported by a tripod position of legs. Now the other set of legs move forward and the cycle repeats. This is called a *tripod gait*, because the creature's weight is always supported by a tripod positioning of legs. John Lovine 2002, Robots, Androids and Animatrons , McGrawHill, New York.

### *3.1.2 Material*

The type of material that is taken into consideration is aluminum sheet and bars. The body and legs of Figure 1 is made from aluminum which is considerably light and at the same time durable. Since the robot must be able to stand on three legs at a time, the material of the legs must be capable of handling the body mass during tripod mode. The type of material to be chosen is based on analysis that is done in Chapter 4 – Methodology / Project Work section. Usually robots material is preferably light and easy to shape. The aluminum characteristic of easy to shape makes it more possible to design the robot. However, analysis will be done to choose the best material available.

### 3.2 Circuit and Components

This section consists of theories on servo motor and its function. The servo motor is used as a muscle for the legs to move. The type of servo motor used is Futaba S9303. Sensors that are taken into consideration for eye function are ultrasonic and infrared sensor. Analysis will be done in choosing the most appropriate type of sensor.

#### *3.2.1 Servo motor*

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled airplanes to position control surfaces like the elevators and rudders.

The servo motor's function fully depends on the pulses (PWM) driven out from the PIC16F84A chip. Different pulses will produce different rotating angle for the servo motor. A pulse width of 2 milliseconds will rotate the servo to a full 180 degrees clockwise, 1 millisecond will rotate the servo at a full 180 degrees anticlockwise and a 1.5 millisecond pulse width will center the servo. These pulses have to be fed to the servo continuously until the servo reaches the targeted position.

The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. In the picture above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, its somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

### 3.2.2 Sensors

To make the robot to be able to interact with the objects around it, eyes are needed. Sensors are used to distinguish between a clear pathway and a blocked one. For this project, two types of sensors are being considered. They are the infrared sensor and the ultrasonic sensor.

#### 3.2.2.1 Infrared Sensor

Infrared (IR) sensors detect low frequency light and are widely used in robotics for tracking, collision avoidance and communication. The advantage of this sensor is that they only detect IR light oscillating at a specific frequency (usually around 40kHz). The 40kHz waveform can be modulated by another signal. The receiver module has also been designed to receive an impressed signal on the 40kHz carrier wave .this produces a robust communication link. Primarily, the receiver module responds only to the 40kHz IR signal permitting the receiver to 'see' the IR light being transmitted from the transmitter, reject other light sources, then allow the modulation on the 40kHz wave to be detected.

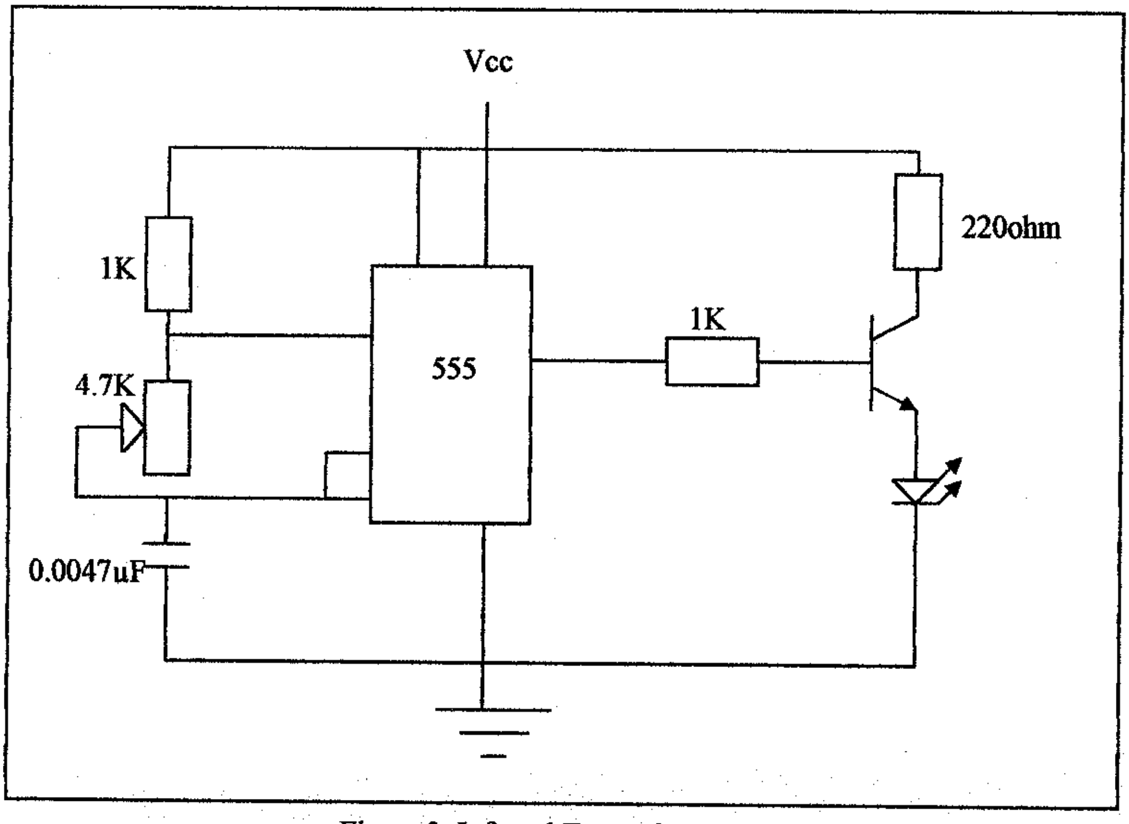



Figure 3 Infrared Transmitter

Figure 3 is a schematic of the transmitter. It uses a 555 timer set up in astable mode. Potentiometer is used to adjust the frequency output. The output of the timer is connected to a 2N2222 NPN transistor. An infrared LED is connected to the emitter of the transistor. When the circuit is on, there will be no light coming out from the LED. The infrared light is not detectable by the human eye.

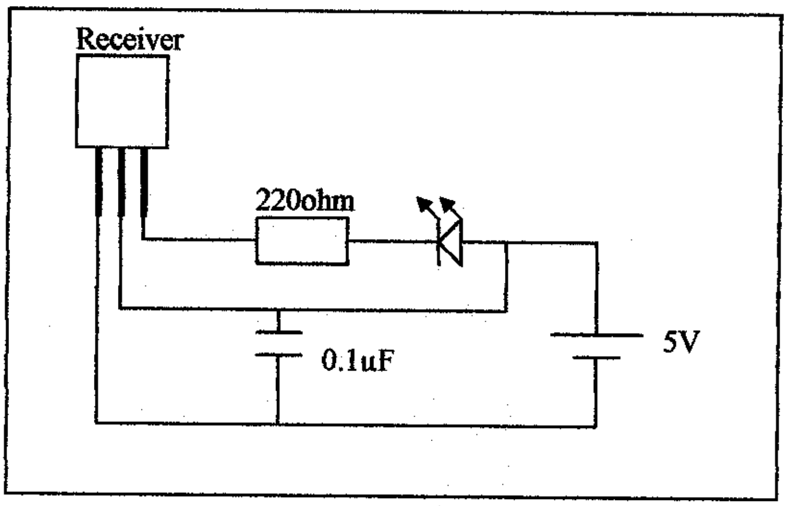



Figure 4 Infrared Receiver

Figure 4 is a schematic of the infrared receiver. The center frequency is 37.9 kHz with a bandwidth of 3 kHz. The output is active low. What this means is when the receiver module detects the signal, the output drops to ground. The output is equivalent to an open collector of an NPN transistor. The output can sink sufficient current to light an LED. In the test circuit, the LED will light when the module is receiving the signal.

### *3.2.2.2 Ultrasonic sensor*

Ultrasonic are often used for ranged finding and collision detection. these modules are used to quickly measure the subjects distance. When interfaced to a microcontroller, the units can accurately measure distance. The basic operation follows the same scheme used for infrared collision avoidance, except we are using sound instead of light. The transmitter transmits a 40khz signal to an ultrasonic transducer. Another transducer is positioned alongside the transmitter transducer. When the robot approaches a wall or obstacle, the 40kHz sound is reflected back to the receiver, whose output increases in amplitude. When the output increases beyond the present point, the comparator trips, relaying that there is an obstacle detected.

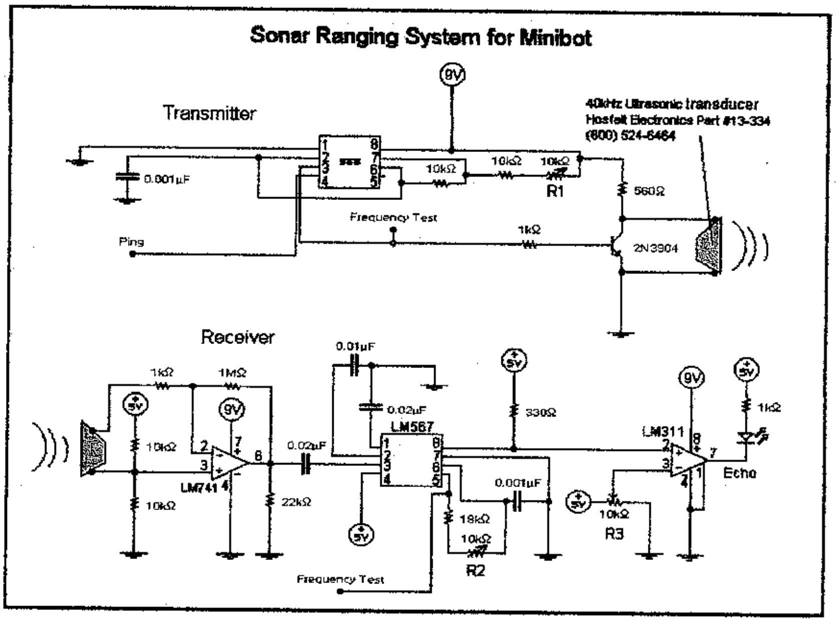



Figure 5 Ultrasonic Transducer

#### *3.2.2.3 Calibrating Ultrasonic transducer (Refer Figure 5 – Ultrasonic Transducer).*

The Transmitter. Using an oscilloscope or frequency counter, set pin 4 **HIGH** and adjust R1--trim potentiometer-- until a 40kHz signal on pin 3 of the 555 timer chip appears. Try to use a trim pot for R1 because it will allow higher resolution in terms of adjustments. One thing to note: add plenty of bypass capacitors on the power supply. Minimize the effects of noise as much as possible. I used 100uF, 0.01uF and 0.001uF by-pass caps on the power supply.

The Receiver. The schematic is straight forward. Again, use trim pots for R2 and R3. Add by pass caps where appropriate. Once the receiver is completed, tuning is required. Again, using an oscilloscope, adjust R2 until the signal on pin 5 of the LM567 is 40kHz. The frequency measured on pin 5 indicates the lock-on frequency of the tone decoder. Next, adjust R3 such that the voltage on pin 3 of the LM311 is at about 2.5 volts. This voltage level causes the output of the LM311 to only swing **LOW** when the output from the tone decoder is less than 2.5 volts.

