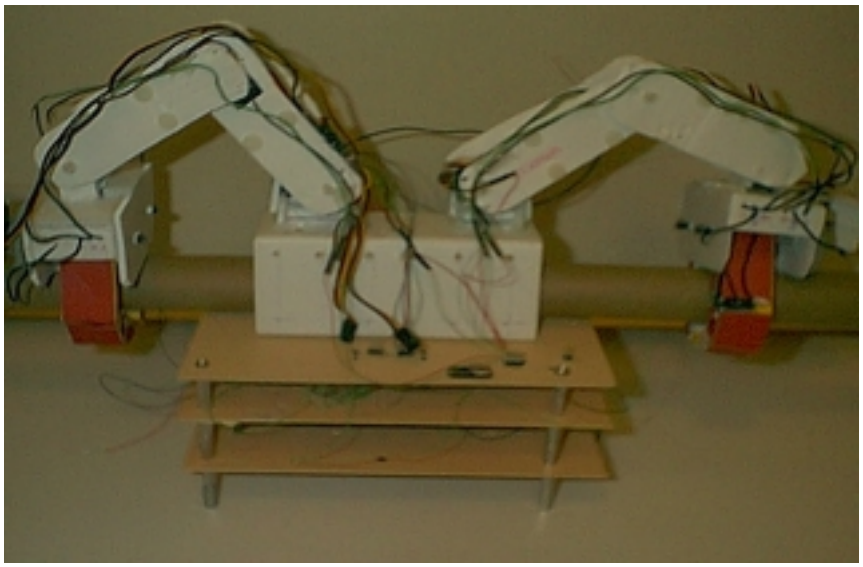
Mobile Robot on High Power Transmission Line

Prepared for:

Dr. J.F. Peters



Prepared by:

Sahar Mosleh:	**ummosleh@cc.umanitoba.ca**
Shawn Schaerer:	**subartik@mts.net**
Kian Yit Tan:	**umtanky@cc.umanitoba.ca**
Phillip Yeung:	**umyeungp@cc.umanitoba.ca**

March 8, 2001

Abstract

Robots are becoming an integral part of society. They are being used in almost every industry including mining, auto manufacturing, heavy industry and even the medical field. The overall goal of this thesis is to design an autonomous robot that crawls on a wire. Umbot 3.0 is two-legged robot that crawls on a de-energized power line avoiding the ends of the wire (obstacles) by moving front to back. It will crawl upside down on the wire and will require no extra control from any outside body. Chapter One discusses the overall control system and interactions; Chapter Two discusses the leg hardware design and leg function. Chapter Three goes through the hardware design of the robot from the micro-controller design to the power supply and Chapter Four gives the details about the sensors for obstacle detection. The Fifth Chapter goes through the design of the robot's software, including leg programming, main control and sensor algorithms.

List of Contributions

Shawn Schaerer	Control strategies; Main Board Design and construction; Communication Software; System reliability; Robot prototype and initial robot design.
Phillip Yeung	Robot design; Leg design and construction; Robot prototype; Robot simulation; Robot testing and System reliability
Sahar Mosleh:	Ultrasonic sensors and interface; Top Master programming; and formatting the final report and System reliability
Kian Yit Tan	Robot simulation and animation; Main Board algorithm and Programming; Leg algorithm; Leg programming and System reliability.

Acknowledgements

We would like to thank Dr. J. F. Peters for his guidance and support to complete this thesis. Further, we would like to acknowledge Dr. J. LoVetri who helped to organize this thesis. Many thanks also goes to A. McKay and K. Biegun from the Department of Electrical and Computer Engineering technical support for their assistance and suggestions.

Table of contents

List of Figures

List of Tables

Introduction

Any work that needs to be done on a high-voltage transmission line carries the risk of human casualties and the loss of potential revenue. The problem with current ice melting and line maintenance is the need to take a section of the high-voltage transmission line out of service in order to perform the desired task. In order to implement a legged robot to eliminate the loss of time and human casualties due to high-voltage; transmission lines will require great amounts of investment and research.

The previous ancestors of this robot, UMBOT 1.1 and UMBOT 2.1 used a six-legged HEXAPOD design. These robots are simply too large and too confined for the task of moving across a power line. The legs in these robots are too weak and thin so that they cannot be used. The next generation, UMBOT 3.0 uses a new leg system that is specified for this task.

The purpose of this thesis is to develop a robot that is capable of moving and exploring on the power line wire. UMBOT 3.0 is implemented with a roller design that helps the robot slide freely across power lines. Two legs are also implemented on the robot so that it is capable of pulling itself forward and backward. Grippers are used so that the robot has a strong grip on the power line for pulling. The robot uses a very complex layered structure where the master controls each component of the robot. Ultrasound sensors are used to avoid obstacles such as the insulators.

There are many factors and assumptions that were considered while developing this robot. One of the main assumptions is that the robot will be placed on a de-energized high-voltage transmission line under room conditions. Power lines in real life are fully energized, meaning that a small movement on the power line could cause a big electrical shock to the object. Since the team has limited skills and background on electrical engineering, this assumption had to be acknowledged before the project began. A conventional power supply will be used instead of constructing an induction based one (for an energized line). When an obstacle is found, the robot reverses direction and continues exploring the area. It does not need to travel around the

obstacle, as this gets very complex. The main focus will be developing and implementing the moving motion of the robot along a de-energized high-voltage transmission line.

This thesis report is divided into sections that describe the designs and methodologies used throughout the project. The robot control system is described in chapter 2 of this report, followed by discussion of the mechanical design in chapter 3. Chapter 4 introduces the background of the servos. Chapter 5 illustrates the fundamental properties of ultrasonic sensors and its calibration procedures. Chapter 6 describes the main board design; the inter-processor communications and power supply design. Chapter 7 goes through the design of the software implementation.

Chapter: 1 Control System

This chapter discusses the way the robot interacts in its environment with its various subsystems.

1.1 Control system approach

The control system of this robot is based on a multi-layered abstract control system (Brooks,1986). Layers allow the robot to increase its level of competence, by allowing each layer to perform its own computations. Each layer has the ability to communicate with the other levels and the ability to perform its own individual task with specific behaviors. The system is abstract in the sense that each subsystem is independent and will behave in a specific manner regardless of the hardware or software that it consists of. An example of abstraction is in the function of the robot's leg. If the leg is commanded to move it should not matter to the commanding subsystem that the leg is a four degrees of freedom type or that it is wheel based. This allows for quick adaptation of the robot to different environments by simply changing some its basic hardware, not its complex behavior strategies.

The robot is designed around three layers. The top layer, which is responsible for object detection, the middle layer that decides what action should take place to a specific situation and the bottom layer which is responsible for controlling the movement of each leg.

The top layer will receive object detection data from the ultrasound sensors. It will send the sensor data to the middle layer via a communication link where it will then send the data to the appropriate leg. The main responsibility of the top layer is object detection. The bottom layer will receive commands via the middle master and move the legs in the specified direction until they are commanded otherwise.

Each of the layers is coordinated through the middle master layer. The middle master acts as a router and central nervous system of the robot. It performs the main control subroutines by coordinating the movements of the legs and responding to object detection signals from the top master.

1.2 Algorithm design

The basic design of the overall function of the robot can be described in the state chart of figure 1 (Peters & Pedrycz, 2000, “Software Engineering An Engineering Approach”, figure 7.9)

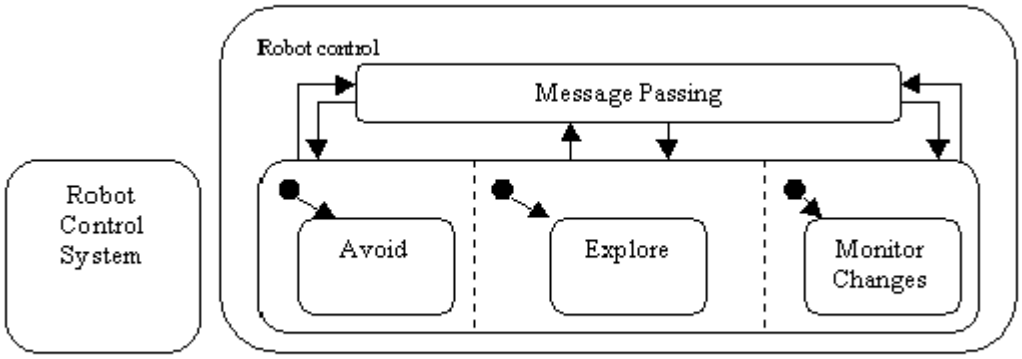


Figure 1: Basic Design of the overall function of the Robot

Chapter: 2 The Mechanical Design

The initial design of this robot was a HEXAPOD type robot meaning that it is capable of moving on the power line with only six legs. This design mimics the action of a spider crawling upside down on its web. This type of action requires a lot of leg strength when it is pulling itself up and down. It also requires a great deal of agility when it extends its legs out to move forward and backward. This means that the body must not be bulky or heavy. The initial design was very difficult because, it was very hard for the robot to hang on to the wire while it moved forward and backward with six legs attached. This would mean that each leg must have three joints or more so that it can move in three degrees of freedom, including the action of gripping. Making sure the robot would travel smoothly was very challenging. A total of eighteen servomotors would have to be used for this design, which would seriously affect the project's budget constraints. Due to these problems associated with this leg design, the concept of the robot leg was changed completely to overcome these initial problems.

2.1 Roller Design

Out of the various alternative ways of implementing the leg design, the roller system is the best method. The idea of the roller system is to allow the robot to be attached to the power line permanently, staying balanced while the legs pull the body forward and backward to maintain stability. One of the big advantages of the roller system is that reduces the number of legs that is required to balance on the power, therefore cutting the cost of the robot. Another big advantage of the roller system is that the roller helps the robot balance on the power line, while the legs are reaching forward. This is very beneficial to the design, so that while the robot is moving it does not fall off easily. Another advantage comes when the robot is going down a decline. At this moment, the robot legs may sit idle, allowing the roller to freely slide down and conserve energy. There is one disadvantage of the roller design. Since the robot must move around obstacles such as insulators in the real world, the roller itself must detach from the power line and attach back to the power line when the insulator is avoided. This task is difficult, as it will require the roller to detach and reattach itself from the line using a locking system to prevent the robot from falling off. This does not pose a problem because the project neglects the presence of insulators on the power lines.

The main function of the roller is to allow the body to move freely across the power line. It does not have the power to move the body without the legs, because it is simply attached to the body on the power line. The purpose of the roller is to provide as little friction as possible for the robot to move on the line. The best method to build the roller is to use durable material that is light and easy to attach to the body. White plastic sheets are used for the frame of the roller because they are very light and have a very durable property. This material is very suitable for building robots.

The roller is a rectangular shaped box with dimensions L=15cm X W=6.5cm X H=6.5cm attached to the top board of the robot. Roller pins are installed inside of the roller. There are six roller pins going across the top part of the roller. There are also two roller pins installed at the sides of the interior roller frame. For each roller pin, there are four metallic beads attached. These beads roll on the power line when the power line is attached to the roller. The pins on the side of the roller keep the power line centered in the roller at all times. The roller system is shown below in figure 2:

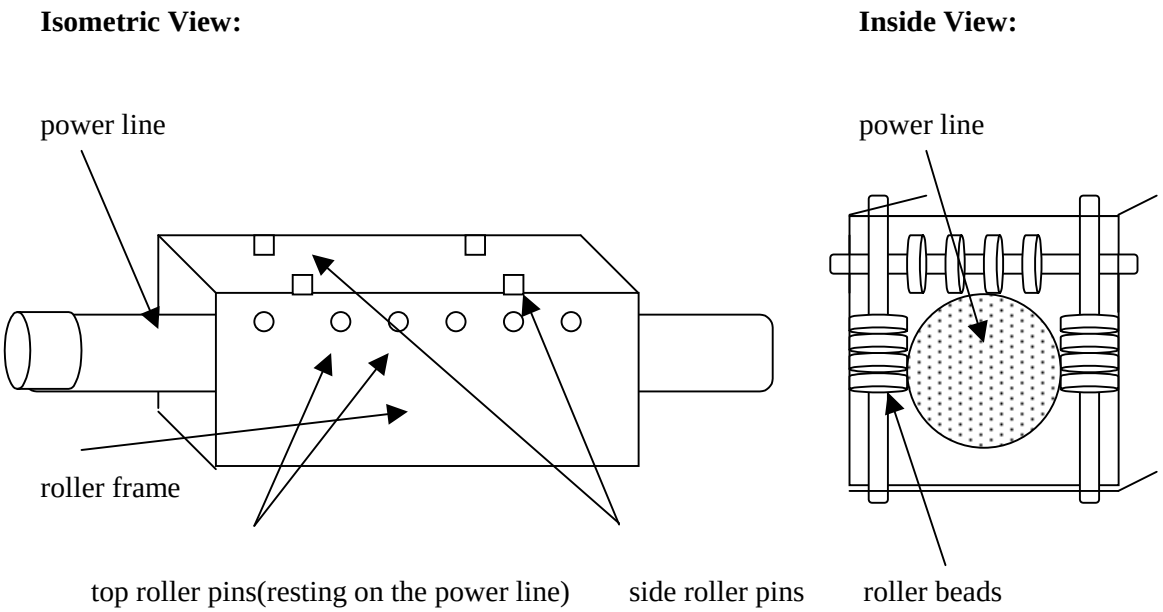


Figure 2: roller design

The beads are placed between the ends of the pin notches to prevent them from sliding off, or out of place. The roller frame was assembled together with hot glue gun and hot glue.

2.2 Leg Design # 1

There are many ways to implement the legs given the roller system used in the robot. The legs can either push forward or pull forward. The primary concern when designing the leg is its strength and reach. The legs must be strong enough so that the entire body of the robot can be moved when the roller is attached. This can be done with both strong leg joints and multiple legs pulling at the same time. The more legs you have in the system, the stronger the robot will be, but it will become more complex. Therefore, there is a trade off between strength and complexity of the design as the stronger the leg needed, the more complex the system becomes.

Secondly, the legs must have a desirable range in reach. The further the leg can reach forward, the faster the robot can move. This means a reduction in leg steps, hence reducing movements on the power line. The bigger the reach, the longer the leg segments, which means the leg becomes heavier and the joints must endure more stress. So there is a trade off between leg reach and leg weight.

After considering all of these factors, the first leg design was built. In the first leg design, there were four legs, each leg having two segments and two joints that moved the body by pulling itself forward. The segments were built with white plastic sheets that were secured with cross members. The cross members are hard plastic tubing that allows two pieces of plastic sheets to be attached together. The use of cross members along with the plastic sheets makes the segments very strong and durable. The use of plastic sheets and cross members is shown below in figure 3:

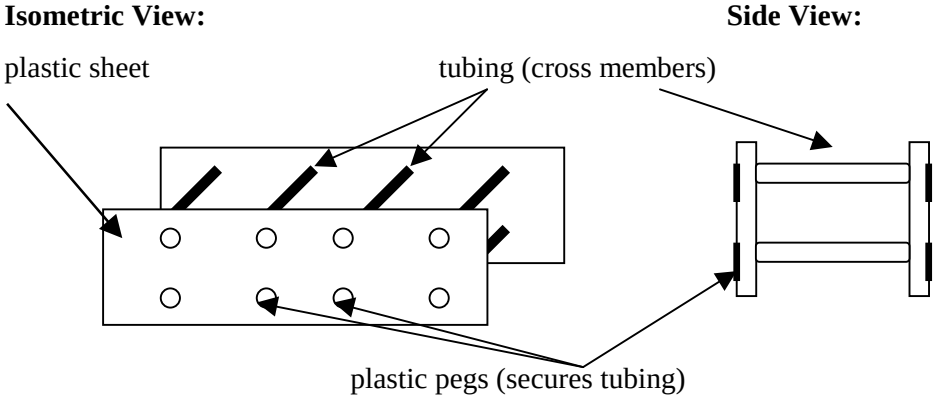


Figure 3: leg structure

There were two leg segments for each leg, the first segment being half the size of the second allows the leg to turn forward and backward. A servo was mounted on the body to allow this type of turning motion. The second leg segment was responsible for gripping and grabbing on to the power line. The joint simply spins this segment against the power line with the servo

mounted on the first segment. The joints are powered by HOBBICO CS-35 high power mini BB servos, capable of delivery a 4.00kg-cm torque when 5V is applied. Each servo weighs about 27g with dimensions of 32mmX17mmX31mm. Together with the movements of the first joint, the leg was able to provide a pulling action that could pull the body forward. A rubber material was attached at the tip of the second segment so that the grip was more powerful. The main advantage of this leg design was its simplicity. Since each leg had only two joints and two segments, the microprocessor only needed to control two servos at max, thus making this design very simple. Another advantage of this leg design was its ability to reach very far outward, requiring fewer movements each time, which was very important. There was one big disadvantage. The legs were not providing a strong enough pull because there were no grippers attached to the end of the leg. This problem was the main reason why the leg had to be redesigned in the later part of the project. Picture of leg design #1 is shown below in figure 4:

Isometric View: **Body Top View:**

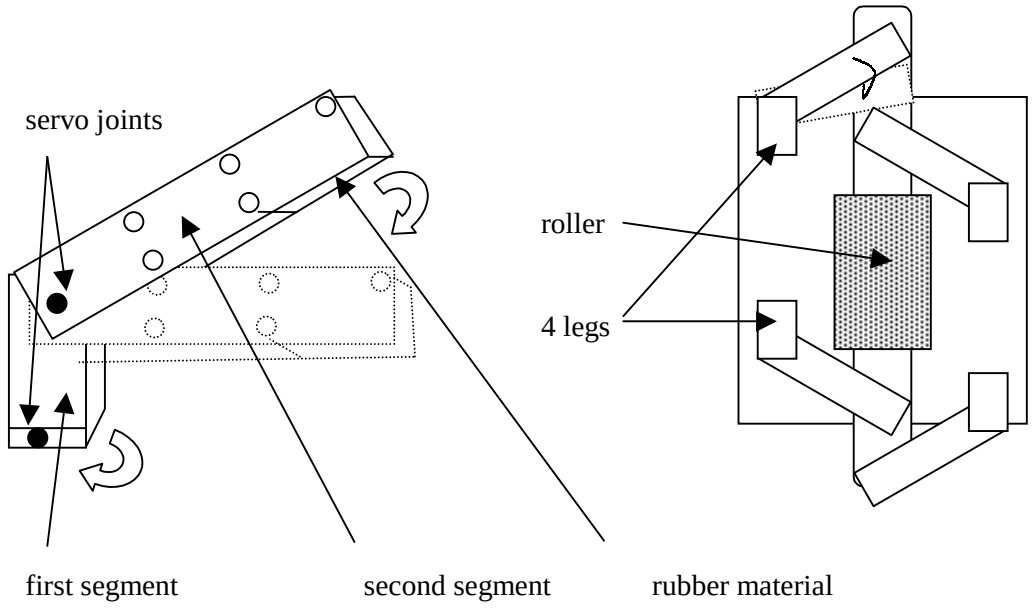


Figure 4: first leg design

2.3 Leg Design # 2

In the middle stage of the project, leg design #1 was tested on the power line. The results were not anticipated. This is mainly because the grip in the second joint was not strong enough to pull on the power line. A different method of the leg design had to be developed so that the

robot could have a strong grip. A gripper was designed to incorporate this task. The gripper design acts like a hand that grabs on the power line as it pulls itself forward. This design was clearly better than the previous design because the gripping is now more powerful.

The gripper has two fingers powered by a servo. Each finger is made with hard plastic for durability. There is some rubber material on the fingers to make sure the grab does not slip away. The frame of the gripper is built with white plastic sheets. The purpose of the frame is to protect the fingers from slipping away. There are two micro switches installed in the gripper, one at palm of the gripper and one attached on one of the fingers. The micro switch at the palm tells the microprocessor that the gripper has reached the power line. The microprocessor then signals the gripper to grab by clamping the two fingers together. The micro switch on the finger is to let the microprocessor know that the grab is successful. The microprocessor will then signal the legs to pull forward. There are two main gears associated with the movements of the fingers as shown below in figure 5:

Isometric View: **Side View:**

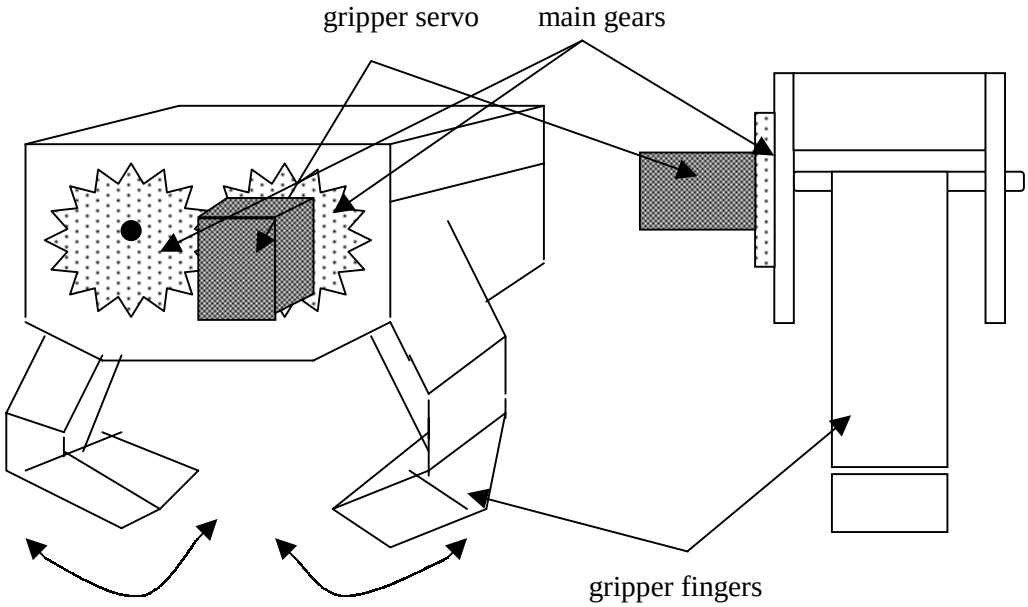


Figure 5 grippers

There are two leg segments attached to the gripper. These leg segments help to extend the gripper forward and backward, providing a pulling movement. There are two of these gripper legs, where each leg will be mounted on top of the roller. There are two leg joints. The first joint is attached on the roller and the second joint is attached on the first segment. There are three servos on each leg. Two are used in the joints and one is used for the gripper action. Each

segment is approximately 7cm and the maximum reach is approximately 17cm. The complete gripper leg is shown in figure 6:

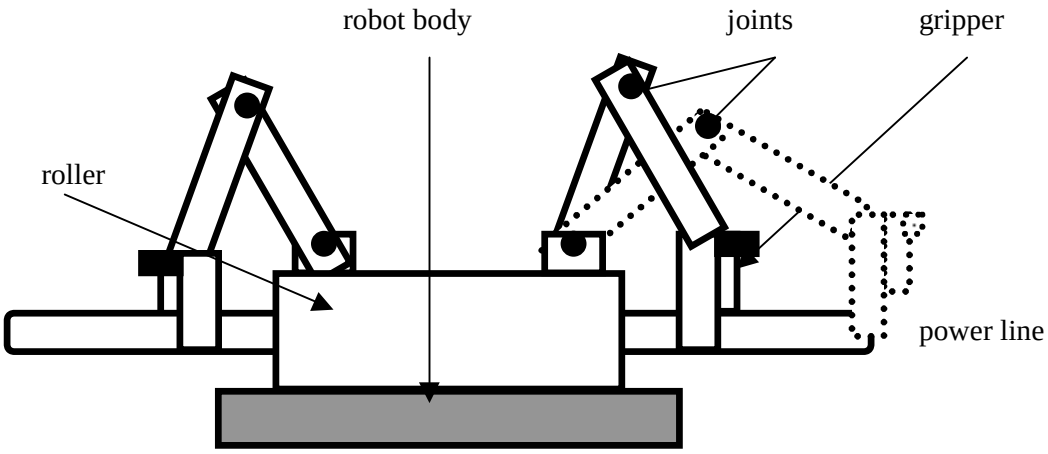


Figure 6: leg design #2

One of the advantages of this design is that it is strong enough to pull the entire body forward with one leg at a time, thus reducing number of legs and reducing the cost. Because the number of legs was reduced, the complexity of the programming required was also reduced. The forward and backward movements contain the same programming code. While the front leg is responsible for moving forward, the back leg is responsible for moving backward. The coding can be duplicated. The non-moving leg hangs loose above the power line while the other leg drags its body forward/backward. Another advantage of this is the length of the reach. Because the leg is comprised of two long segments, the reach is increased. Since the reach of the leg is far, wireless video cameras and heat guns can be attached to the tip of the gripper. This can help the user monitor the conditions on the surface of the power line and use a heat gun to de-ice ice patches.

2.4 Final Leg Design

The final leg design is just a small revision of the second leg design. The difference between these two leg designs is that the final leg design has an extra hand wrist joint. This extra wrist joint is very important because it helps the gripper to align perpendicularly on the power line. When this wrist joint is not installed, the leg cannot put the gripper on the power line

perpendicular with the center of the gripper. This could cause a scooping action when the legs are pulled. The final leg design is shown below in figure 7:

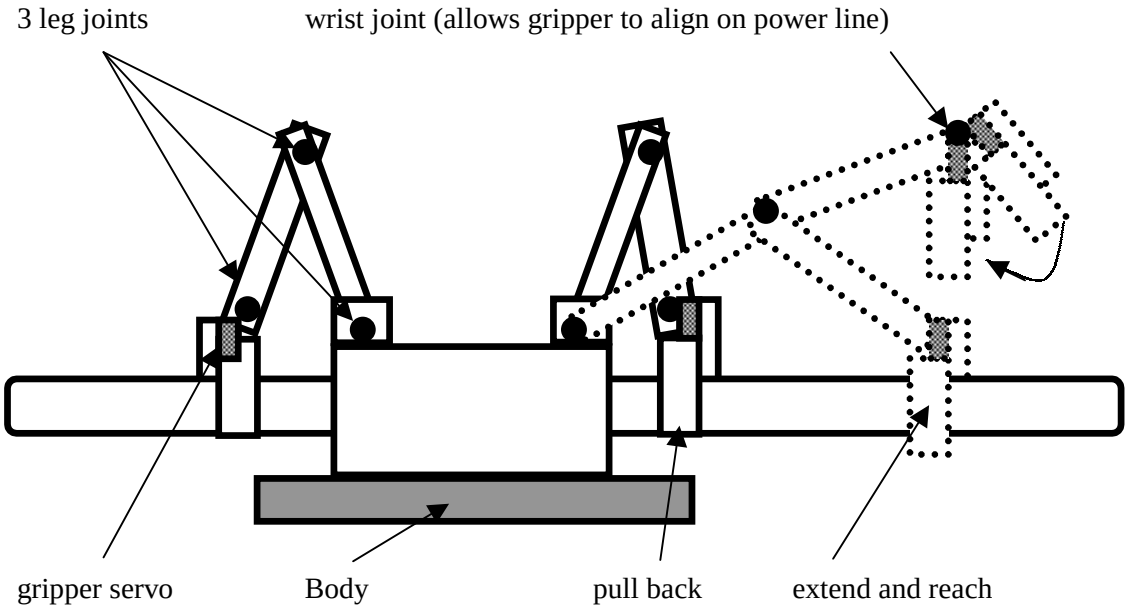


Figure 7: final leg design

Chapter: 3 Servo motors

Servomotors are geared dc motors with positional control feedback. They are used for position control. The angles in which the servos rotate are controlled by pulse width modulation (PWM), which is provided by the output of port B. There are many ways to produce the PWM. One of the ways is by using Port A which generates 32 ms pulses (assuming an 8 MHz XTAL) in which this is the natural roll-over time of the free-running clock (FRC). Pulses were generated using PORT B because the frequency of 32 ms was too large and thus the servos were very weak in strength. Producing the PWM using assembly language was made easy using "Jonathan W. Valvano's Microcomputer Simulator" in which the results could be simulated. The timing diagram can be observed by using the oscilloscope in the simulator.