### 3.3 Microcontroller

The brain of the robot will be controlled by a chip called PIC16F84A, in which it will be programmed to perform the required tasks. The PIC16F8X has up to 68 bytes of RAM, 64 bytes of Data EEPROM memory, and 13 I/O pins. A timer/ counter is also available. The PIC16CXX family has special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals. The SLEEP (power-down) mode offers power saving. The user can wake the chip from sleep through several external and internal interrupts and resets.

# CHAPTER 4

# METHODOLOGY / PROJECT WORK

Process of designing the robot consists of three different sections, the body (mechanical parts), circuit (electrical) and programming (PIC16F84A).

Before proceeding with construction, a few designs were made and analysis was done to choose the best design. Designs were based on criteria in the following order:

i) Balance – The robot must be able to stand stably during stop/idle mode and walking mode (tripod mode).

   a. Legs – size and distribution

   b. Body – weight distribution throughout

ii) Weight – The weight of the body must not exceed the attribute of the servo motor;

   a. Servo specifications:

      i. Type – Futaba S9303

      ii. Speed – 0.19 sec/60°

      iii. Torque – 7.2kg-cm (100 oz-in)

      iv. Size – 40.5 x 20 x 39.5 mm

      v. Weight – 64.5 g

iii) Microcontroller – PIC16F84A

### 4.1 The Body

At earlier stage, the author had two different body designs. The designs are named Spybot-I and Zurix-II. Below is the sketch of both designs:

#### *4.1.1 Design A (Spybot-I)*

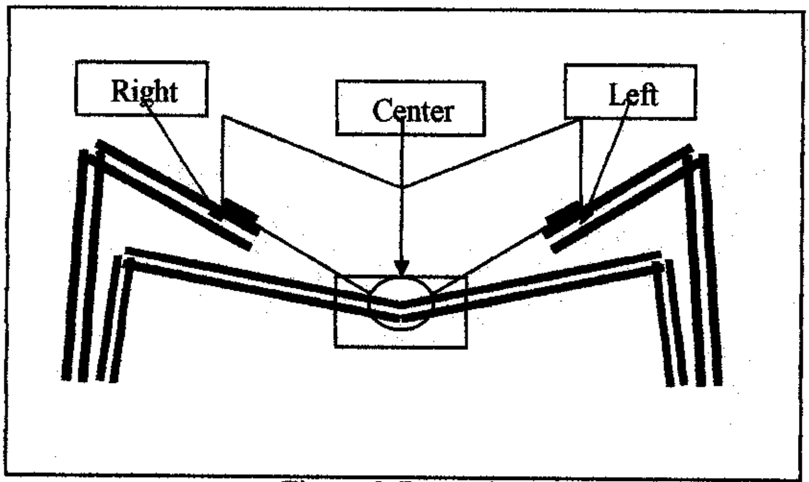



Figure 6 Front view

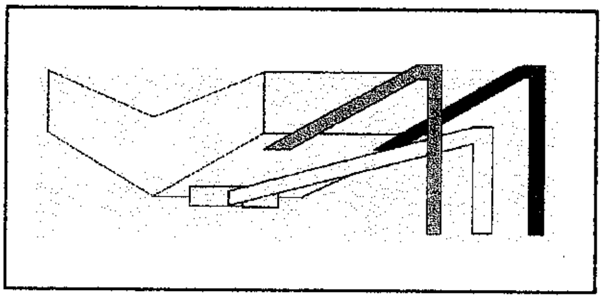


Figure 7 3D view

Figure 6 shows the front view of the robot. As seen, the shape of the body is almost similar to a spider. The name itself symbolizes the shape of the robot (Spider Robot). The front and back legs on each side are connected via a small metal rod thus making both of them move at the same time. Figure 7 shows the 3 dimension view of the robot. As seen the legs are bent into an angle of less than 90 degrees. This design was made to have a robot with wide body and long legs for stability purposes.

### *4.1.2 Design B (Zurix-II)*

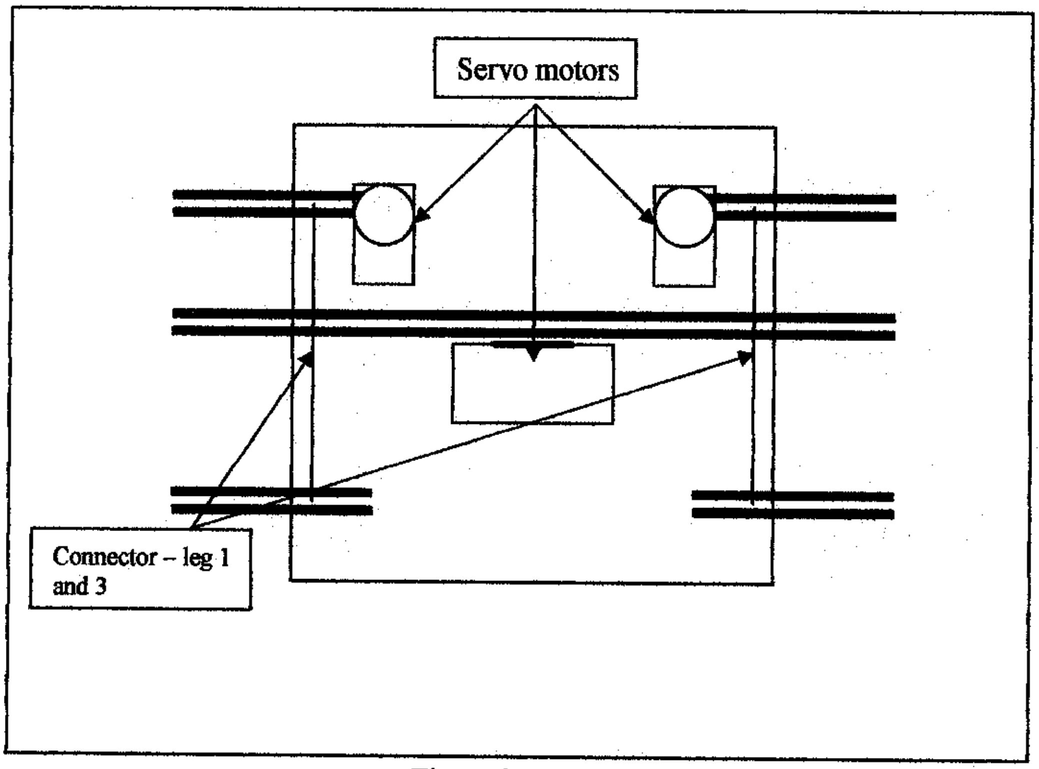



Figure 8 Top View

Figure 8 shows the top view of the second design (Zurix-II). The body design is basically aluminum bars joint together using nuts and bolts. The second design is somehow heavier than design A. Analysis and discussion on both designs are done for both designs (Chapter 5 – Results and Discussion). Similar to design A, the front and back legs are also joint together. In other word, both robots uses the same concept of walking – tripod gait. Design B (Zurix-II) is somehow simpler than design A, but more durable and heavier. Design B uses solid aluminum bars that even by dropping the robot, it would not damage or immobilize the robot.

Both designs are analyzed based on the criteria needed. Results and discussion of the chosen design is elaborated in Chapter 5 – Results and Discussion.

### 4.2 Dimensions

The author chooses aluminum bars to build the robot. The robot will consists of square and hollow bars and tied up using 2 inches screws. Results and analysis shows that Design B (Zurix-II) fits most of the criteria (Refer Chapter 5- Result and Discussion). Below is the dimensions of Zurix-II.

The dimension of the robot is as follows:

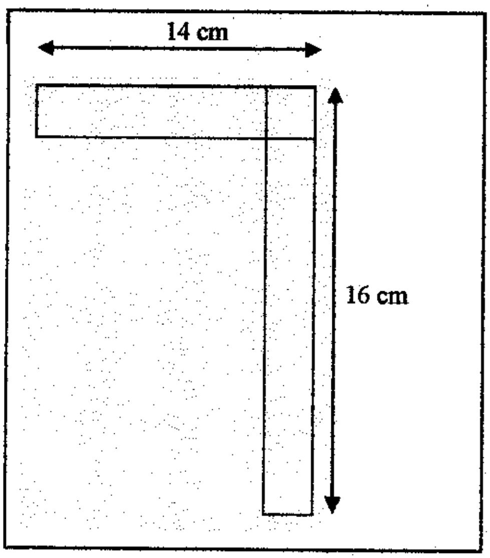



Figure 9 Leg

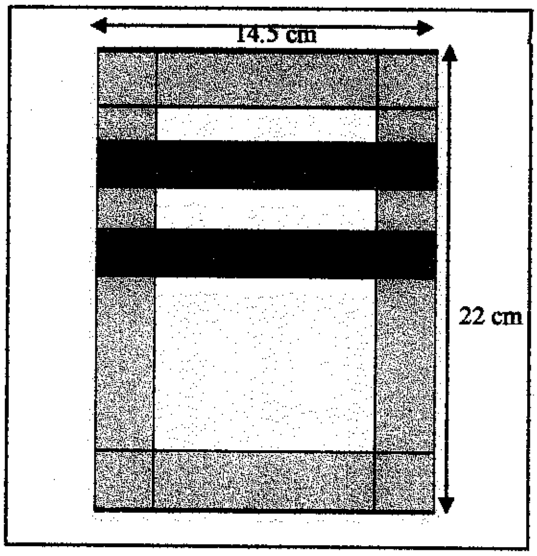



Figure 10 Body Top View

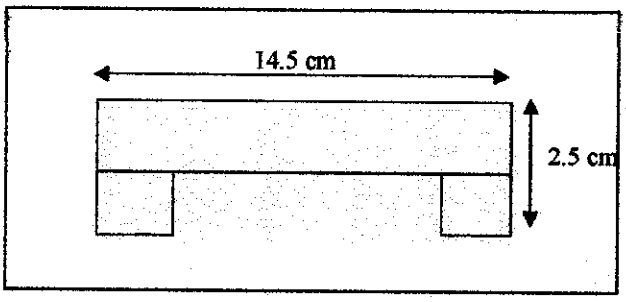



Figure 11 Body Front View

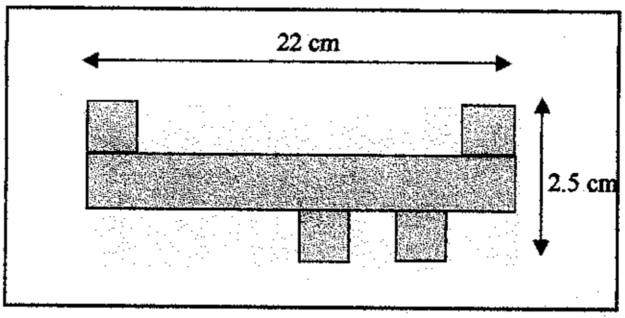
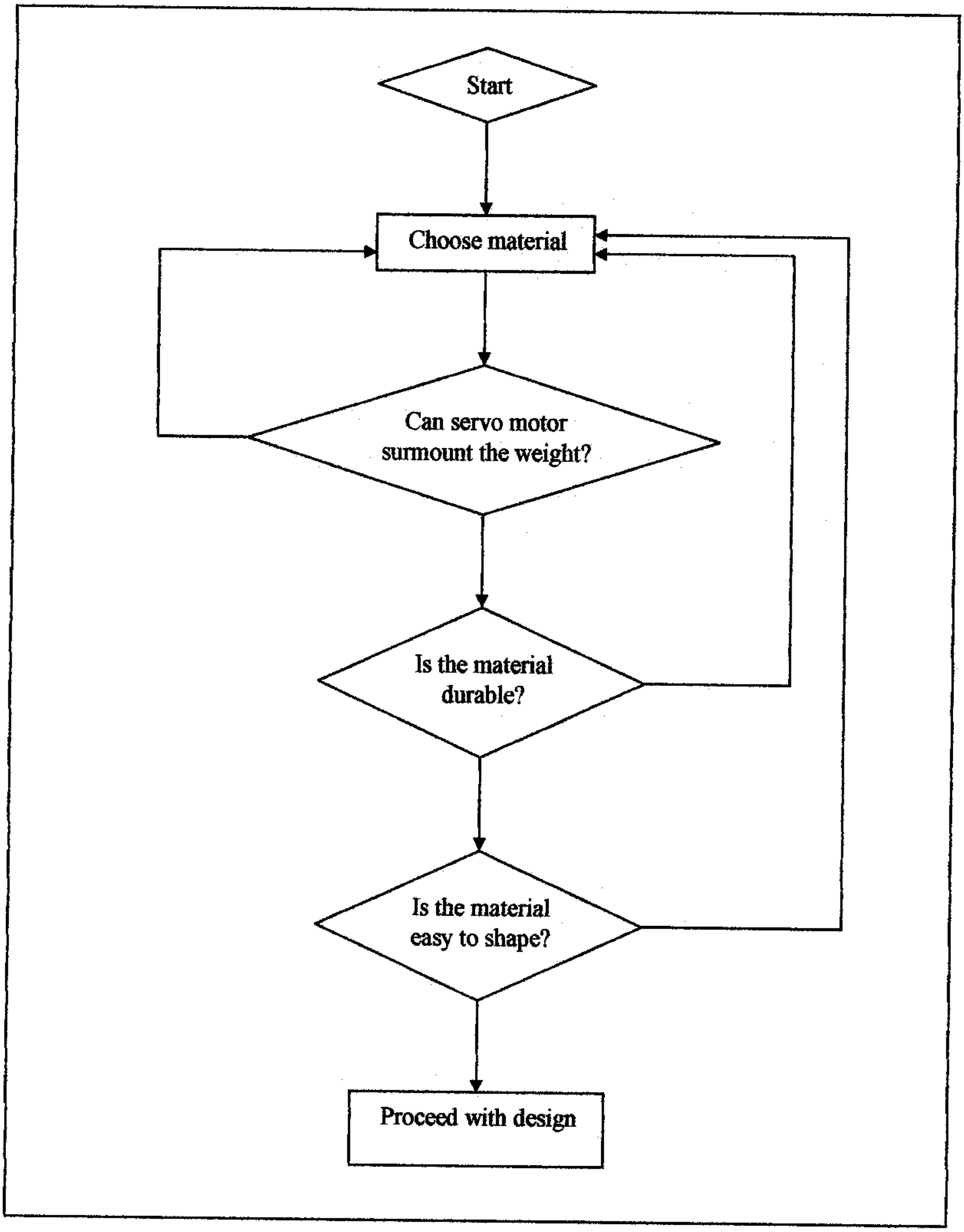



Figure 12 Body Side View

Figure 9 shows the dimension of the robot leg. The leg is made out of light aluminum 'U' shaped bar that has been bent to create an 'L' shaped leg. The leg is enforced with nut and bolt, and an aluminum sheet as a bending support.

Figure 10 shows the top view of the main body. The body is made of square and hollow aluminum bars which has been cut and put together using nut and bolts. The material used for the body is slight heavier than the legs. This is to ensure durability of the robot as the body has to support the servo motors and the circuits. Figure 11 and 12 shows the front and side view of the robots body.

### *4.2.1 Flow diagram – Choosing the body's material.*



Referring to the flow diagram, the first step in choosing the material is the weight constrains. The weight must not exceed the maximum lifting capability of the servo motor. The servo motor must be able to lift the load (material) and hold it in that position without damaging the servo motor internal mechanism.

The second step is to make sure that the chosen material is durable. A material may be light but at the same time may not be durable enough to sustain the robot balance. Improper material may lead to out of balance and weak robot. The material must be strong enough to sustain the weight of the robot during tripod mode (Refer Chapter 3 – Theory).

The third step in choosing the material after going through the first and second step is to analyze weather the material is easy to shape or not. A material such as plastic can be both lightweight and durable. But it might not as easy to shape. Considerations in choosing the material has to be looked upon based on the criteria and limitations.

## 4.3 The Circuit

The circuit of the robot can be divided into four sections. They are the PIC16F84A circuit, voltage regulator, ultrasonic sensor and overall connection. The main function of the chip is to maneuver the robot.

PIC16F84A circuit

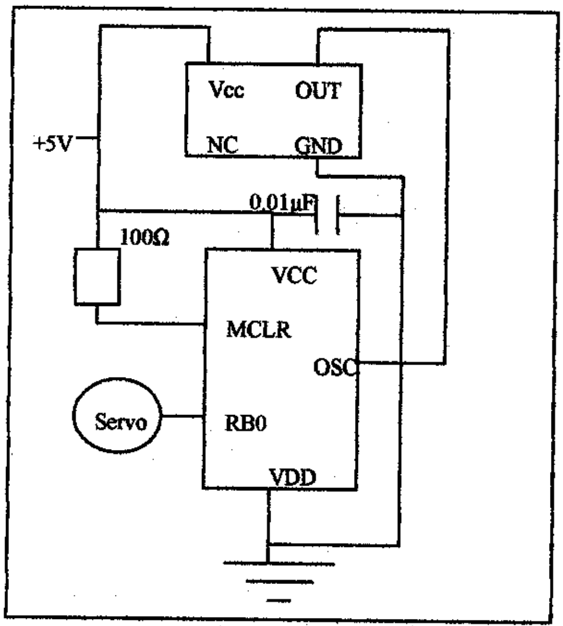



Figure 13 Main Circuit

Above (Figure 13) is the circuit for the PIC16F84A. The chip is powered by a 5 volts supply and run by a 4MHz crystal clock (oscillator).

Beside is the picture of the PIC circuit. As seen the circuit only consists of four major components, which are the PIC16F84A, clock, resistor and a capacitor.

### *4.3.1 Voltage regulator circuit*

Since both the servo and the microcontroller uses a 5 volts power supply, a voltage regulator is designed to step down a 9 volt battery. Regulator type AN7805 is used to obtain a 5 volt output. Below is the circuit diagram for the voltage regulator.

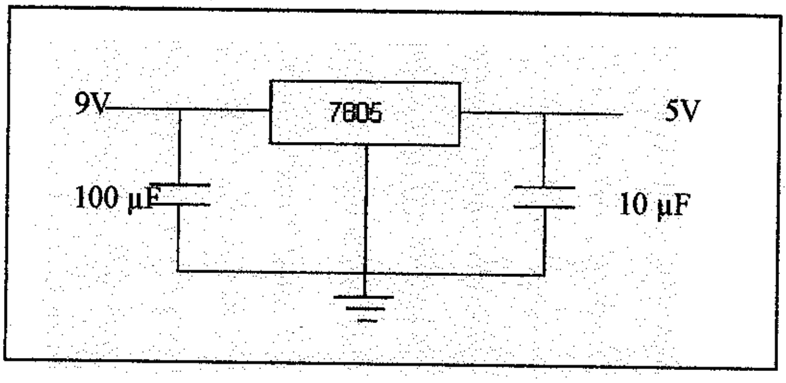



Figure 14 Voltage Regulator

The circuit is designed to contain two voltage regulators, in which one will power up the circuit, the other one will supply power to the servo motor. This is to ensure a maximum current is supplied to the motor for maximum output power.

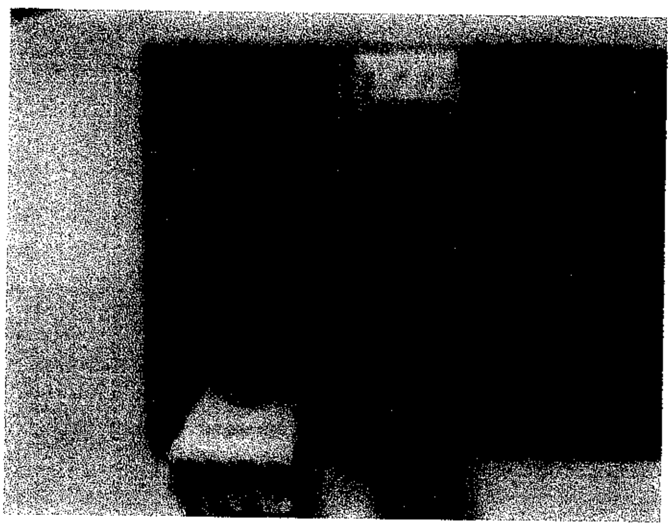


Figure 15 - Left is the completed circuit of the voltage regulators.

### *4.3.2 Ultrasonic Sensor circuit*

For the ultrasonic sensors; the transmitter emits an ultrasonic signal (40kHz). The 555 timer chip of the transmitter provides the driving 40kHz signal. Every time the reset pin (pin4) of the 555 timer goes high, a resulting signal of 40kHz on pin 3 is used to drive the ultrasonic transducer. Then, the receiver simply listens for the return echo after it bounces off an object. The small echo signal, when detected, is amplified 1000 times using a standard operational amplifier (LM741 op-amp). The signal is then fed into a tone decoder (LM567) set to lock onto a 40kHz signal. The output of the tone decoder is HIGH when no echo is heard and swings LOW when an echo is detected. The output from the tone decoder can now be fed into a microcontroller or some other type of IC to determine when an echo was received. To help minimize false triggering, the output is fed into a voltage comparator set to trigger at the appropriate level. The LED at the output of the comparator acts as a visual indicator when an echo is detected (very useful when debugging). The typical range of this system is from a few inches to 5-6 feet, depending on the quality of the components, shielding, and most important, tuning.