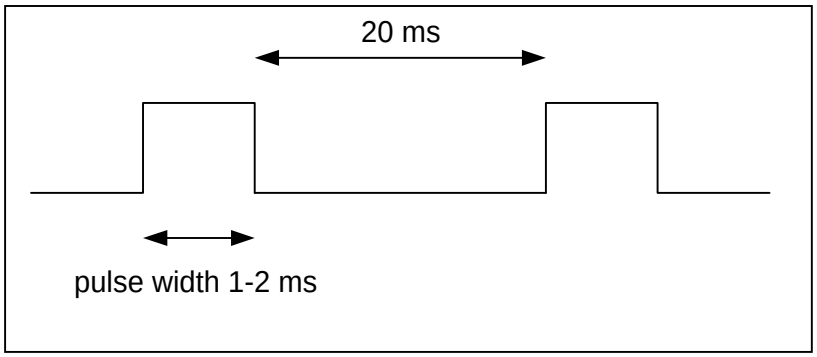


Figure 8: Timing diagram of servos

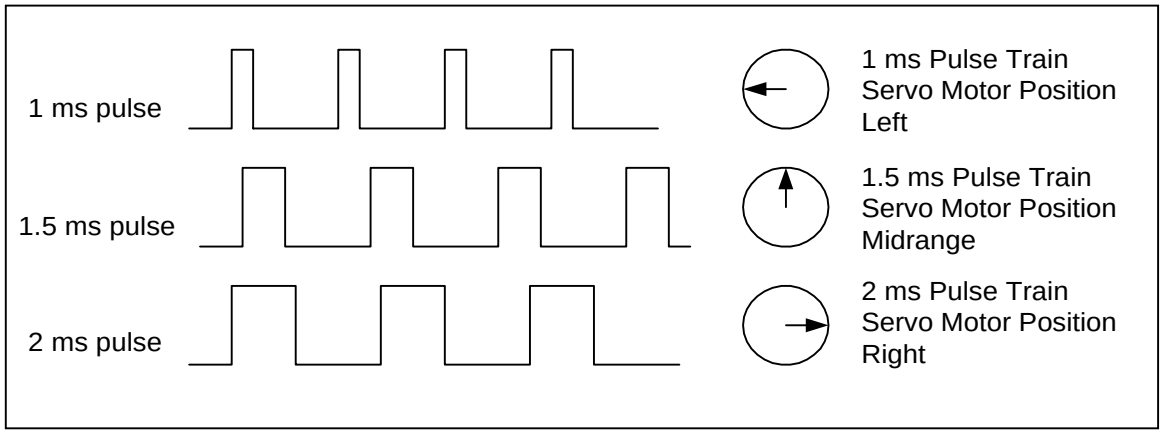


Figure 9: Timing diagram affects position of servos

A pulse width of 1-2 ms would provide counter clockwise to clockwise rotation. The ideal frequency to provide. This would be 20 ms if the frequency were less than 18 ms, decoding failure will result and thus, the servo will fail to respond. Large frequencies will result in servos moving with a jitter, thus providing slower movement but sacrificing torque and strength.

High-pulse (ms)	1.0	1.25	1.33	1.50
Position (deg)	0	45	60	90

Table 1: High-pulse (ms) versus position (deg) of servos

The three-wire connectors for RC-servos are as follows:

Pin 1 = White = Signal
Pin 2 = Red = +5 V
Pin 3 = Black = Signal + Power Ground

The advantage of using the RC-servo is that using PWM can easily control the angles. However, the speed of the rotation of the servos cannot be controlled. There are some suggestions on controlling the speed of the rotation of the servos by increasing the frequency of the pulse widths. However, this will cause the servo to lose its strength and become unable to lift the leg of the robot.

All 4 servos of the robot leg have to be pulsed simultaneously to maintain control. If any of the servos are not pulsed, the servo will lose control and will cause the leg of the robot to fall.

The positions of the legs of the robot are based on the 0 position legs as shown in the figure below.

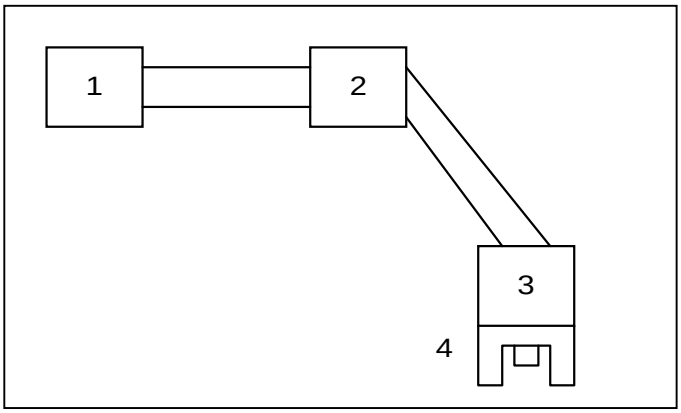


Figure 10: Zero position of robot leg

There are 5 major positions in the legs of the robot. The legs of the robot are given 1-second intervals to align themselves. The 5 major positions of the legs of the robot are as follows:

Position 1

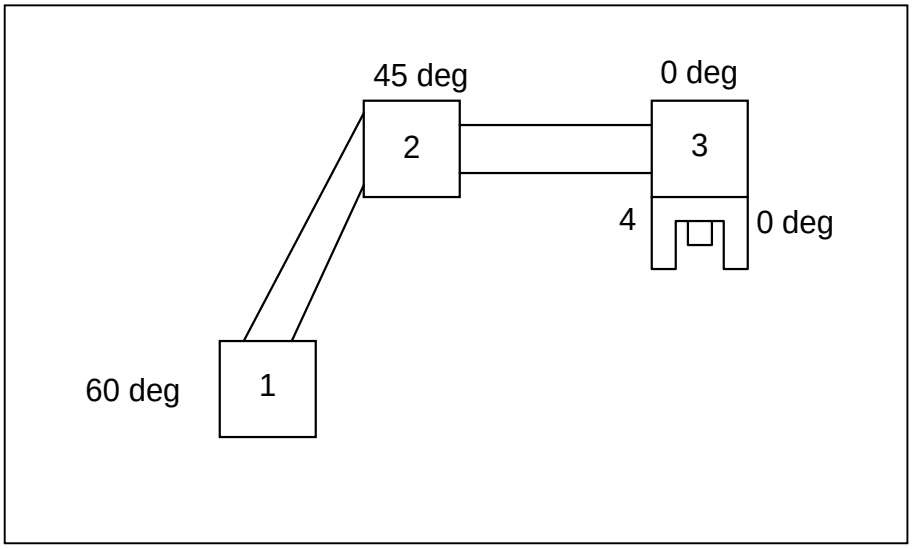


Figure 11: Position 1 of robot leg

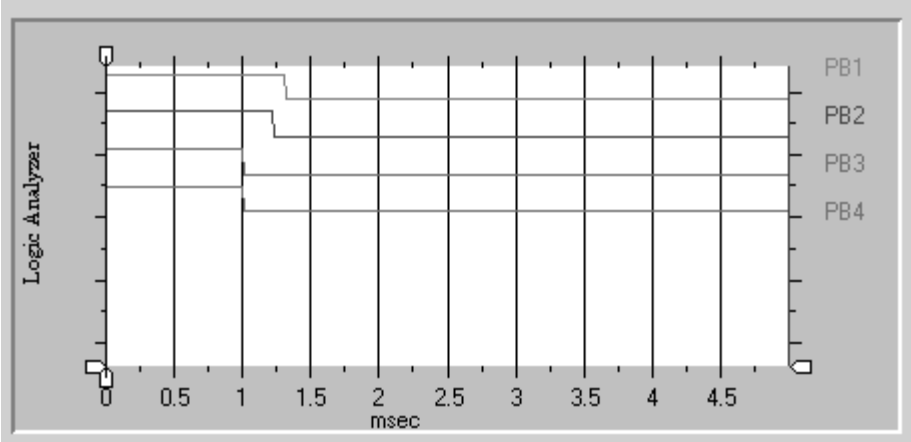


Figure 12: Output of Port B for position 1 of robot leg

The leg of the robot is raised and prepared to grab the line.

Position 2

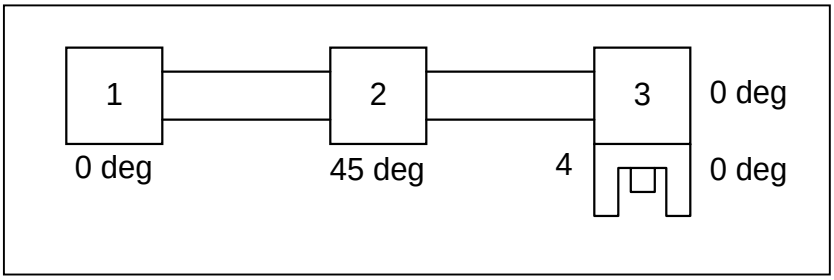


Figure 13: Position 2 of robot leg

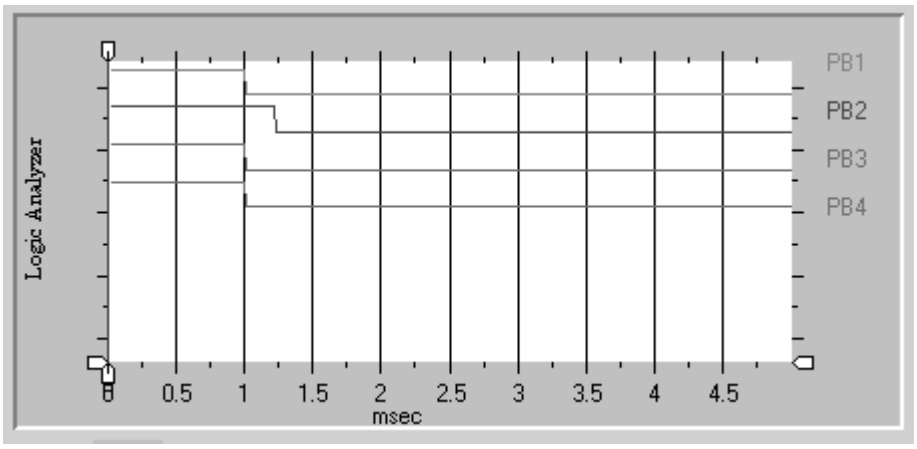


Figure 14: Output of Port B for position 2 of robot leg

The leg of the robot reaches forward to grab the line. It will not stop until the bump sensor is activated upon contact with the line.

Position 3

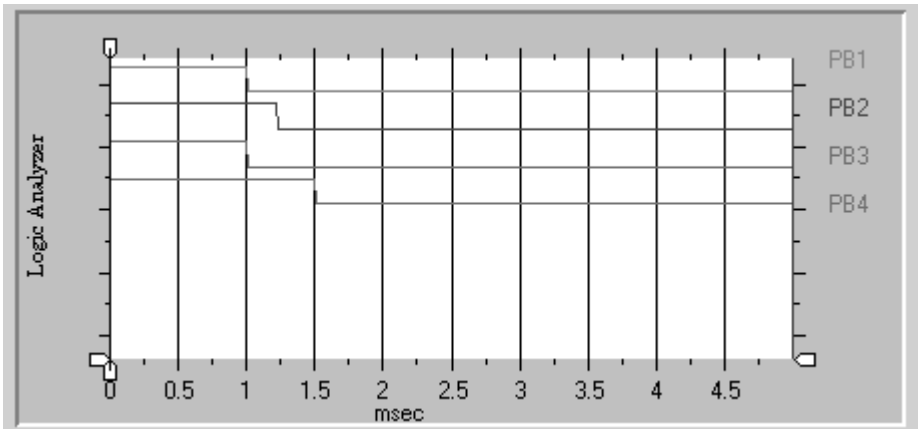


Figure 15: Output of Port B for position 3 of robot leg

The clamp of the robot leg is closed. The clamp of the robot leg will not stop closing until the bump sensor is activated. This is to make sure that the clamp has gripped the line.

Position 4

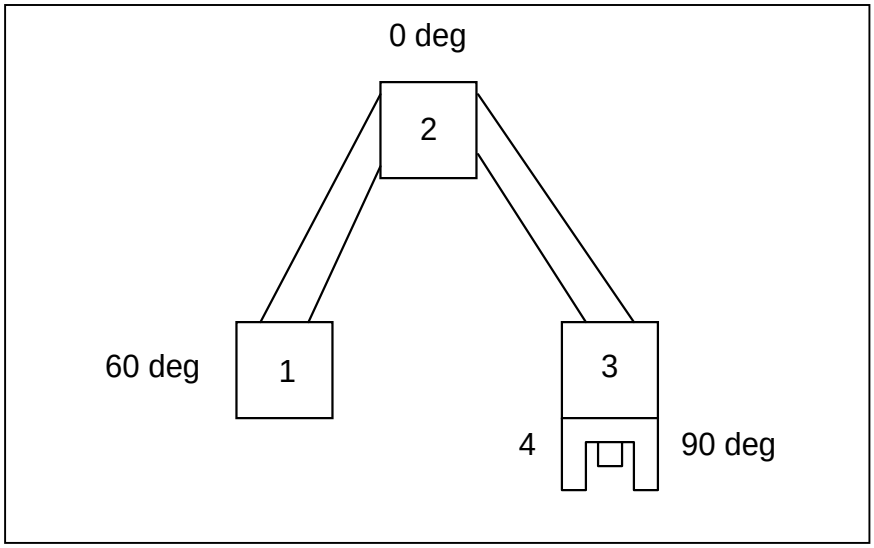


Figure 16: Position 4 of robot leg

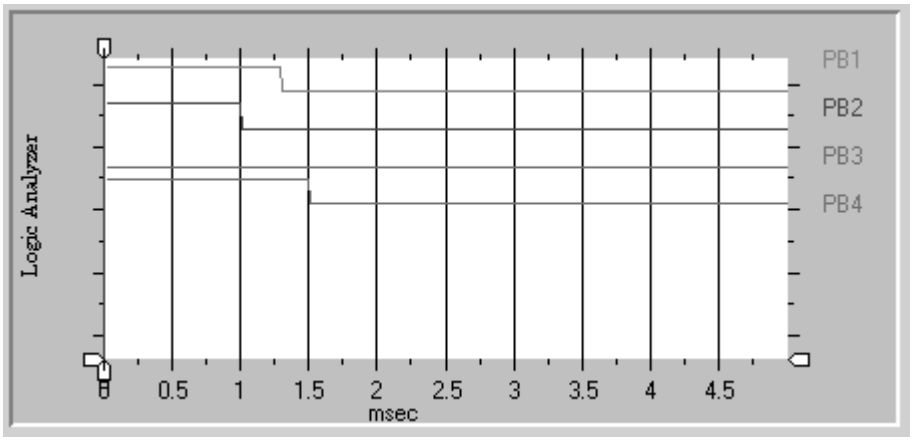


Figure 17: Output of Port B for position 4 of robot leg

The leg of the robot pulls the body forward or backward depending on where the leg is positioned. Servo 3 is deactivated so that it will be able to negotiate the angle required to pull the body of the robot forward.