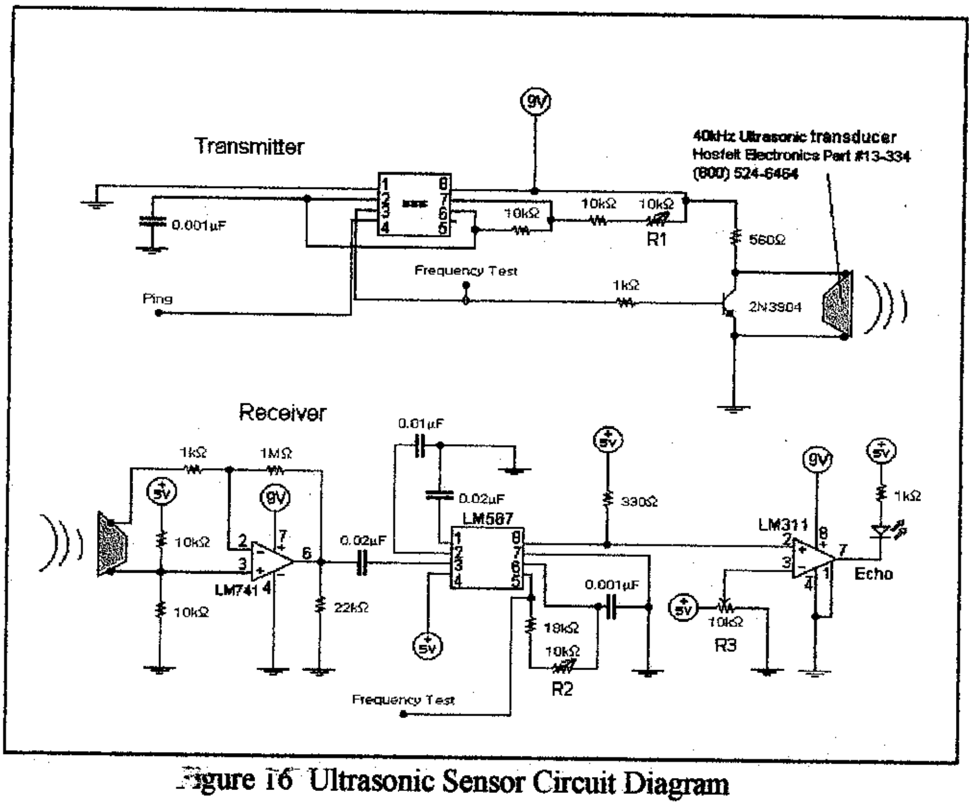



Figure 16 Ultrasonic Sensor Circuit Diagram

#### *4.3.3 Overall Connection*

The overall connection is as follows:

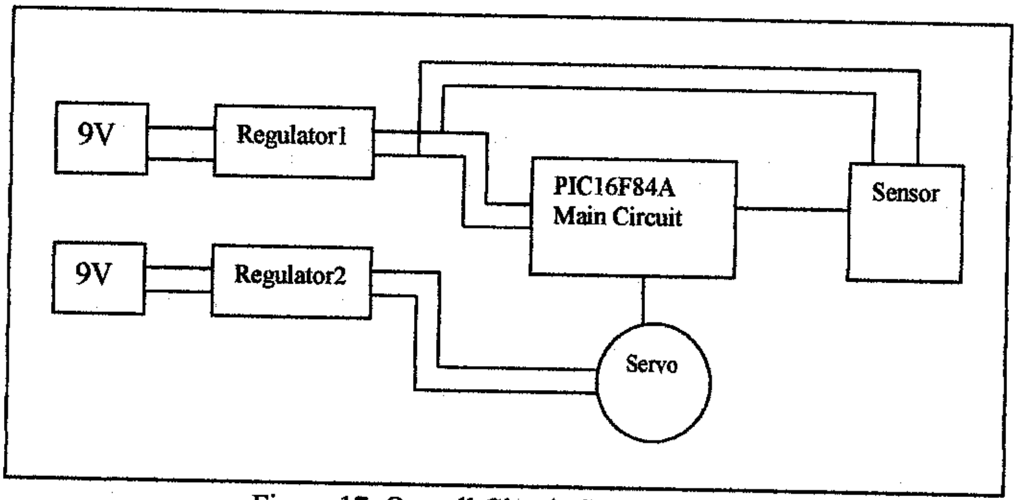
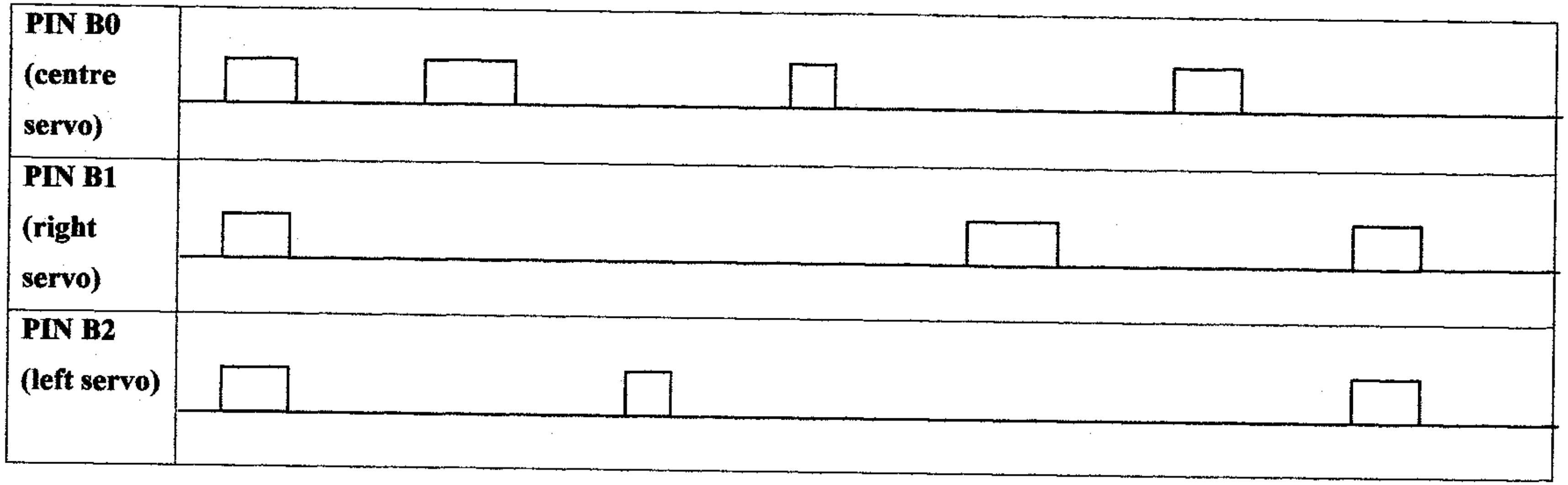
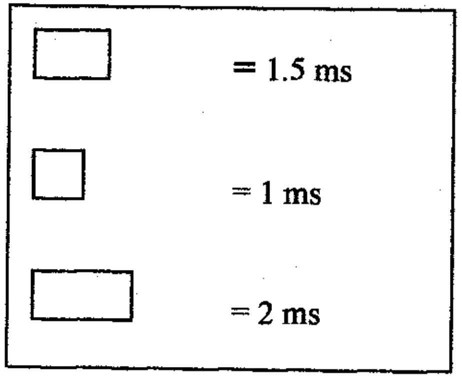



Figure 17 Overall Circuit Connection

Figure 17 shows the overall circuit connection for all electrical components. As seen the main circuit is the chip, whereas all the other devices and components are connected to it. The chip symbolizes the brain of the robot. The servo in Figure 17 represents three servo motors which each are allocated to a different port from the main circuit (output ports B0, B1 and B2).




The timing diagram above shows the output of Pin B0, Pin B1 and Pin B2 looking through an oscilloscope. These are called pulses where the chip PIC16F84 produces and sends it to the servo motor. Depending on the length of the pulse, the servo motor will rotate accordingly to the pulse width. A 1.5ms pulse will rotate the servo motor to the center, 1 ms pulse will rotate the servo motor to the left and a 2 ms pulse will rotate the servo motor to the right.

# CHAPTER 5

# RESULTS AND DISCUSSION

After findings and research, results are obtained. Mainly, research and analysis was done by trial and error. For instance, the weight of the robot for both designs was unknown until both designs are complete. Weight testing can only be done after the design if fully complete. The robot's leg for both designs uses the same material – 'U' shaped aluminum bar. These aluminum bars are found to be durable, light and easy to shape. The body for *Spybot – I* uses aluminum sheet where all components will be placed inside the body and will not be seen whereas *Zurix –II* uses hollow aluminum bars that is joint together making a frame. One disadvantage of using these hollow aluminum bars is that it is more heavily compared to aluminum sheet, but it is more durable and its solidity ensures that the robot will not easily be crushed if dropped.

On the electronics section, coding for pulses have been obtained using C language. These pulses are sent to the servo motor for rotation angle. Connection for all circuit have been done and proved to work. The servo motor is capable of lifting both designs during walking mode. However, *Spybot – I* shows imbalance during tripod mode (Walking mode). The location for both servo motors and legs are not able to cope with the wide body of *Spybot – I*. Although *Sypbot – I* is much lighter than *Zurix – II*, it cannot be taken as the main design due to this disproportion. Balance is one of the criteria, and *Spybot – I* does not meet that requirement. Refer to Table 5.1 for summary of *Spybot – I* and *Zurix – II*.

## 5.1 Mechanical Design

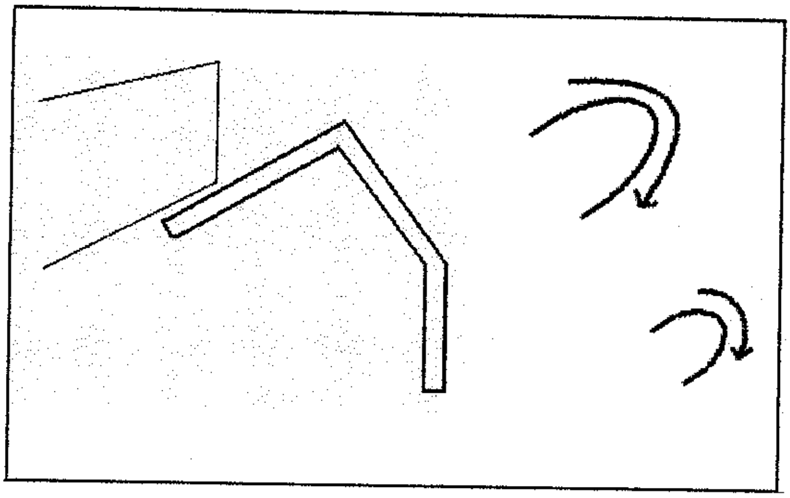
After research and analysis, the Author concluded the following findings:

| | Spybot - I | Zurix - II |
|---|---|---|
| **Balance and Stability** | | |
| ➢ Still/Idle Mode | ✓ | ✓ |
| ➢ Walking Mode (Tripod Mode) | χ | ✓ |
| **Weight** | | |
| ➢ Compared between *Spybot-I* and *Zurix-II* | Light | Heavy |
| ➢ Weight within servo motor's attribute? | ✓ | ✓ |
| **Legs** | | |
| ➢ Able to withstand body weight during stop/idle mode | ✓ | ✓ |
| ➢ Able to withstand body weight during walk/tripod mode. | ✓ | χ |
| **Body** | | |
| ➢ Balanced weight distribution during walking/ tripod mode | χ | ✓ |

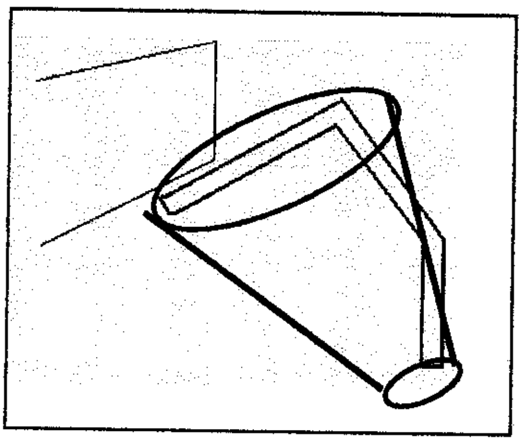
Table 1 Design Comparison

Table 1 shows the difference between Spybot-I and Zurix-II. After analysis, the author concluded that the best design would be Zurix-II. The table shows that Zurix-II legs are not able to withstand the weight of the robot during walking/ tripod mode. Although, the weight of the robot is still within the servo's attributes. To counter the strength of the legs, the author decided to write a program that will maintain the legs at a position

The author had two different designs for the robot. At the beginning of the project work, the author had designed a robot of which its body is solid aluminum. Testing for single servo has been done and analysis shows that the design was not suitable due to the rotation angle of the legs. Moving legs will create a coned shaped movement. As seen in Figure 5.1, the upper part of the leg is larger in diameter when turning, while the lower part of the leg is smaller.



Left : Figure 18 Rotation of leg for Sypbot-I



Left: Figure 19, shows a clearer picture of the coned shaped movement made by the robot's leg. Change of design has to be done in order to eliminate this problem.

### *5.1.1 Other problem faced on Spybot-I*

Another problem faced is that the body of the robot is too heavy. The servo motor is capable of lifting the body during operation mode, but unable to maintain the body at that position. Continuous supply of pulse would solve this problem, however, state function has to be used and this will cause the coding to be extremely long.

### *5.1.2 Zurix-II*

The body of the robot is changed while the legs are maintained. The angel of the legs are eliminated and it is now perpendicular to the ground. The robot looks just like a frame, in which its weight is reduced to servo motor function purposes.

One factor which was not taken into consideration earlier was the ability of the servo motor to maintain the lift position. The servo must maintain the lift position for other movement to take place. To do this, continuous pulse has to be given to the servo motor while lift mode, while other servo motor do other functions. To do this, an alternative design for the circuit was considered. Instead of using a single PIC16F84A chip, the robot will be equipped with four chips instead.

Three chips are programmed to continuously supply a pulse, each at different rate, while one chip is used to control these chips. The servos are all connected to the three chips so that each servo can be chosen to give a movement from each of the pulse. The master chip controls the output of the pulse going to the servos. This way, two servos may run at the same time, at each different pulse rate.

## 5.2 Circuits and components

This section contains the results of the electrical and microcontroller section. The components are put together and tested on *Spybot – I* and *Zurix – II* and proved to work the pulses sent to the servo motors are correct and shows tripod movements. Outputs from the chip are port B0 for center servo, B1 for right servo and B2 for left servo.

### *5.2.1 Pulses for the servo motor*

The servo motor is capable of rotating only 0 to 180 degrees. The degree of rotation depends on the length of the pulse given to the servo. At 1ms pulse, the servo will rotate the shaft all the way to the left, while 2ms pulse will turn the shaft all the way to the right. By varying the pulse width between 1 and 2 ms, the servo motor shaft can be rotated to any degree position within its range. Each pulse is fed into the servo 25 times before feeding a different pulse into the servo. C coding is used for programming the chip.

To create a high output from the PIC, delay it for 1ms and low the output for 17ms.

```
Output_high(PIN_B0);       ← Create a high output at pin B0
Delaytime_ms(1000);        ← Maintain the high output for 1ms
Output_low(PIN_B0);        ← Create a low output at pin B0
Delaytime_ms(17000);       ← Maintain the low output for 17ms
```

The pulses fed to the servo motor need to be refreshed at every 17ms. This is to ensure that the servo reaches the wanted position. The coding below represents the usage of 'for' to produce a pulse that is refreshed every 17ms and looped for 25 times.

```
for (i=1;i<=25;i++)
{
Output_high(PIN_B0);
Delaytime_ms(1000);
Output_low(PIN_B0);
Delaytime_ms(17000);
}
```

*5.2.1.1 The relationship of pulse width to servomotor armature position are shown below.*

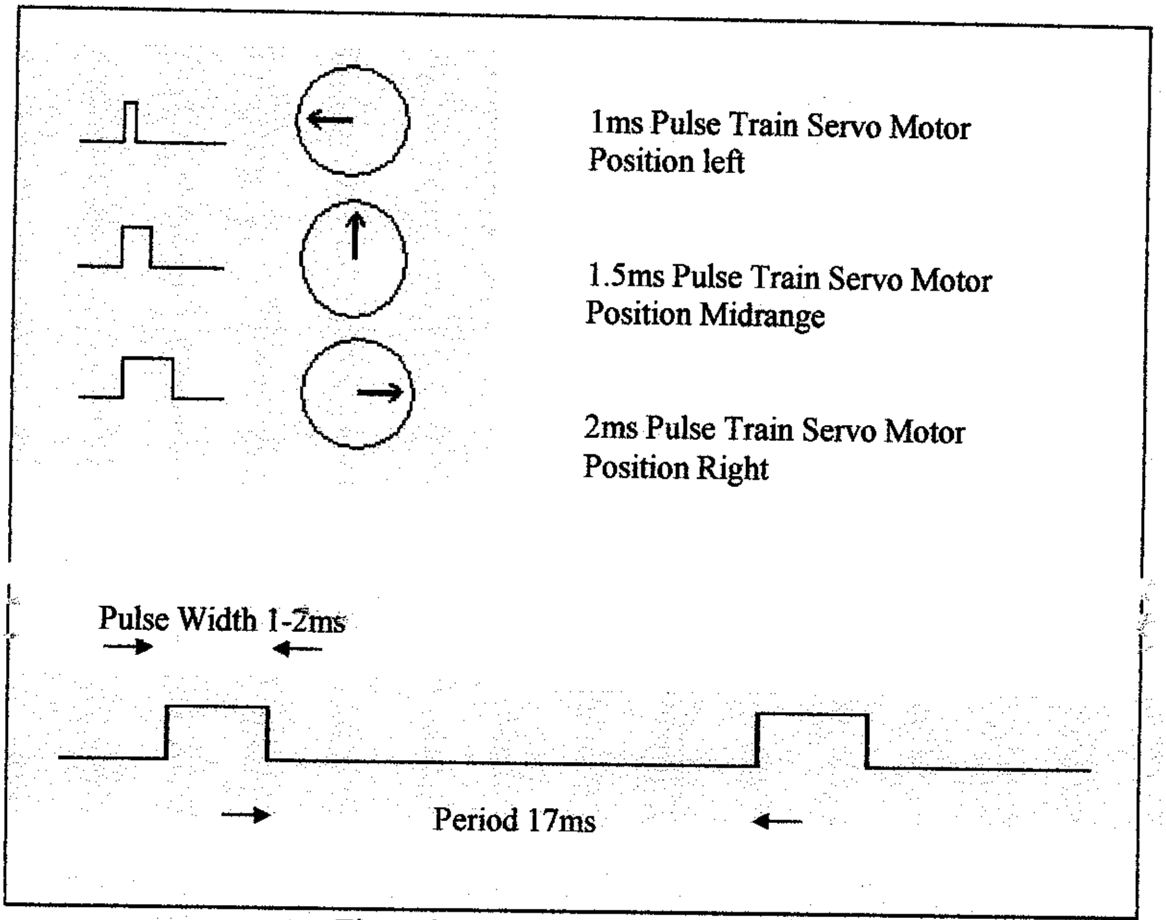



Figure 20 Servo motor armature position

Figure 20 shows the relationship of pulse width to the servo motor armature position. As seen in the figure, the pulse width determines the rotation of the servo motor. Figure 20 illustrate in details of the rotation of the servo motor and the pulse width. Elaboration on the pulses given to each servo at a time is on the next section.

*5.2.1.2 Flow diagram of robot's operation*

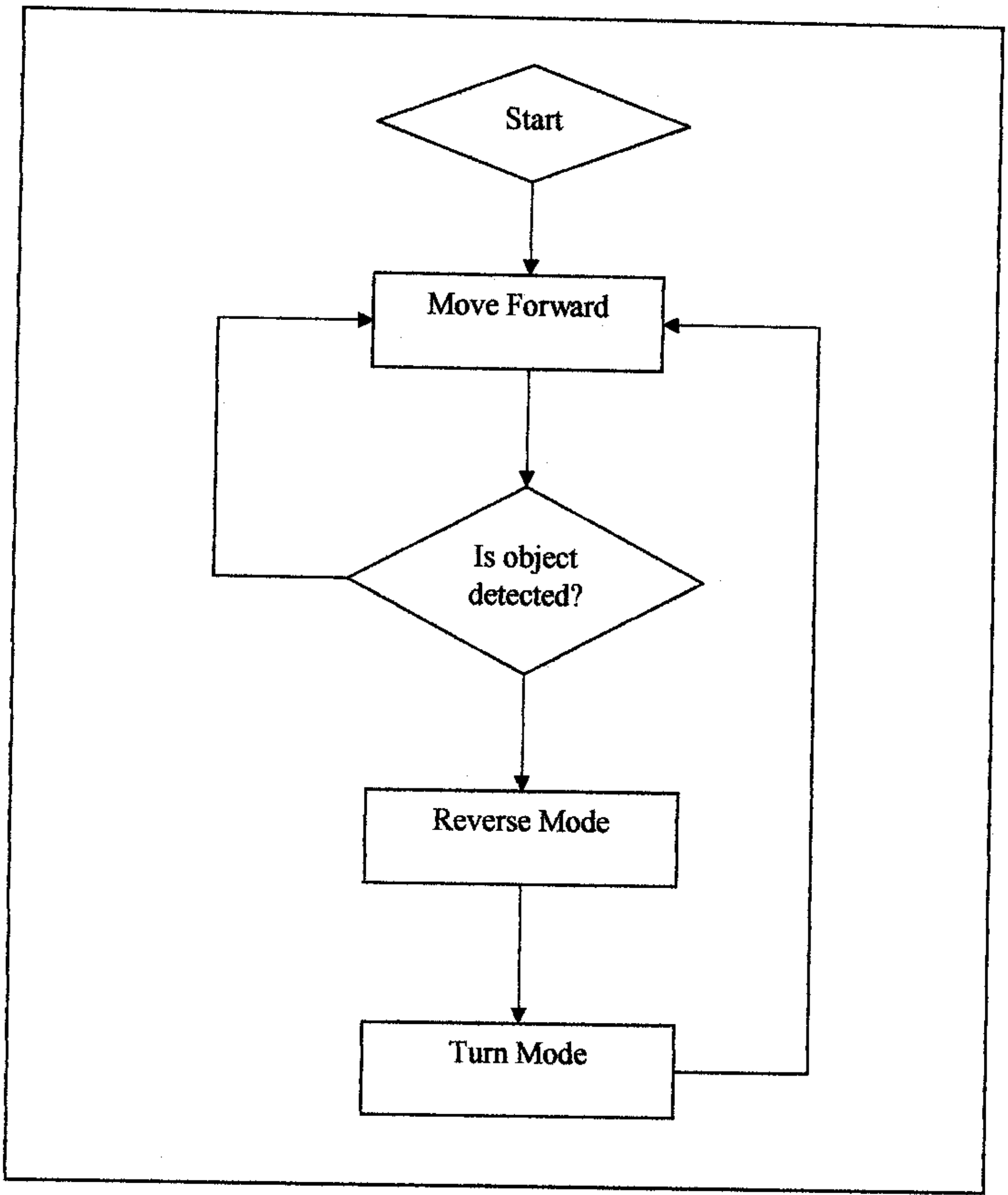



Figure 21 Robot's Movement

Figure 21 shows the flow diagram of the robot's movement. Upon starting the robot, it will start walking straight. The ultrasonic sensor will pickup any obstacles along the way. If there is no obstacle along the way, the robot will keep on walking forward. If the ultrasonic sensor senses an obstacle, the robot will change it's protocol to reverse mode. The robot will perform reverse and turn mode. This loop will continue until the robot is switched off. Other sensors may be added to the robot to make it livelier. For instance, light resistive sensors may be added to make the robot act as an insect whereas it will try to avoid light and move to dark areas.