Position 5

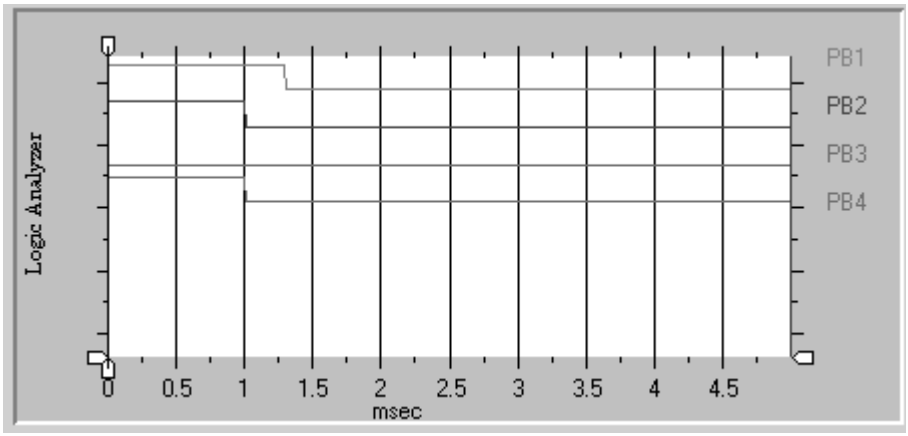
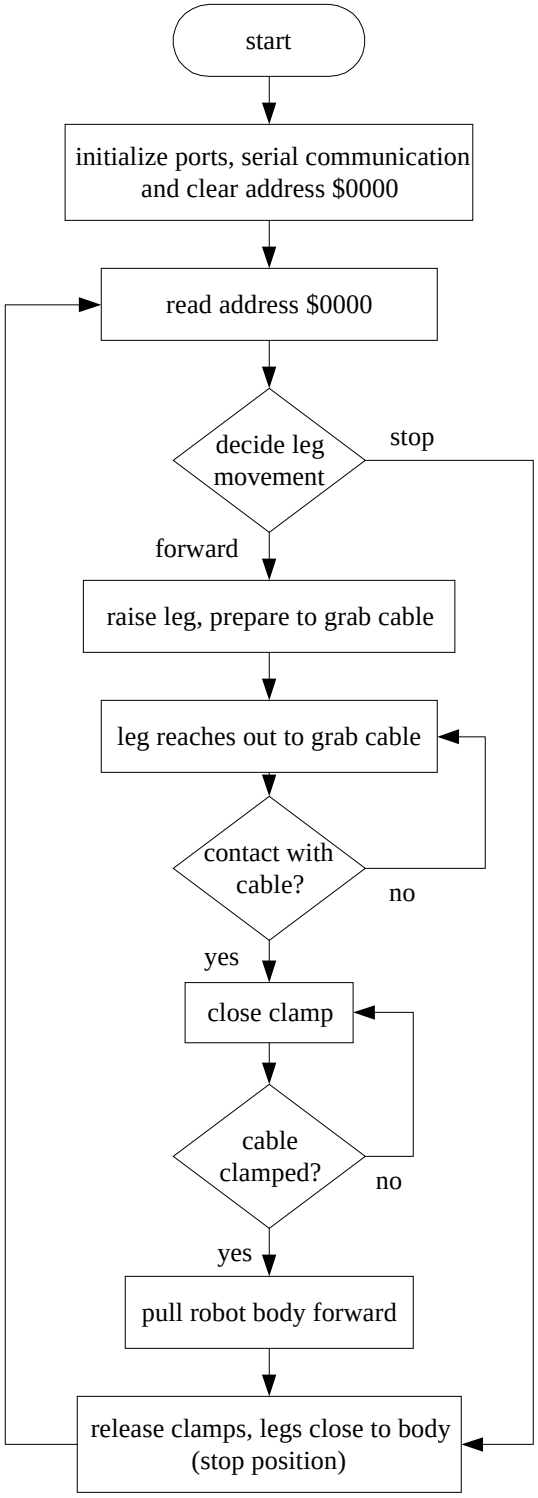


Figure 18: Output of Port B for position 5 of robot leg

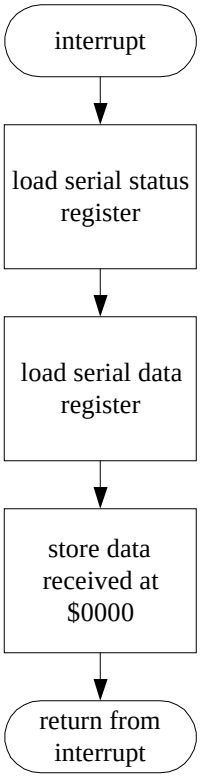
The clamp of the leg of the robot is released. This is also known as the stopping position because if the leg of the robot receives command from the master controller to stop, this will also be its position.

The leg of the robot receives commands from the master controller through the receive (RX) line. Whenever a command is transmitted to the leg of the robot, it will be stored in memory location $0000. The main program will poll memory location $0000 to see what the master controller wants it to do. Thus, there is a weakness shown here. The robot will not be able to stop immediately. It will have to finish its step before it is able to stop.

3.1 Flowchart of the Code to Control the Leg of the Robot



3.2 Interrupt Routine of Leg of Robot



Chapter: 4 Overview of Sensors

The important feature of the robot is the sensors. The sensors provide the robot with the information about the surrounding environment. Although more than one type of sensor are applicable for this robot, the sensor that has been used for this project is an ultrasonic sensor for obstacle detection.

This chapter starts by elaborating on the features of the ultrasonic sensors. It introduces the components of the sensor and discusses its main properties. It then presents the implementation. This includes the sensor calibration and the software used to interface the micro-controller.

4.1 The framework of Ultrasonic sensors

Ultrasonic sensors are commonly used for a wide variety of proximity, or distance measuring applications. These devices typically transmit a short burst of ultrasonic sound toward a target, which reflects the sound back to the sensor. The system then measures the time for the echo to return to the sensor and computes the distance to the target using the speed of sound in the medium.

4.2 Sensor Properties

Ultrasonic sound is a vibration at a frequency above the range of human hearing, usually greater than 20 kHz. The microphones and loudspeakers used to receive and transmit the ultrasonic sound are called transducers that operate at frequencies between 40 kHz and 250 kHz. The following section evaluates the quality of ultrasonic sensors based on two factors: the detection range of sensor and the type of detectable targets.

4.2.1 Detection range of sensor

The range of detection depends on the attenuation of sound as a function of frequency and humidity. As the sound travels, the amplitude of the sound pressure is reduced due to friction in the transmission medium (air friction). Knowing the value of this absorption loss, or attenuation,

is crucial in determining the maximum range of a sensor. The attenuation of sound in air increases with the frequency, and at any given frequency the attenuation varies as a function of humidity. The value of humidity that produces the maximum attenuation is not the same for all frequencies. Above 125 kHz, for example, the maximum attenuation occurs at 100% relative humidity (RH); at 40 kHz, maximum attenuation occurs at 50% RH.

4.2.2 Type of the target

There are two types of target to detect: a flat surface target (for example a liquid surface), and non-flat surface such as sphere, cylinder, ellipsoid, etc. When the sound pulse is reflected from a large flat surface, then the entire beam is reflected. (See Figure 19):

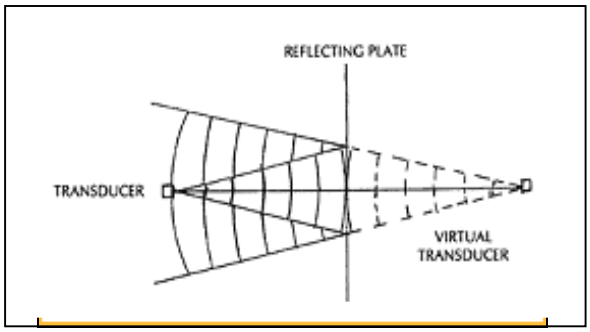


Figure 19 A sound beam reflected from a flat surface is equivalent to the sound as generated from a virtual transducer at an equal range behind the reflecting plate.

This total beam reflection is equivalent to a virtual source at twice the distance. Therefore, it is important that the reflecting surface be larger than the entire sound beam to ensure total reflection. In addition, the target must be perpendicular to the sound beam.

For the non-flat type of the target, the echo levels are affected differently. Figure 20 illustrates the behavior of a small sphere as a target. It is shown that the sphere intercepts only a portion of the sound beam and then reradiates the sound pulse.

The radiation of the sound pulse depends on Target Strength (TS). Target Strength is the measure of the reflectivity of a target. It is defined as 10 × the logarithm to the base 10 of the intensity of the sound returned by a target at a reference distance from its "acoustic center," divided by the incident intensity of the transmitted sound pulse.

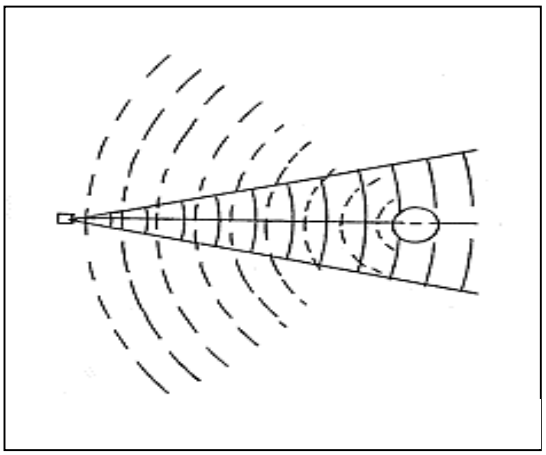


Figure 20 **Error! No index entries found..**

The TSs of simple geometric shapes can be theoretically computed; Table 2 contains the expressions of TS for a few types of target forms. When using this table, all dimensional units must be the same, including the reference range, R_0, the range distance to the target, R, and all dimensions of the targets.

Form	t (TS = 10 log t)	Definitions	Direction of Incidence	Conditions
Sphere	$a^2/4$	a = radius of sphere	any	ka>1 R>a
Cylinder, Infinitely Long	aR/2	a = radius of cylinder	normal to axis of cylinder	ka>1 R>a
Cylinder, Finite Length	$aL^2/2\lambda$	L = length a = radius	normal to axis of cylinder	ka>1 $R>L^2/\lambda$
Smooth Convex Object	$S/16\pi$	S = total surface area of object	average over all directions	All dimensions $>\lambda$
Ellipsoid	$(bc/2a)^2$	a,b,c = semimajor axes of ellipsoid	normal to major axis	ka,kb,kc>1 R>a,b,c

Table 2: Theoretical Target strength for simple Forms. Ro= reference range; k=2p/l; R=range to target

Such idealized computations of TS should be used only as an approximation of real targets, since actual targets are usually not simple reflectors but rather are complex with multiple surfaces of reflection. The sound reflecting from each of these multiple surfaces will produce echoes of different amplitudes that will sum together when they return to the sensor. Since the sound pulse is reflected at different times by the various reflecting surfaces as it propagates across the target, the individual echoes will be different in both amplitude and phase. The total received echo will therefore be a complex summation of these multiple pressure waves of different amplitudes and phases.

4.3 Sensor Mathematical model

To measure distance, a short burst of ultrasonic sound, more specifically 40 kHz sound, is sent out through a transducer. The sound bounces off an object and another transducer receives the echo. A circuit then computes the time it took between the transmit pulse and the echo and calculates the distance as follows:

$$L_o = (vt \cos \phi) / 2$$

Where v is the speed of the sound in the media (for air $v = 330$ *m/s* or $v = 13{,}030$ *in/s*), t represents the time that takes the ultrasonic beam to hit the target and bounces off (round trip), and ϕ is the angle that the beam produced by the transmitter makes with the x axis (see Figure 21).

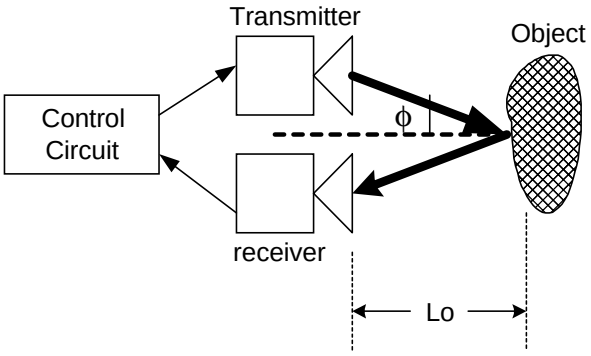


Figure 21 Basic arrangement of the ultrasonic distance measurement

4.3.1 Architectural model

Figure 22 illustrates the general architecture of an ultrasonic sensor. It is composed of many small blocks. Each block is a simple circuit with few components. First a 5.0688 MHz crystal generates a precise timing signal. Then it is divided into 39.6 kHz by a 4040 ripple counter IC. This counter divides up 5.0688 MHz (5000 kHz) by 128 that give the result of 39.6 kHz that is closed to desired 40 kHz. The output of 4040 ripple counter is sent two ways. Considering the transmitter section, the 39.6 kHz signal is applied to one half of a NAND gate used as a pulse gate. The other half of the NAND gate is connected to a pulse generator trigger. This generator sends out a timed pulse that allows the NAND gate to pass the 39.6 kHz tone for a specific amount of time (roughly 400 μs). The output of the NAND gate is not powerful enough to drive the transducer directly. Therefore, the signal is first boosted by a driver transistor and then it is passed through a band pass filter to eliminate the background noise. The result is a 400 μs burst of ultrasonic sound from the transducer.