#### *5.2.1.3 Microcontroller outputs*

The outputs for the servo motors are taken as B0 for center servo, B1 for right servo and B2 for left servo. The needed timeline to produce movement for the tripod gait robot is simplified in Figure 22.

Figure 22 shows the output from the microcontroller in pulses that are sent to the servo motor for operation. Each pulse is sent in sequence and does not collide. This is to ensure that the robot perform one movement at one time. Figure 22 shows the loop of pulse for forward movement. Pulses for reverse operation are basically the inverse of forward movement (Refer appendix for coding details).

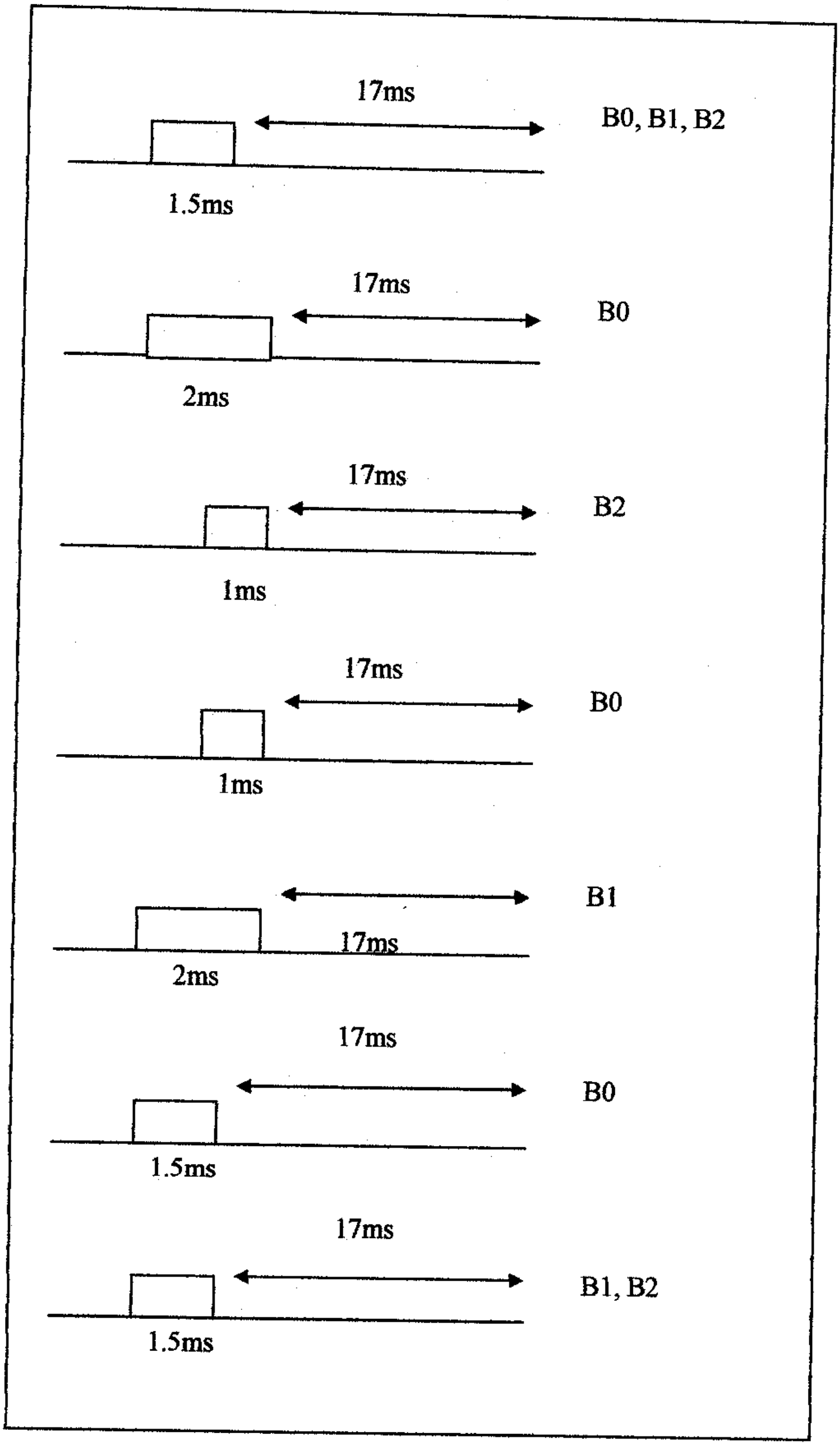
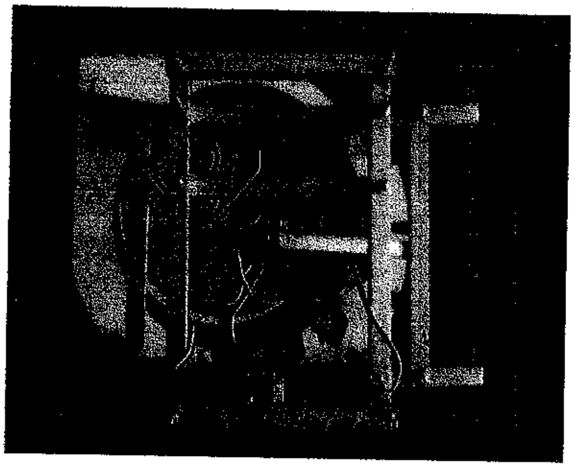
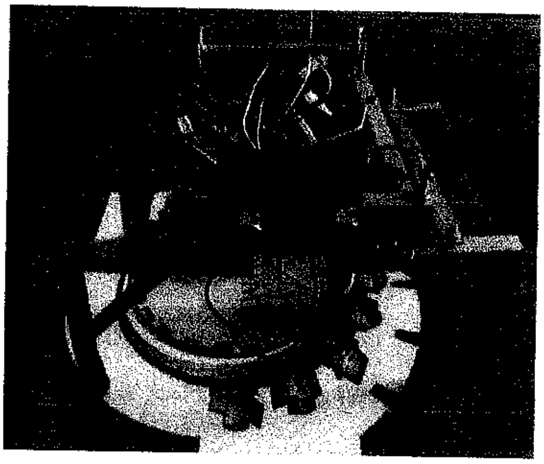


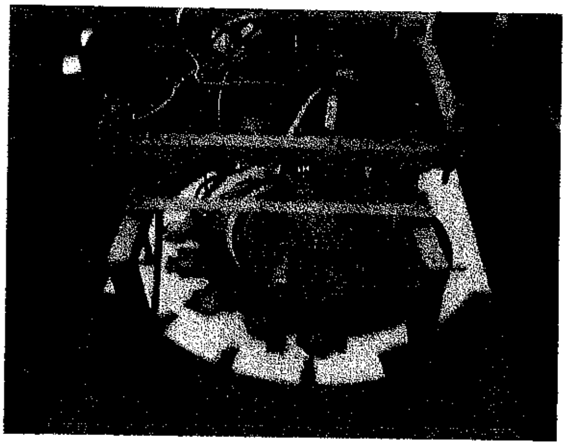

Figure 22 Timeline

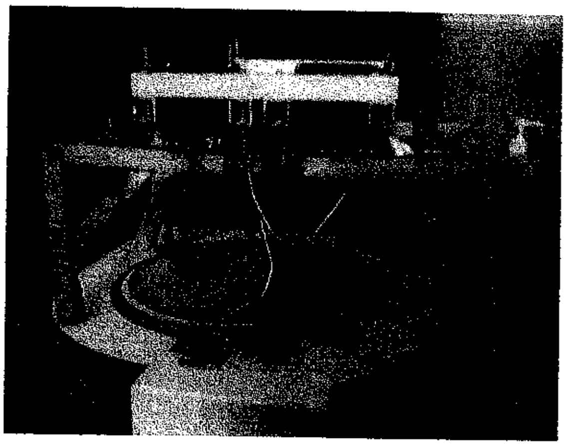
B0, B1 and B2 is the output for center, left and right legs from the PIC16F84A. As seen on the diagram above, the repeating 17ms represents the refresh rate at which the same signal will be given to the servo at each gap of 17ms. This is to ensure that the servo reaches the wanted position. If the same signal is given to the servo at which the servo is at the timed position, the servo will maintain that position until a different signal is given.

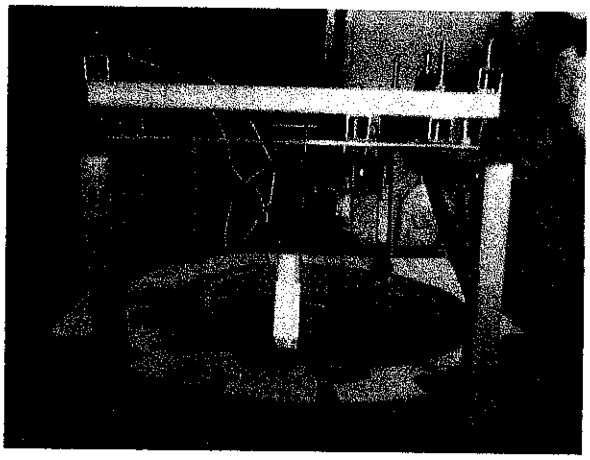
### *5.3 Pictures of the completed Robot (Zurix-II)*











## 5.4 Robot's Performance

The completed robot is tested for performance in terms of stability, walking, and obstacle avoidance. There are two different methods of programming the robot. One is by path planning, which a pre determined path is programmed on the chip. For example, a pre determined path may be walking in circle. However, *Zurix – II* has an onboard sensor for obstacle avoidance purposes thus the need for pre determined path is not necessary.

### *5.4.1 Stability*

*Zurix – II* shows stability during standing still and walking mode. It is capable of standing on 3 legs at a time. However, since it is only using three servo motors, the turning mode may cause instability on rough terrains. Adding more servo motors may fix this problem.

### *5.4.2 Walking mode errors*

During forward movement, the robot tends to walk towards the left. This may be due to un-calibration of the servo motors. Each servo motor may have different timing for a rotation. This too may lead to errors. Repeat calibrations may mend this problem.

### *5.4.3 Obstacle avoidance*

PIC16F84 chip does not support analogue to digital converter and since the ultrasonic sensor's output is in analogue, the obstacle avoidance may not be as efficient. The ultrasonic sensor's output if fed to a relay where switching takes place for forward or reverse mode. Using PIC16F877 instead of PIC16F84 may solve this problem as PIC16F877 have a built in A/D converter.

# CHAPTER 6

# CONCLUSION AND RECOMMENDATION

This project is 85% completed and currently operational. The microcontroller which is used (PIC16F84A) can be reprogrammed to perform other tasks. The six legged robot is suitable for explorations as well as a start of an AI (Artificial Intelligent). The robot is capable of walking forward, reverse and turn. Other modes of operation can be done just by simply reprogramming the chip.

Three objectives of this project have been accomplished. The robot has not been able to turn correctly. Research and investigation to fix this problem is still in progress.

In choosing the design, Design A was not chosen due to a problem that occurs during walking mode. The design's weight is light, however is very unstable. The number of legs was insufficient to surmount the wide body. Design B however is heavier but very stable during walking mode. Design B was chosen because of this stability.

The chip used is PIC16F84. This chip can be upgraded to PIC16F877 for more outputs. It also contains A/D converter for a more precise range system – sensor.

Improvements and new designs which can be made to this project is the ability to follow its master. A sensor will be placed on the master while the other on the robot. This will create a new 'robot pet' which will allow new inventions to come. Other sensors such as infrared, or hearing sensor can be added to the robot to give it more life.

The author hopes that this project will come in handy to those who are interested in robotic designs. This report is expected to become a guide for beginners into programming a PIC to designing the mechanism of the robot.

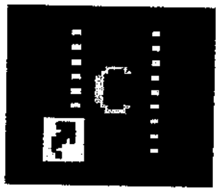
## References


1) John Lovine 2002, Robots, Androids and Animatrons , McGrawHill, New York.
2) Walker notTexasRanger http://www.renchbots.com/walker/walker.htm
3) Robot Type http://binky.thinkquest.nl/~ll106/Knowledge/Robottypes/main.html
4) Seattle Robotic Society http://www.seattlerobotics.org/guide/servos.html
5) Kam Leang Robotics http://www.leang.com/robotics/info/articles/minison/minison.html

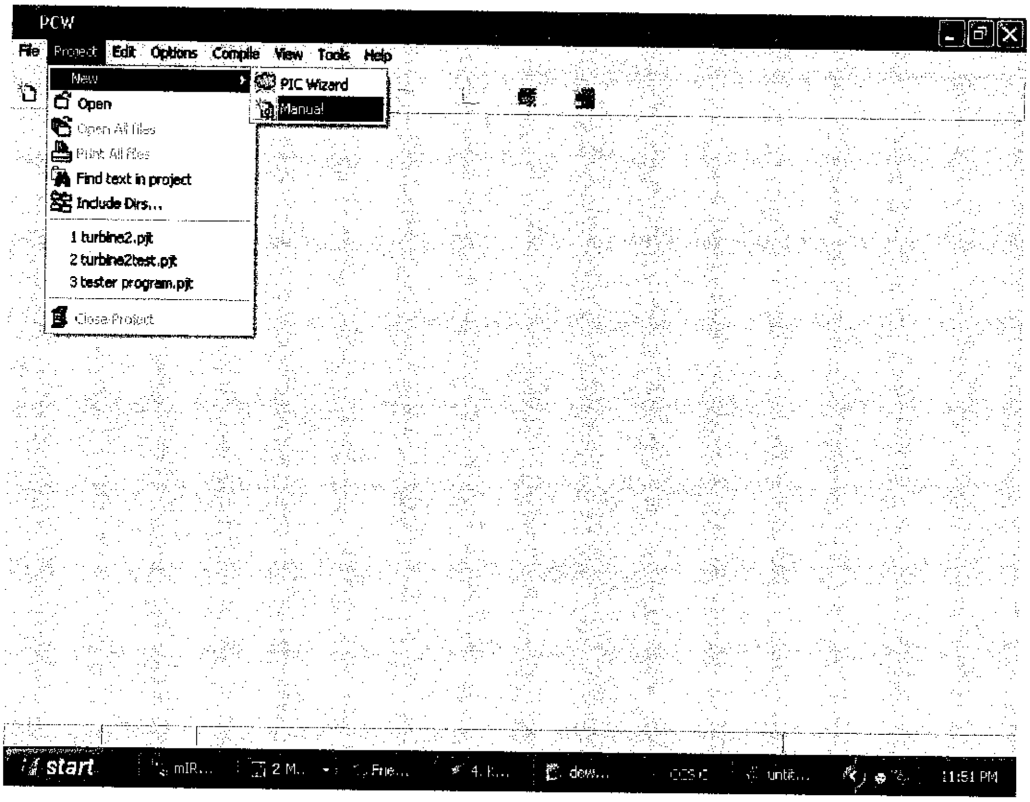
# Appendix

## *Using PIC C Compiler*

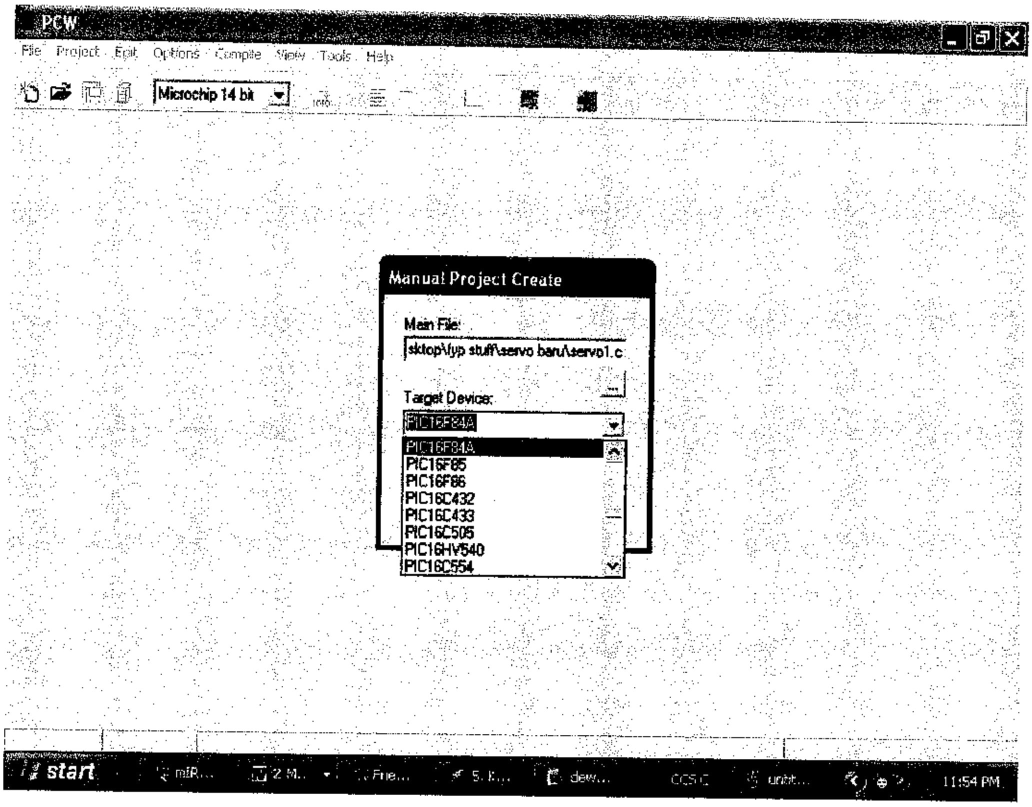
Start the PIC C Compiler by clicking on the icon shown below:



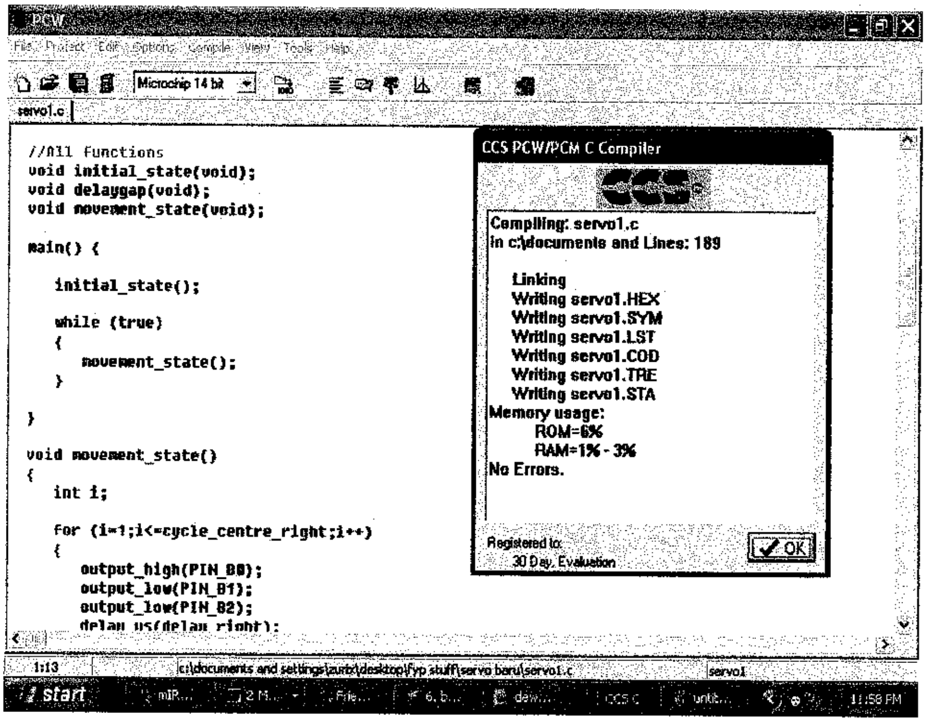
Once the program starts, click on project and select new. From the menu, select new and another drop down menu will appear. The user may choose to use the PIC Wizard or Manual. For this project, the author chooses Manual.



Another window popup will appear for the user to choose the type of PIC to be used, and the location of the C file which was created earlier. The author chooses PIC16F84A for the PIC selection.




After the codes are loaded into the program, select the option compile. The PIC C Compiler will start compiling the c codes. A popup window will appear indicating there are any errors while compiling.

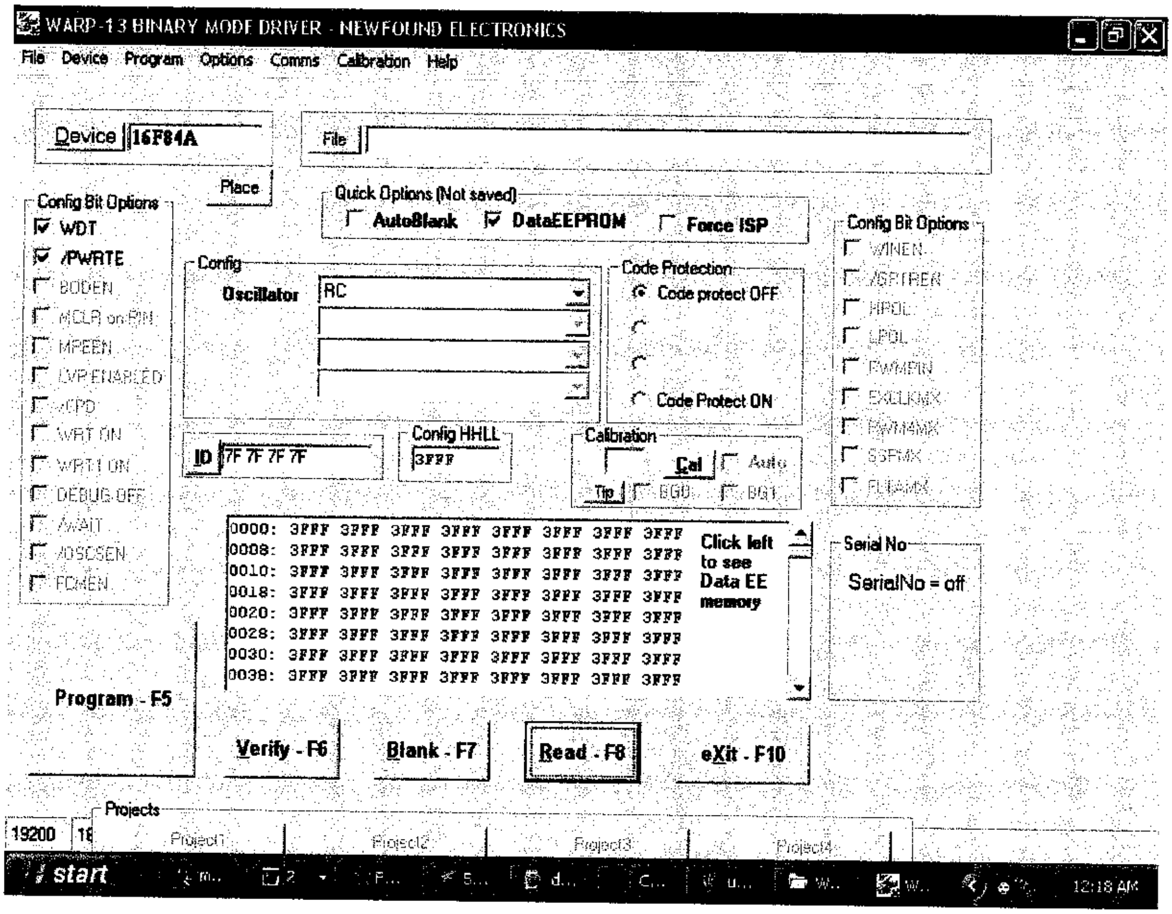



The PIC C Compiler program will create a hex file with similar filename with the original c file. The hex file is now ready to be uploaded into the PIC.