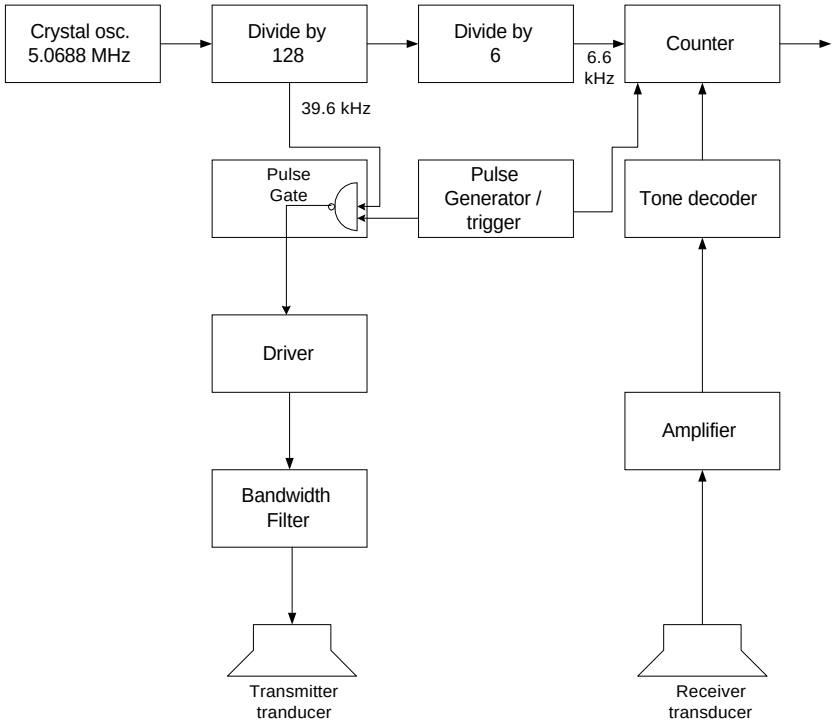


Figure 22 The block diagram of ultrasonic measuring system

When the generator sends out its pulse, it also triggers a counter to reset and starts counting. This counter is clocked at a precise speed by a 4017 IC hooked up to divide the 39.6 kHz ultrasonic tone by 6, resulting 6.6 kHz beam that is roughly half of the speed of sound (in inches)

at sea level (165 m/s or 6600 inch/sec). Since the frequency is halved before it gets to the counter, there is no need to divide the resulting time difference later on to account for the round trip of the ultrasonic burst. Each tick of the counter represents one inch, which means that this system is fundamentally limited to accuracy of one inch or less. This accuracy is suitable for most applications.

The counter will count continuously. If there is any echo pulse, it will be captured by the receiver transducer, which will stop the counter. The reverberation signal is very weak. Thus, it must be amplified. A 567- tone decoder IC is wired up to listen only for sounds that occur at about 40 kHz. Therefore, when the echo pulse is received, the tone decoder latches onto it. And changes its output. At this time, the counter is stopped and the distance in inches is displayed.

4.4 Implementation

This section details the sensor calibration procedure and the program developed to interface the sensor with the micro-controller (top master) .

4.4.1 Calibration Procedures

- The calibration procedure of the ultrasound sensor used is as follows:
- Apply power to the unit
- Allow several minutes warm up time before calibration.
- Set gain control to 50% (See Figure 23)
- Connect a DC voltmeter to sensor: + lead to pin 6, - lead to pin 2.
- Rotate the zero adjust pot fully counter-clockwise (12 turn pot).
- Place target at maximum desired distance. Adjust full-scale pot to 5 VDC.
- Place target at minimum desired measurement distance. Rotate zero adjust pot clockwise to 0 VDC.
- Test the 0 and 5 volt settings by slowly moving the target from minimum to maximum positions. If minor adjustments are required, always adjust the full-scale pot first, and then the zero adjusts pot.

- To calibrate the gain setting, place the target at the maximum desired detection distance. Rotate the gain control fully counter-clockwise; slowly rotate gain control clockwise until detection occurs. Rotate an additional 1/16 turn.

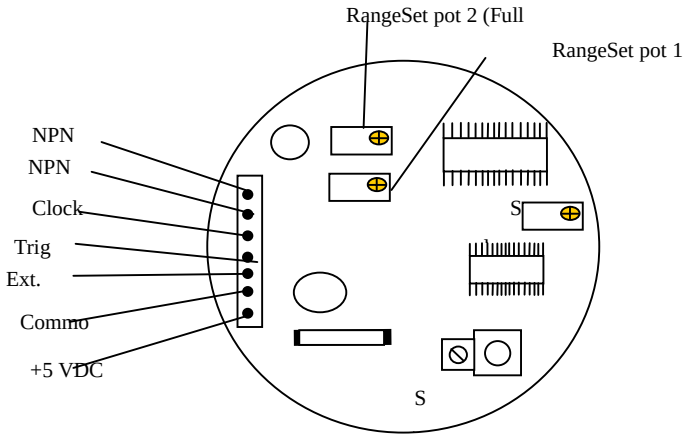


Figure 23 The electronic circuit of the ultrasonic sensor

4.4.2 Interface

Figure 24 shows a flow chart of the software. This associated program is divided into five main modules. Once the program starts, it initializes the serial communication and clears some registers and variables. Then the program polls the data from the back and from the front sensor. If it happens that a sensor (either front or back one) detects an obstacle, it will send signals through the top master controller and changes the state of the robot through the mid master controller. Table 3 lays out every combination of the signal that can be obtained from the sensor and actions taken by the program. The reader can refer to appendix B for the complete program used for the sensor.

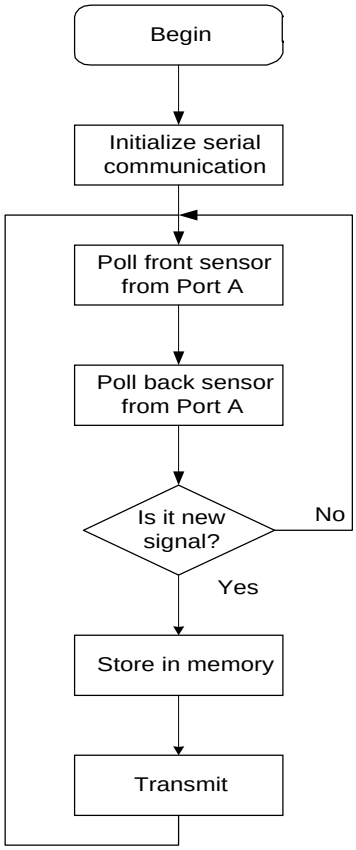


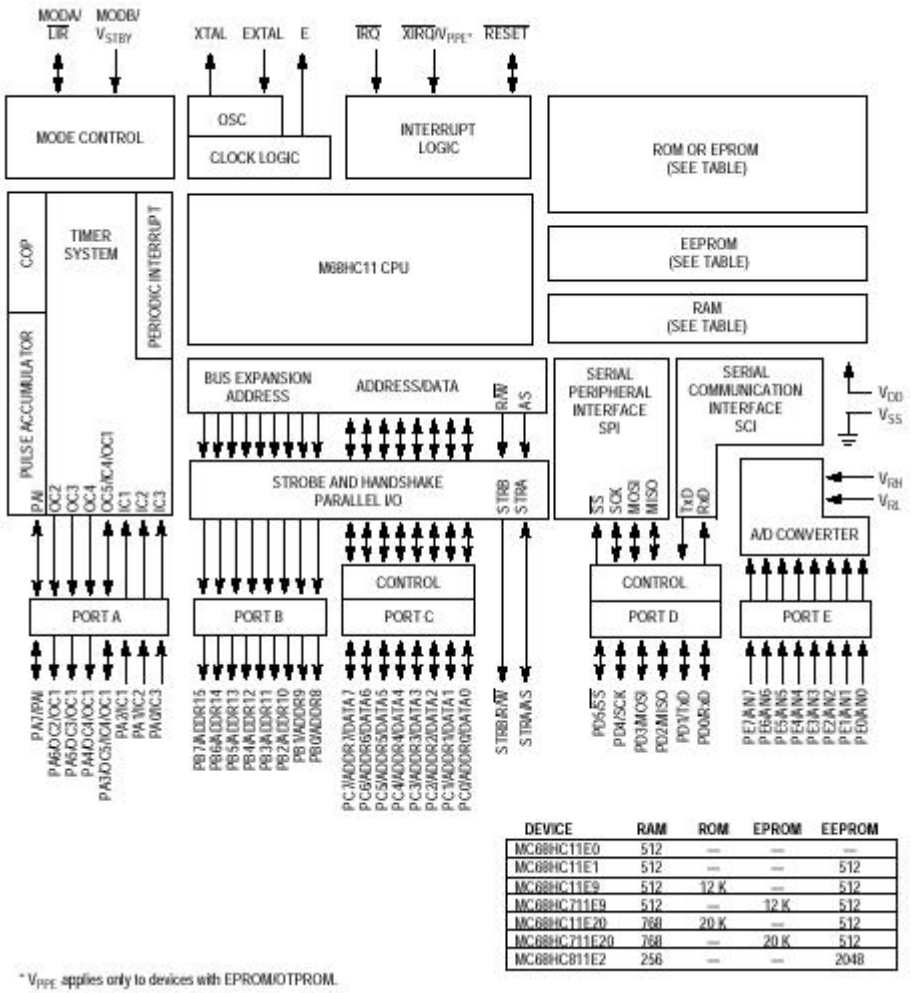
Figure 24 State chart of the interfacing program

.Front Sensor (A0)	Back Sensor (A1)	Top Master Sent	Mid Master Interpretation
0	0	Sends nothing	No new action
0	1	Sends $31	Go Forward
1	0	Sends $32	Go Backward
1	1	Sends $33	Stop

Table 3: Possible states of the robot

Chapter: 5 Hardware Design

This chapter describes the overall hardware design of the robot. For this robot, the Motorola MC68HC811E2 micro-controller was used. The chip is based on the popular Motorola 68HC11, an eight bit micro control with built in peripherals (see Figure 25). This particular modal has 2-Kilobytes of EEPROM, which is essential for rapid prototyping and software development.



DEVICE	RAM	ROM	EPROM	EEPROM
MC68HC11E0	512	—	—	—
MC68HC11E1	512	—	—	512
MC68HC11E9	512	12 K	—	512
MC68HC711E9	512	—	12 K	512
MC68HC11E20	768	20 K	—	512
MC68HC711E20	768	—	20 K	512
MC68HC811E2	256	—	—	2048

* V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 25 Building Block of the micro-controller 68HC11

This micro-controller has been chosen because of its availability, ease of use (The designers are familiar with its operation) and its use of EEPROM as explained above.

5.1 Main Board Design

The initial design of the robot was for four legs; originally, the main board hardware was designed around these considerations. After many design revisions to the robot hardware, the final body design used only two legs.

The following hardware design still uses the same design as the six-legged machine only now with two leg processors. Initially the robot was to have six micro-controllers one for the middle layer, one for the top layer control and four for the bottom layer (One for each of the leg subsystems). In the final design, there are only four micro-controllers, one for the top layer, one for the middle layer and two for the bottom layer.

The main board consists of four micro-controllers, two signal multiplexers, power supply and all necessary interface electronics for the sensors and servos (see Figure 26).

The multiplexers allow the middle master to direct communications between each of the other micro-controllers. One multiplexer is used to allow data to be transmitted from the middle master to a specified micro-controller and the other to receive data from the other micro-controllers. The multiplexers are controlled by the middle master. The middle master sets the appropriate control line of the multiplexers to set up a communications channel to the selected micro-controller. When a multiplexer's control lines are set to a specific channel, the middle master has a fixed communications link with the specified micro-controller. All other micro-controller communications blocked until the data transfer is complete. The top layer micro-controller is interfaced to two ultrasound sensors that detect obstacles in front or back of the robot. Each leg is interfaced to four servos and two bump sensors (as explained in chapter 2). A schematic for the main board can be found in appendix C.

When wire wrapping the main board, it was discovered that the pins of the micro-controllers were too short to be wrapped. In response to this, a special daughter riser board (DRG) had to be manufactured to correct this problem (see figure 27).

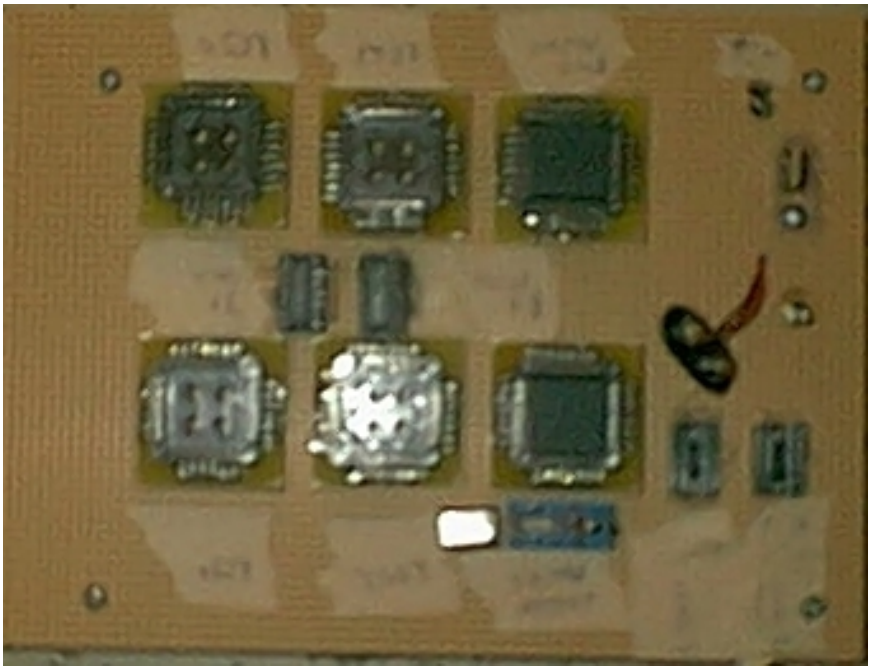


Figure 26 Main board

As a result, the micro-controllers plug directly into the DRG and the DRG is wired to the board.

Figure 27 DRG

5.2 Inter-processor Communications

To allow the micro-controllers to communicate with each other, a serial communication strategy was employed. The use of serial communications on the 68HC11 allows the micro-

controller to continue data processing while new data is being transmitted or received. Continuous data processing can be achieved because the serial port takes care of all transmissions in hardware separately. Thus allowing data to be read or stored in parallel from the serial port register when needed.

The serial ports of each of the micro-controllers are connected to a channel on the multiplexers, with the middle master connected to the common channel of the multiplexers. Essentially the transmit pin of the middle master is connected through the transmit multiplexer to the receive pins of each micro-controller. The receive pin is connected through the receive multiplexer to the transmit pins of each micro-controller.

An interrupt strategy was implemented to allow the other micro-controllers to initiate communications with the middle master. For the top master to send data to the middle master, PB0 of the top master is connected to the XIRQ pin of the middle master. When the top master asserts PB0 the middle master will set the multiplexers to the top masters channel and data transfer will occur. To allow a leg to initiate communication, PB0 of every leg micro-controller is tied to the IRQ pin of the middle master and to separate pins on PORTC.

To distinguish which leg wants to communicate with the middle master, the middle master reads its PORTC after the IRQ pin is asserted. This indicates which micro-controller (Leg) needs communication.

This overall hardware design achieves a real time hardware response and a simple communications protocol.

5.3 Example of hardware control

This example will illustrate how the complete hardware functions. For example if the robot encounters an object the top master will assert its PB0 pin which in turn causes the middle master's XIRQ pin to be low. The middle master answers this request by setting the control pins of the multiplexers to the appropriate setting. Then data transfer occurs from the transmit pin of the top master to the receive pin of the middle master. Next the middle master sets the multiplexers' control pins to the specified leg and data is transferred from the transmit pin of the middle master to the receive pin of the leg.

5.4 Power supply design

All of the hardware of the robot requires a five-volt DC power supply except for the ultrasound sensor, which requires at least nine volts. Two separate power supplies are used for the robot, one for the main board hardware and the second for the servos and ultrasound sensors. Shown in figure 28 is the power supply circuit. The first supply uses a nine-volt battery for its input and the second a 9.6-volt NICAD battery pack.

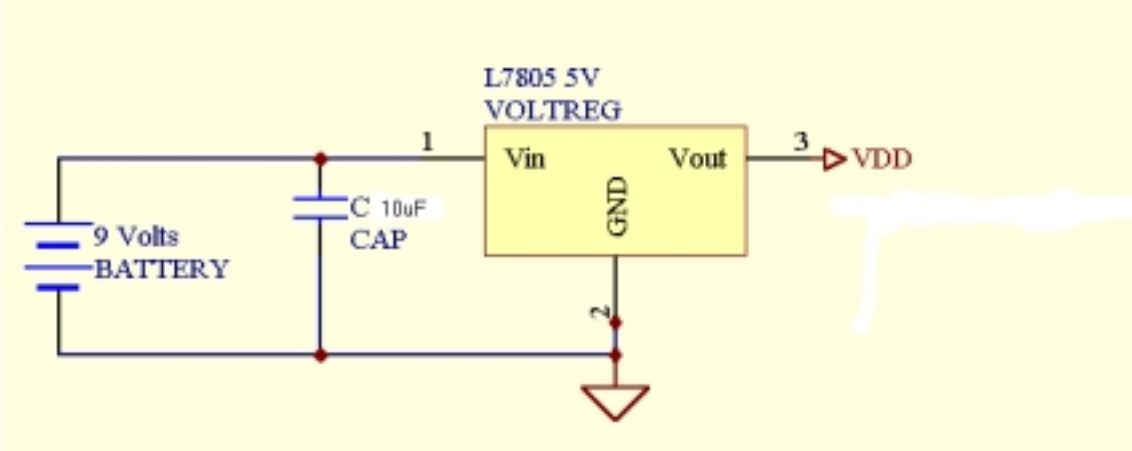


Figure 28 Ultrasound Sensors Power Supply Circuit

Chapter: 6 Software and Communication Design

6.1 Communications

A system of communication had to be established in order for the master controller, sensors and legs to communicate between each other. An easy way to handle the communications between the micro-controllers of the robot is to use the serial-port interrupt routine, which will cause an interrupt every time a character is received. The interrupt routine will then check the receive buffer, grab the data and put it to memory location $0000 automatically.

The serial communication routine that was used is as follows:

```
INITSCI  SEI
         LDAA  #$30        ; BAUD 9600
         STAA  BAUD
         LDAA  #$00        ; M=0, 8 DATA
         STAA  SCCR1       ; 1 STOP
         LDAA  #$2C
         STAA  SCCR2
         CLI
         RTS
```

First of all, the interrupt mask is being set to avoid the initialization routine being interrupted. The baud rate is then set to 9600. The character format is set as start bit, 8 data bits and 1 stop bit.

Next, the receiver and transmitter is enabled and the receiver is set to cause an interrupt should any data comes to it.

The code that was used to receive data through the RX line is as follows:

```
                ORG         $FFD6
```

```
                FDB         INSCI

RDRF            EQU         $20

INCHAR          LDAA        SCSR        ; STATUS
                BITA        RDRF        ; RDRF
                BEQ         INCHAR

                LDAA        SCDR
                STAA        $0000
                RTI
```

The interrupt vector in $FFD6 is set to recognize the starting address of the interrupt routine. The Serial Communication Status Register is then read to clear the Transmit Data Register Empty Flag, clear the Transmit Complete Flag, clear the Receive Data Register Full Flag and to clear the Idle Line Detected Flag. It will also check to see if the data has arrived. If the data is still being received, it shall wait until the data receive process is completed. The data is then read from the Serial Communication Data Register and put into memory location $0000. To transmit data to the receiving end, data can be written to the Serial Communication Data Register.

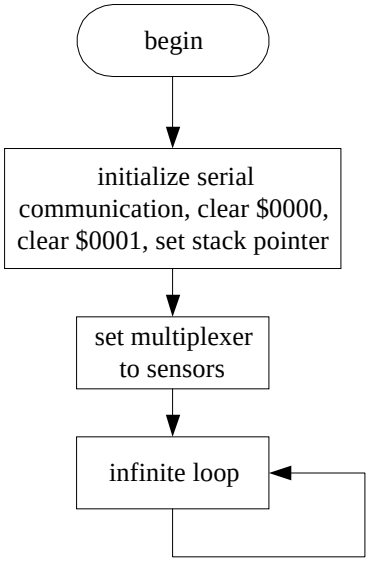
6.2 Master Controller

The master controller will be controlling the activity of the robot. It will begin by initializing the serial communication in which it will receive information from the sensors, clear memory location $0000 and $0001 and initialize the stack pointer. It will then set up the multiplexer to enable access to the sensors. After this, it will wait to be interrupted by the sensors.

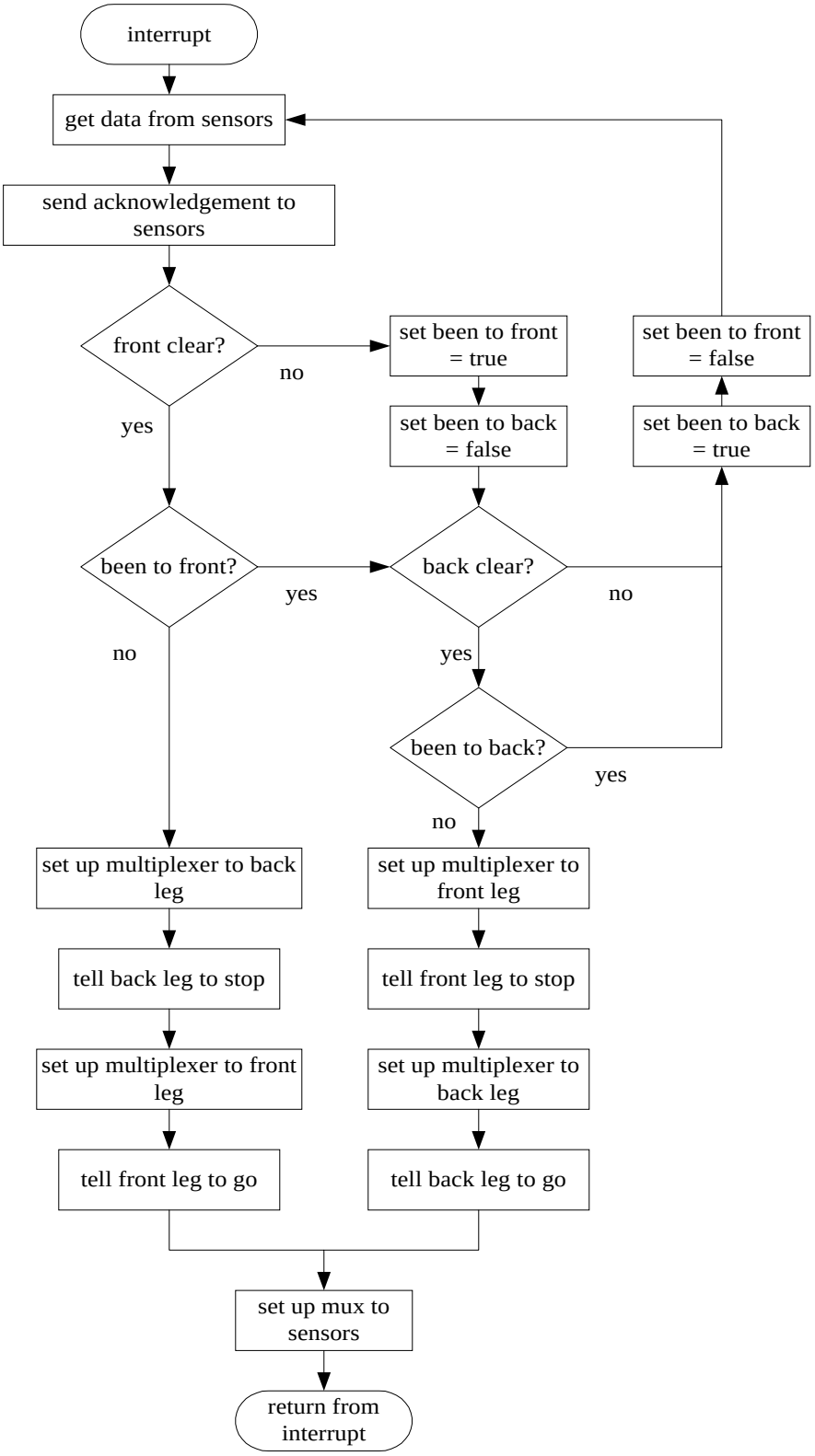
Upon interruption by the sensors, it will then proceed to its interrupt routine. It will receive data from the sensors via the RX line and acknowledge this by sending back to the sensors. It will then decide on what to do depending on the data from the sensors. Before it can issue commands to the legs, it will have to set up the multiplexers to gain access to them. It will also

keep track of its movements by checking memory location $0001 in which it will only change its movement from forward to backward and vice versa if and only if it reaches an obstacle.

6.3 Flowchart of Master Controller



6.4 Flowchart of Interrupt Routine of Master Controller



Chapter: 7 Reliability

System Reliability

System reliability calculations are a vital part of any complex system design. However, due to the nature of the project, accurate numbers are difficult to predict when the design is in the prototype stage. An overview of system reliability and the approach taken in this report can be found in (Peters & Pedrycz, 2000, "Software Engineering An Engineering Approach", Ch 15)

7.1 Main board reliability

The Main board, system reliability diagram is shown figure 29. Reliability data for the electronic components are not readably available, in data books or on the WWW. Therefore, a gross estimate was made in order to get some preliminary results. Each micro-controller and MUX was initially assigned a reliability of 0.98, and would be adjusted as data samples are received. The reliability for the construction block in the system diagram was found by counting the number of errors for every ten wires wrapped in the main board prototype. This was discovered to be eighty percent reliable.

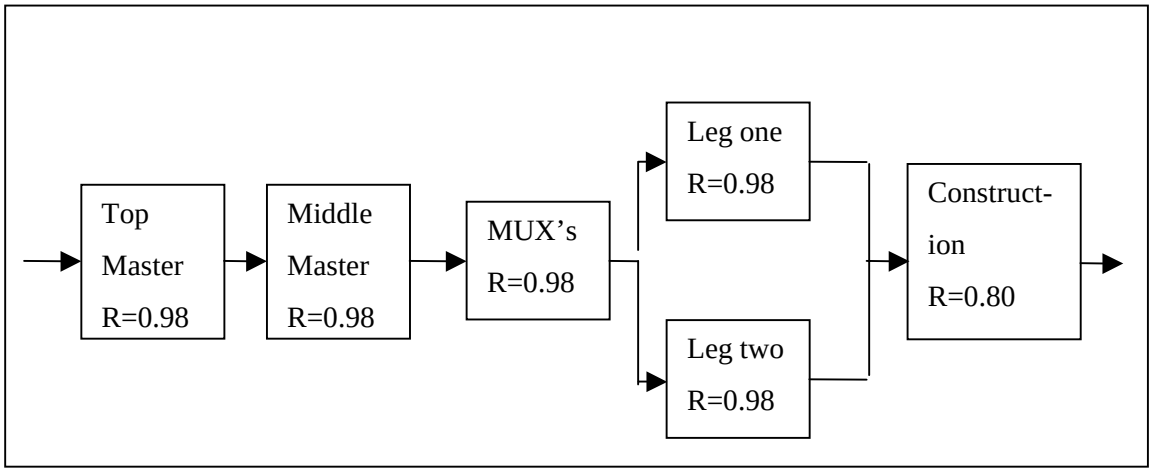


Figure 29 Main board reliability

The total reliability for the main board system is:

RT = 0.98*0.98*0.98*(0.98+0.98-0.98*0.98)*0.80 = 0.75

7.2 Mechanical Part Reliability

The mechanical design reliability can be divided into four sections; they are roller reliability, leg reliability, servo reliability and gripper reliability. The mechanical design reliability must be divided since there are many parts in the mechanical design and the total reliability for the mechanical design will be the product of these individual parts.

The roller reliability is estimated to be R = 0.985, meaning that it is very reliable. The reason for the high reliability is because the roller is made mainly made of highly durable plastic. This type of plastic is resistant to water and low temperatures. However, the pins inside the roller may have a chance to snap, therefore not making the roller 100% reliable.

Each leg is highly reliable since each is made with highly durable plastic which is resistant to water and low temperatures. Each leg is secured with durable plastic pins and hard plastic tubes that help strengthen the overall structure of the leg. However, the leg joints might become lose over time, which makes the leg not 100% reliable. The reliability of each leg is estimated to be R = 0.99, making the total leg reliability Rleg = 0.99*0.99 = 0.9801.