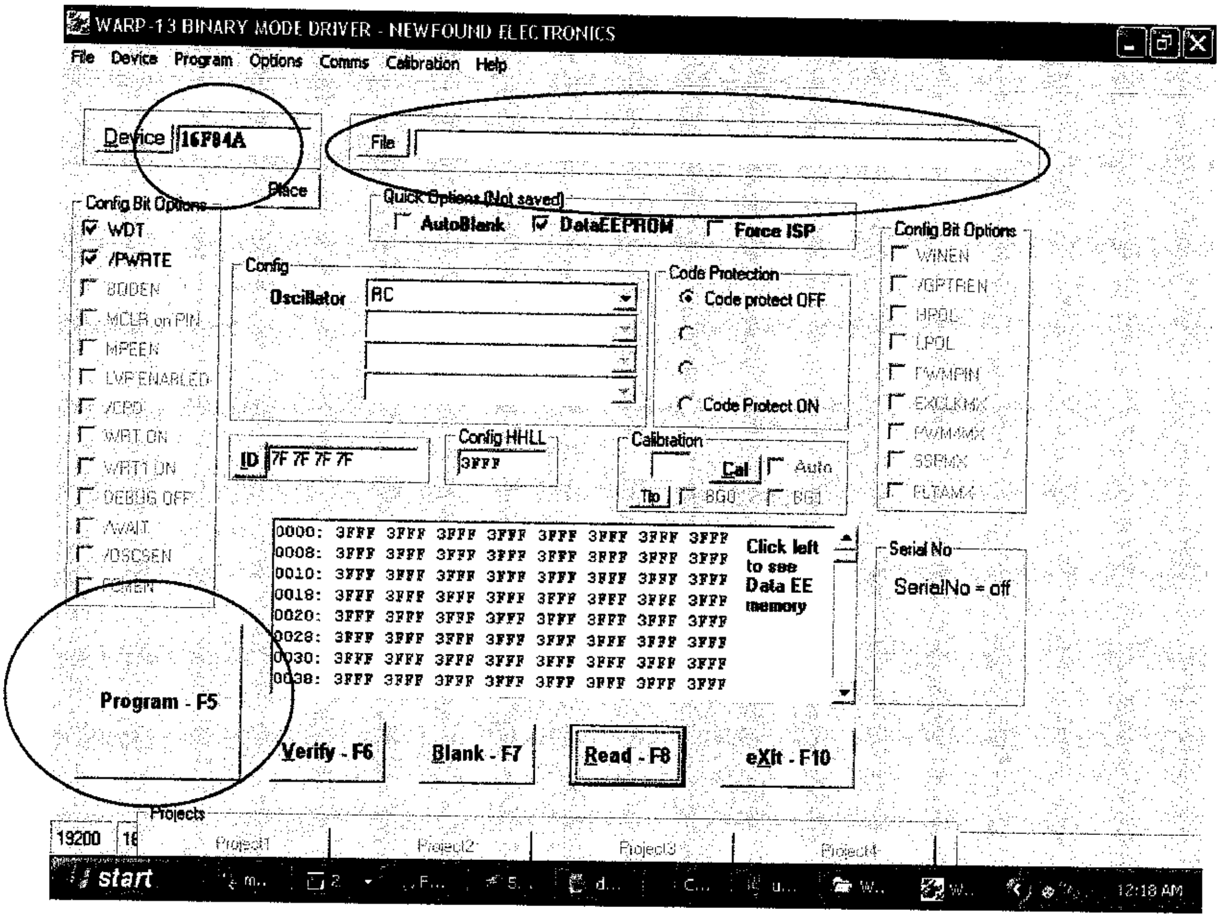
***Using the WARP13 Program***

Click on the WARP13 icon as shown below:

The main window of WARP13 program will appear as shown below




Select the device type from the device menu and choose the hex file that was build using the PIC C Compiler program. Pressing the key F5 will start the burning process.


WARP-13 BINARY MODE DRIVER - NEWFOUND ELECTRONICS
File Device Program Options Comms Calibration Help
Device 16F84A
File
Place
Config Bit Options
WDT
/PWRTE
BODEN
MCLR on PIN
MPEEN
LVP ENABLED
/CPD
WRT ON
WRT1 ON
DEBUG OFF
AWAIT
/OSCSEN
Quick Options (Not saved)
AutoBlank
DataEEPROM
Force ISP
Config
Oscillator
RC
Code Protection
Code protect OFF
Code Protect ON
Config Bit Options
WINEN
/GPTREN
HPOL
LPOL
PWMPIN
EXCLKMX
PWM4MX
SSPMX
FLTAMX
ID 7F 7F 7F 7F
Config HHLL
3FFF
Calibration
Cal
Auto
Tto
BG0
BG1
0000: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0008: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0010: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0018: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0020: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0028: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0030: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
0038: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
Click left to see Data EE memory
Serial No
SerialNo = off
Program - F5
Verify - F6
Blank - F7
Read - F8
eXit - F10
Projects
19200
Project1
Project2
Project3
Project4
start
12:18 AM

PIC16F84A coding

```
#include <16F84A.H>                          > initialize PIC16F84A
#fuses  XT,NOPROTECT,NOWDT                   > Set mode for PIC16F84A for
                                                    XT = external oscillator
                                                    NOPROTECT = No password protection
                                                    NOWDT = No watch dog
#use delay(clock=4000000)                    > Set clock speed (4MHz)

//All public value
#define delay_time 17       //ms             > Define Delay time gap for 17ms
#define delay_centre 1500   //us             > Define Delay for center position (1.5ms)
#define delay_right 2000    //us             >Define Delay for right position (2ms)
#define delay_left 1000     //us             > Define Delay for lest position (1ms)

//Function below defines the cycles per delay to obtain one second pulse.
#define cycle_right_left 112
#define cycle_centre_right 53
#define cycle_right_centre 55
#define cycle_centre_left 56
#define cycle_left_centre 55

//All functions
void initial_state(void);
void delaygap(void);
void movement_state(void);

//Start main functions
main() {
  initial_state();                           > Goto function 'initial_state'
  while (true)                                      > Set loop within 'while'
  {
    movement_state();                        > Loop 'movement_state' function
  }
}

void movement_state()                        > Coding for function 'movement_state'
{
  int i;
        for (i=1;i<=cycle_centre_right;i++)  > Loop based on defined function for 1 sec
  {
        output_high(PIN_B0);                 > Set output pin B0 to high
        output_low(PIN_B1);                  > Set output pin B1 to low
        output_low(PIN_B2);                  > Set output pin B2 to low
        delay_us(delay_right);               > Call delay timing for rotation.
        delaygap();                                 > Call delay for 17ms gap between loops
```

```
}

      for (i=1;i<=cycle_centre_left;i++)            > Loop based on defined function for 1 sec
  {

        output_low(PIN_B0);                          > Set output pin B0 to low
        output_low(PIN_B1);                          > Set output pin B0 to low
        output_high(PIN_B2);                         > Set output pin B0 to high
        delay_us(delay_left);                        > Call delay timing for rotation.
        delaygap();                                            > Call delay for 17ms gap between loops
  }

      for (i=1;i<=cycle_right_left;i++) > Loop based on defined function for 1 sec
  {

        output_high(PIN_B0);                         > Set output pin B0 to high
        output_low(PIN_B1);                          > Set output pin B0 to low
        output_low(PIN_B2);                          > Set output pin B0 to low
        delay_us(delay_left);                        > Call delay timing for rotation.
        delaygap();                                            > Call delay for 17ms gap between loops
  }



      for (i=1;i<=cycle_centre_right;i++)           > Loop based on defined function for 1 sec
  {

        output_low(PIN_B0);                          > Set output pin B0 to low
        output_high(PIN_B1);                         > Set output pin B0 to high
        output_low(PIN_B2);                          > Set output pin B0 to low
        delay_us(delay_right);                       > Call delay timing for rotation.
        delaygap();                                            > Call delay for 17ms gap between loops
  }

      for (i=1;i<=cycle_left_centre;i++)            > Loop based on defined function for 1 sec
  {

        output_high(PIN_B0);                         > Set output pin B0 to high
        output_low(PIN_B1);                          > Set output pin B0 to low
        output_low(PIN_B2);                          > Set output pin B0 to low
        delay_us(delay_centre);                      > Call delay timing for rotation.
        delaygap();                                            > Call delay for 17ms gap between loops
  }



      for (i=1;i<=cycle_left_centre;i++)            > Loop based on defined function for 1 sec
  {
```

```
            output_low(PIN_B0);                  > Set output pin B0 to low
            output_high(PIN_B1);                 > Set output pin B0 to high
            output_high(PIN_B2);                 > Set output pin B0 to high
            delay_us(delay_centre);              > Call delay timing for rotation.
            delaygap();                                  > Call delay for 17ms gap between loops
   }
}
void initial_state()                             > Initial state performed at earlier stage
                                                         program
{  int i;
          for (i=1;i<=cycle_left_centre;i++)     > Loop based on defined function for 1 sec
   {
            output_high(PIN_B0);                 > Set output pin B0 to high
            output_high(PIN_B1);                 > Set output pin B0 to high
            output_high(PIN_B2);                 > Set output pin B0 to high
            delay_us(delay_centre);              > Call delay timing for rotation.
            delaygap();                                  > Call delay for 17ms gap between loops
   }
}


void delaygap()                                  > Delay function for 17ms
{
  output_low(PIN_B0);                            > Set output pin B0 to low
  output_low(PIN_B1);                            > Set output pin B0 to low
  output_low(PIN_B2);                            > Set output pin B0 to low
  delay_ms(delay_time);                          > Call Delay 17ms
}

                                                         >Program end and loop
```