There are eight servomotors in this system. The total servomotor reliability will be the product of R's of these individual servos. Each servo motor is not 100% reliable, when there is an over exceeded force applied to the servo, it could have a chance to snap. This happened during the project as one of the servos broke when it could not handle to heavy weight of the leg. This problem was solved when the leg length was shortened. Each servo has an estimated reliability of R = 0.98, hence dropping down the total servo reliability to

Rservo = 0.98 ^ 8 = 0.85.

The gripper is made with the same plastic material as the leg, therefore making the gripper hard and durable. The gripper reliability is estimated to be the same as the leg reliability, so Rgripper = 0.9801.

The mechanical design reliability is Rroller * Rleg * Rservo * Rgripper = 0.804.

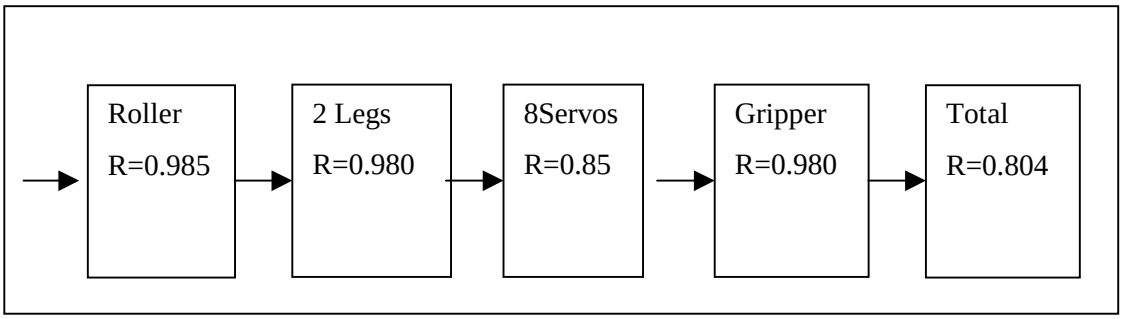


Figure 30 Reliability of mechanical parts

7.3 Software Reliability

There are many issues to be thought about when discussing the software reliability of the robot. Some of the issues that would be worth thinking about are the interaction of the robot when loading the software into the robot. The accuracy of the pulses generated by the robot, the way the robot responds to the code, execution time and accuracy of communication.

It is noted that the results of the software reliability would not project an accurate result, as the robot should be subjected to more severe testing. The figures calculated below are rough estimates of the reliability of the code that controls the legs and the main body.

7.3.1 Software Reliability of Program which Controls the Servos

P(generating accurate pulses to legs) = 0.98

It is rather tricky to coordinate the pulses of the legs. If the leg is interrupted too many times, it will tend to generate an inaccurate pulse to the servos, which may cause the servos to align in the wrong positions.

P(receiving accurate data from main body)=0.97

Accurate information must be obtained from the main body in order to perform specific tasks. If the information obtained is wrong, it would be rather disastrous as it might cause the robot to collide into obstacles instead of avoiding them. The robot is only able to obtain information from the main body after it has finished all of its routines. For example, if it was told to move forward, it will have to go through all five positions first before deciding what to do next.

P(locating the line) = 0.99

It depends on the algorithm to locate the position of the line. Once the location of the line is estimated, the robot leg is then lowered till it touches the bump sensors.

P(stopping in time) = 0.98

If an obstacle is detected in front, the main body will send a message to the robot leg, telling it to stop and. It should stop in time in order to avoid obstacles.

Thus, it can be calculated from the figures above that the overall reliability of the robot leg is: 0.98*0.97*0.99*0.98= 0.92227212

7.3.2 Software Reliability of Program which Controls the Main Body

P(receiving accurate data from sensors) = 0.99

Communication is very important in the sense that the main body should obtain accurate information from the sensors. If the information received is not accurate, the main body will not be able to make the right decision, especially in avoiding obstacles.

P(sending signals to leg)=0.98

Reliability in sending signals to the leg is very important because the robot leg may respond incorrectly if the wrong signals are sent.

P(deciding correct state of robot) = 0.99

After receiving data from the sensors, the main body must then decide what the robot should do. Making the right decision is tricky because the main body must accommodate and solve all the different problems that the robot might encounter.

Thus, it can be calculated from the figures above that the reliability of the program which controls the main body is: R=0.99*0.98*0.99 = 0.960498

7.4 Ultrasonic Sensor Reliability

Discussion on the reliability of the ultrasonic sensor system is divided into two parts: the reliability of the sensor functionality (hardware reliability) and the reliability of the software written to interface the sensor with micro-controller (software reliability).

7.4.1 Hardware Reliability

Reliability of a sensor is its ability to perform a specific function based on some conditions for a stated period. Statistically, it can be viewed as the probability that the sensor will function without failure over certain time or a number of uses. Reliability of the sensor depends on its operating environment. There are some environmental factors that affect the performance of the ultrasonic sensor. Some of these factors are temperature, humidity, and pressure. A common approach that manufacturers use to calculate the reliability of sensors is to operate sensors within the range of the highest and lowest limit of declared factors. For example, a sensor can be specified to operate between the range of -40 ^{o} to 85^{o} C, and relative humidity (RH) of 50% to 99% RH, and supply voltage of +5V to +15V.

A number of tests can be done with respect to different combinations of temperature, humidity, and power supply of the defined ranges to figure out the number of failure tests within a certain time. After a set of tests are done, the failure intensity of the sensor is calculated which in turn will specify how reliable the sensor is (See appendix **A**).

The manufacturer of the purchased ultrasonic sensor did not provide us with any precise value related to reliability of the sensor. However, in the information sheet that was delivered with the sensor, it briefly mentions that in indoor or protected outdoor places, the sensor is highly insensitive to temperature, humidity, and pressure changes. Therefore, it can be assumed that the sensor is more than 99% reliable.

7.4.2 Software Reliability

In general, the reliability of software can be measured based on two factors: the correctness of the software and the ability of the software to recover against unexpected transaction and/or system failure. Correct software can be defined as a program that is bug free and produces the correct result. It is usually a simple task to check the correctness of the short programs. It is especially true for the cases when the input domain is bounded and short, Input instances are fed to the software and if all results are correct, the software is correct. On the other hand, to judge the degree of the reliability of the software based on the software ability to recover from a system or a transaction failure is not an easy issue. This requires developing recovery routines that can keep track of the changes made to the state of the system as the system is running. After a failure, the recovery routines come into action to bring the sensor system to the most stable state that it had before the failure occurred.

The software developed to interface the sensor with the micro-controller does not include a recovery algorithm; but it can definitely be enhanced for future work. However, the software is correct and 100% bug free. The input domain includes four sets of inputs (11, 01, 10, and 00 as explained I the implementation section). All the input instances have been simulated and the results were all satisfactory.

7.5 Robot Reliability

The overall reliability of the robot system is shown in figure 31.

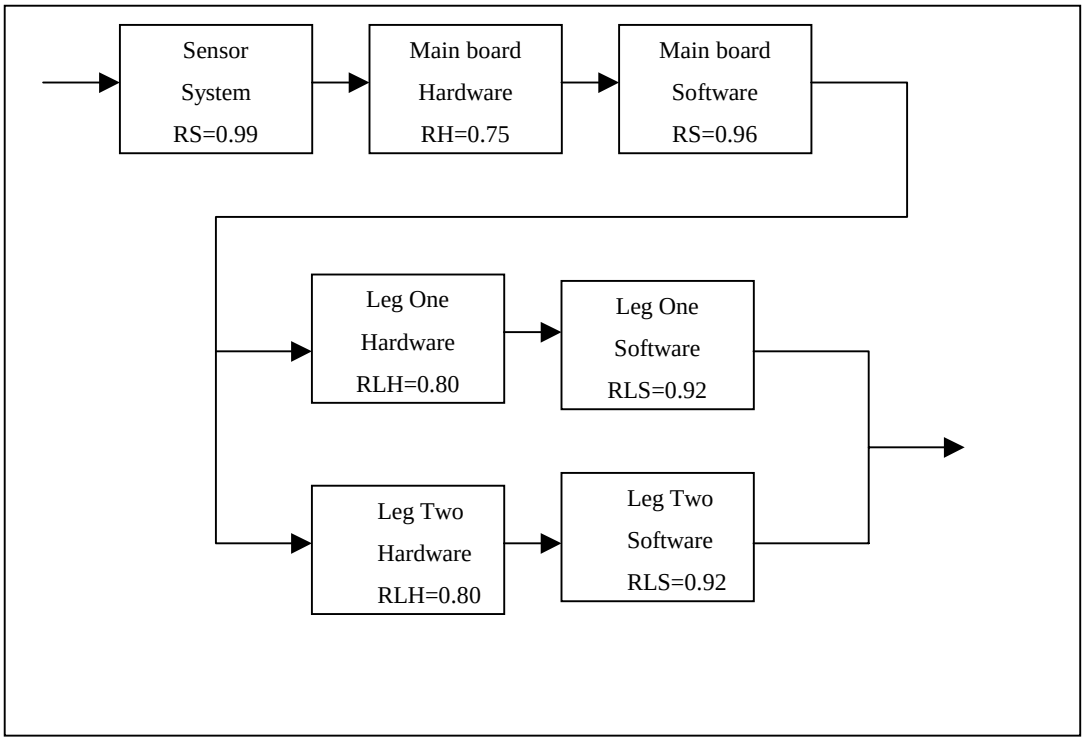


Figure 31 Overall system reliability

The total reliability for the robot system is:

RST = 0.99*0.75*0.96*(0.80*0.92+0.80*0.92 –(0.80*0.92*0.80*0.92)) = 0.66

Chapter: 8 Results and Discussion

This robot was designed to crawl along a de-energized power line. Theoretically, the robot was to move back and forth along the line avoiding any obstacle encountered. This proved to be a difficult task.

Individually every sub-system of the robot functioned as designed. The main board micro-controllers are able to communicate to one and other at a baud rate of 9600 BPS. Any bit-rate below this will result in communication error.

The leg system provides an optimal way for the robot to traverse the power line. Each leg has three degrees of freedom, allowing the leg to extend itself efficiently. Every joint of the leg worked as anticipated, the first joint was able lift the leg up and the second joint was able to extend the gripper forward. The wrist joint correctly aligned the gripper perpendicularly on top of the power line. The gripper was very strong when tested. Every part of the leg worked synchronously with each other and it was able to pull the body forward. Although the pull up movement was very slow due to the heavy weight of the leg, but it was adequate. When the roller system was tested along with the legs, it was noted that the roller did not provide a smooth sliding movement to the body. It is recommended that the roller be redesigned for the next generation of UMBOT.

At the final stage of the project, all individual components were combined and tested. All programs were burnt into memory and the microprocessors were installed in their respective daughter boards. The roller and leg systems were mounted on top of the main body board and all the necessary control lines were installed. Finally, the power supply was attached to the bottom of the main board and the robot was ready to run. When the power switch was turned on, smoke was coming from the voltage regulator and the voltage regulator got very hot. As this was happening, two servos in one of the legs turned very fast and snapped, causing these two joints to break. It is anticipated that the servos were drawing too much current from the power supply causing the voltage regulator circuit to burn. Therefore, the robot was not functional. One solution to solve this problem is to properly interface each servo so that each servo can not draw

exceeded amount of current from the power supply. If these servos are properly interfaced, they would not snap because they would draw less current making itself to spin slower.

Chapter: 9 Recommendation

Although this robot can be mobile on a power line, revisions of the current design are always needed. Technology in the twenty-first century will grow without limits; there will always be generations and generations of better design. One recommendation of the next generation could be implementing extra legs and detachable roller for obstacle avoidance, so that it can move over insulators in the real world. This new leg must be both long and strong allowing maximum support when hanging loosely across the insulators. The roller should be made with a locking system to prevent itself from detaching accidentally during obstacle avoidance.

One recommendation would be putting heat guns and wireless video cameras on the tip of the grippers, allowing the user to monitor the conditions of the power line and uses the heat gun to melt ice patches. The wireless video camera would transmit video signals to the television so that the user can monitor the conditions on the power line in a station. Then the user could signal the microprocessor to activate the heat gun to melt the ice. Once the ice is melted, the robot can then move forward to further explore the conditions on the power line.

Another recommendation would be to install incline and decline sensors on the robot. When the robot is moving on decline, the decline sensor signals to the microprocessor, and the microprocessor would idle to legs so that the robot can slide down slowly using its roller. A Breaking system could be installed in the roller to prevent the body sliding too fast and prevents robot from crashing. When the incline sensor signals to the microprocessor, the microprocessor applies 6V instead of 5V across each of the servos to increase power, so that it does not move too slowly on an incline.

Other recommendations include using stepper motors for building stronger joints, using induction power drawn from the energized line to energize the robot. Implementing an ultrasonic distance sensor, to measure how far away the robot is to obstacles.

Chapter: 10 Conclusion

Building a robot that can maneuver on power transmission line is a very difficult task. The better the leg design the better the mobility of the robot. Building a good leg design is very important to the functionality of the power line robot. Programming the leg movements must be precise, where each small degree of error may cause the robot to fall off to the ground. The movements of the robot must also be small and precise because any big movements may cause a powerful electrical shock. When this happens it is disastrous. It could cost thousands of dollars in damages.

This project focuses mainly on the development and design of a mobile robot on a de-energized high-voltage transmission line. Although there were many assumptions made before the project began, this robot acts as the basic building block of future generations of UMBOTS to accomplish this task. The goals have been met in this thesis. A complete new leg design and hardware layered structure was developed. The legs are capable of pulling the body forward and backward and inter-processor communication was successfully completed.

There were many problems encountered during this thesis. There were a few problems designing a six-legged system in the beginning. But this problem was solved using a roller design with two gripper legs. There was also a problem getting the ultra-sound sensors working because the team had no experience with ultra-sound sensors before. Overall, problems were mainly associated with the leg design, there were three complete revisions of the leg design throughout the project.

Bibliography

1. J.F.Peters, and W.Pedrycz, 1999. Software Engineering: An engineering approach. R.R. Donnelly & Sons Company
2. Jacob Fraden, 1996 Handbook of Modern Sensors. Physics Designs and Applications
3. Ken Ferens, 1999. 24.361 Microprocessor Systems
4. Brooks, R. A., 1986 A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation RA-2(1):14-23, 1986
5. Carrano, Helman, Veroff, 1998. Data Abstraction and Problem Solving with C++
6. Jonathan W. Valvano, 2000. Embedded Microcomputer Systems Real Time Interfacing
7. Motorola M68HC11, 1991. Reference Manual
8. Motorola M68HC11, 1991. Technical Data
9. Gordon Mccomb, 1987. Robot Builders Bonanza 99 Inexpensive Robotics Projects. McGraw-Hill
10. Leo L. Beranek, 1954. Acoustics, McGraw-Hill
11. Frank Massa, 1942. Acoustics Design Charts, The Blakiston Company

Appendix A

Specification for sensor

The ultrasonic sensor that is used in this project has following features:

- Range: Detects the range of 6 inch to 10 feet at the stable temperature
- Beam Angle: 15 degree nominal (single beam coin shape pattern)
- Frequency: 10 Hz frequency stable that can be externally triggered for more frequency
- Outputs: Two adjustable NPN open collector outputs, transistors are continuously energized during detection time
- Environment: environment required is indoors or protected outdoor environment. It is highly insensitive to temperature, humidity and pressure change.
- Size: The size is about 1.7 inches overall diameter and 0.97 inch depth.
- Weight: It is approximately 0.6 ounces.
- Power Requirement: It is 8.0 – 16.0 VDC and maximum 30 mA current capacity

Appendix B

Listing of programs

Source Code of the Program which Controls the Legs of the Robot

```
****************************
* VARIABLES
****************************

START   EQU     $F800
RESET   EQU     $FFFE   ; RESET VECTOR
BAUD    EQU     $102B   ; SERIAL BAUD RATE CONTROL
SCCR1   EQU     $102C   ; SERIAL CONTROL REGISTER 1
SCCR2   EQU     $102D   ; SERIAL CONTROL REGISTER 2
SCSR    EQU     $102E   ; SERIAL STATUS REGISTER
SCDR    EQU     $102F   ; SERIAL DATA REGISTER
PORTB   EQU     $1004   ; PORT B
PORTC   EQU     $1003   ; PORT C
DDRC    EQU     $1007   ; PORT C DIRECTION REGISTER

****************************
* RESET VECTOR
****************************

        ORG     RESET
        FDB     START

************************************************
* INIT INTERRUPT TO RECEIVE DATA VIA SERIAL COMM
***************************

        ORG     $FFD6
```

```
        FDB     INSCI

  ***************************
  * MAIN PROGRAM
  ***************************

  *********************
  * INITIALIZE
  *********************

          ORG     START
          LDS     #$00FF
          LDAA    #$00
          STAA    DDRC    ; set all port pins in port C to be
inputs
          STAA    $0000
          JSR     INITSCI  ; INITIALIZE SERIAL COMMUNICATIONS
          JSR     INITSERVO
          JSR     STARTPOS ; STARTING POSITION

  ********************************
  * POLLING
  * 01 WOULD MEAN FORWARD
  * 00 WOULD MEAN TO STOP VIA ALPHA5
  ********************************

  POLL    LDAA    $0000
          CMPA     #$01
          BEQ     ALPHA1 ; FORWARD
          CMPA     #$00
          BNE POLL ; FORCED TO USE JUMP INSTRUCTION
          JMP ALPHA5 ; STOPPING POSTION


  ******************************
  * INITSERVO
```

```
   ******************************

   INITSERVO LDAA    #$00
             STAA    $1004


   ************************************************************
*
   * FORWARD
   * LEG HAS 5 DIFFERENT POSITIONS
   * ALPHA 1 - LEG PREPARES TO GRAB LINE
   * ALPHA 2 - LEG REACHES OUT FOR LINE
   * ALPHA 3 - GRIPPER CLOSES
   * ALPHA 4 - LEG PULLS FORWARD
   * ALPHA 5 - GRIPPER OPENS (STARTING POSITION)
   *  EACH  POSITION  HAS  1  SECOND  ALLOCATION  FOR  POSITIONING  OF
SERVOS
   ************************************************************

            LDAB  #$30
   ALPHA1   LDX   #123
            LDAA  #$0F
            STAA  $1004
            JSR   DELAY
            LDX   #27
            LDAA  #$03
            STAA  $1004
            JSR   DELAY
            LDX   #10
            LDAA  #$01
            STAA  $1004
            JSR   DELAY
            LDX   #2500
            LDAA  #$00
            STAA  $1004
            JSR   DELAY
```

```
        DECB
        CMPB  #$00
        BNE   ALPHA1


ALPHA2  LDX   #123
        LDAA  #$0F
        STAA  $1004
        JSR   DELAY
        LDX   #27
        LDAA  #$02
        STAA  $1004
        JSR   DELAY
        LDX     #2500
        LDAA    #$00
        STAA    $1004
        JSR     DELAY
        LDAB    PORTC ; CHECK FOR SENSOR
        BITB    #$01 ; CHECK BIT 0
        BNE     ALPHA3 ; IF DETECTED, CLAMP LINE
        BRA     ALPHA2


ALPHA3  LDX     #123
        LDAA    #$0F
        STAA    $1004
        JSR     DELAY
        LDX   #27
        LDAA  #$0A
        STAA  $1004
        JSR   DELAY
        LDX   #33
        LDAA  #$08
        STAA  $1004
        JSR   DELAY
        LDX     #2500
```

```
            LDAA    #$00
            STAA    $1004
            JSR     DELAY
            LDAB    PORTC ; CHECK FOR SENSOR
            BITB    #$02 ; CHECK BIT 1
            BNE      ALPHA4 ; IF CLAMPED WIRE, STOP CLAMPING, PULL
FORWARD
            BRA     ALPHA3

            LDAB    #$30
   ALPHA4   LDX     #123
            LDAA    #$0D
            STAA    $1004
            JSR     DELAY
            LDX   #37
            LDAA  #$09
            STAA  $1004
            JSR   DELAY
            LDX #23
            LDAA    #$08
            STAA    $1004
```

Source Code of the Program, which sends sensor signal to Master Controller

```
Bud             equ     $ 102b
Sccr1           equ     $ 102c
Sccr2           equ     $ 102d
Scsr            equ     $ 102e
Scdr            equ     $ 102f
portA           equ     $ 1000
OldA0           equ     $ 0000
OldA1           equ     $ 0001
signal          equ     $ 0002
;*****************************************************************
                org             $FFFE
                fdb             start
```

```
start           org     $F800
                lds     #$00FF ;stack area
                jsr     SciInit
                ldaa    #$00
                Staa    OldA0  ;clear oldA0
                ldaa    #$00
                Staa    OldA1  ;clear oldA1
polling         Ldaa    portA   ;start to pull signal from sensors port A (A0,A1)
                Anda    #$01    ;extract  A0 from port A
                Ldab    portA   ;start to pull signal from sensors port A (A0,A1)
                Andb    #$02    ;extract  A1 from port A
                cmpA    OldA0  ;compare new signal with oldA0
                Bne     CheckA0With0
                cmpb    OldA1
                Bne     CheckA0With0
                Bra     polling
;*******************************************************************************
CheckA0With0    cmpA    #$00    ; Check if new value from A0=0?
                Bne     CheckA1with01
                cmpb    #$00
                Bne     Sent02
                Bra     polling
;*******************************************************************************
CheckA1with01   cmpb    #$02
                Bne     Sent01
                jsr     Sent03
                Bra     polling
;*******************************************************************************
Sent01          jsr     StoreToMemory
                Ldaa    #$31
                Staa    signal
                Jsr     TransmitProcess
                Bra     polling
```

```
;*******************************************************************************
Sent02          jsr         StoreToMemory
                Ldaa        #$32
                Staa        signal
                Jsr         TransmitProcess
                Bra         polling
;*******************************************************************************
Sent03          jsr         StoreToMemory
                Ldaa        #$33
                Staa        signal
                Jsr         TransmitProcess
                Bra         polling
;*******************************************************************************
TransmitProcess bra         Transmit
Transmit        ldab        Scsr            ;Check Status
                Bitb        #$80            ;check for the tdre
                Beq         Transmit
                Ldaa        signal
                Staa        Scdr
Aknowledge      Ldaa        Scsr
                Bita        #$20    ;check for the tdre
                Beq         Aknowledge
                Ldaa        Scdr
                cmpA        signal
                bne         Transmit
                rts
;*******************************************************************************
StoreToMemory   staa        OldA0
                stab        OldA1
                rts
;*******************************************************************************
SciInit         ldaa        #$30    ;9600 baud rate
                Staa        Bud
```

```
        Ldaa    #$00    ;check the mode
        Staa    Sccr1
        Ldaa    #$0c    ;check  tie = rie =0, te = re =1
        Staa    Sccr2
        rts
```

```
;*******************************************************************************
;** MAIN BODY CODE
;**
;** THE MAIN BODY GETS DATA FROM THE SENSORS AND THEN DECIDE TO
;** MOVE
;** FORWARD OR BACKWARD BY GIVING INSTRUCTION TO THE LEGS
;*******************************************************************************

BAUD    EQU   $102B     ; SERIAL BAUD RATE CONTROL
SCCR1   EQU   $102C     ; SERIAL CONTROL REGISTER 1
SCCR2   EQU   $102D     ; SERIAL CONTROL REGISTER 2
SCSR    EQU   $102E     ; SERIAL STATUS REGISTER
SCDR    EQU   $102F     ; SERIAL DATA REGISTER
PORTB   EQU   $1004     ; PORT B

        ORG   $FFFE
        FDB   MAIN

        ORG   $FFD6
        FDB   INCHAR

        ORG   $F800
```

```
MAIN            LDS   #$00FF
                CLRA                ; CLEAR ACCUMULATOR A
                STAA  $0000
                STAA  $0001         ; BEEN TO FRONT
                STAA  $0002         ; BEEN TO BACK
                LDAA  #$01
                STAA  PORTB         ; SET UP MUX, ENABLE ACCESS TO/FROM
SENSORS
                JSR   INITSCI       ; SETS UP SERIAL COMMUNICATION

AGAIN           BRA   AGAIN

*************INITSCI*********************
* INITALIZE SCI
* THIS RITUAL IS EXECUTED ONCE  ONLY RECEIVER INTERRUPTS
* INPUTS: NONE    OUTPUTS: NONE
* REGISTERS MODIFIED: A,CCR

INITSCI   SEI
                LDAA  #$30          ; BAUD 9600
                STAA  BAUD
                LDAA  #$00          ; M=0 8 DATA
                STAA  SCCR1         ; 1 STOP
                LDAA  #$2C
                STAA  SCCR2
                CLI
                RTS

*********************************
** POLLING ROUTINE
*********************************

RDRF            EQU   $20
```

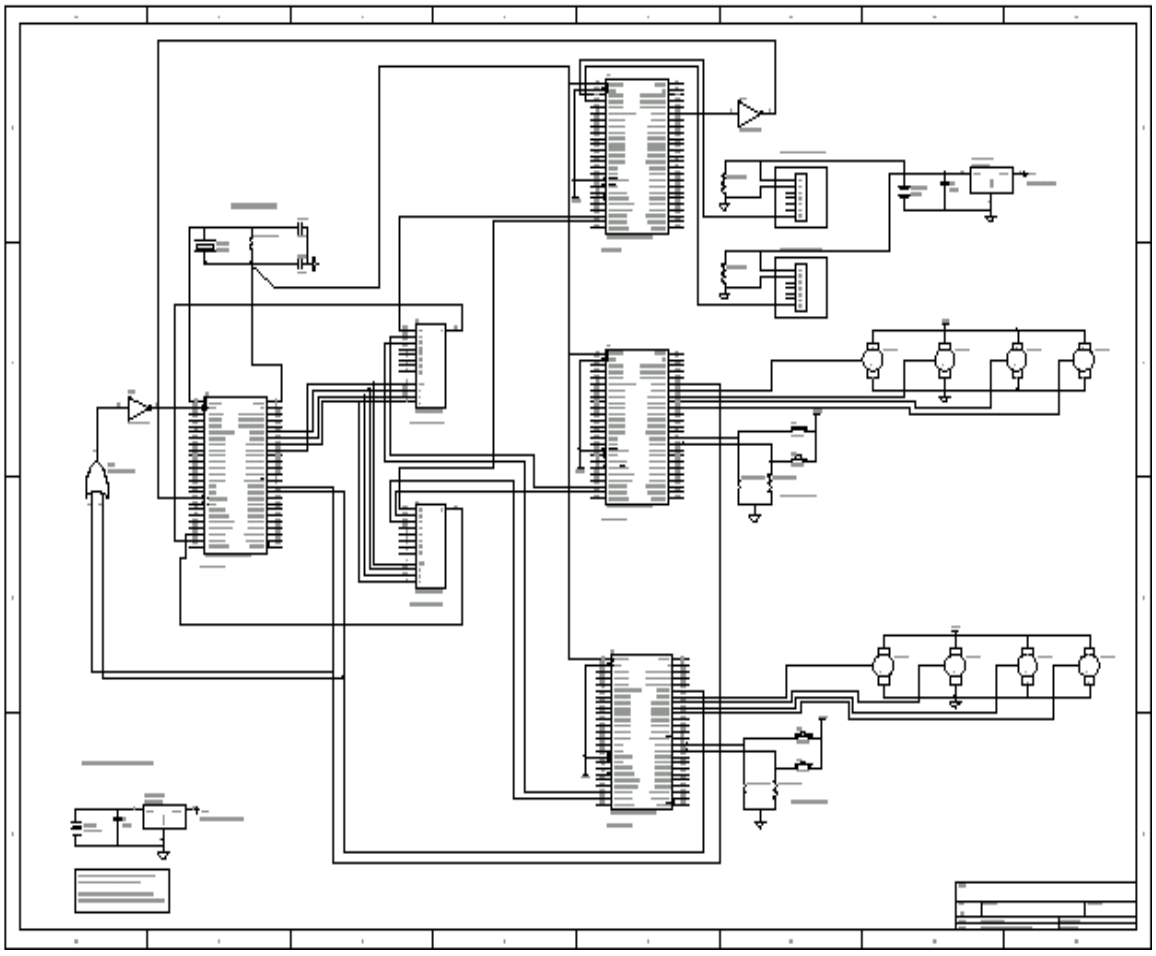
```
INCHAR  LDAA  SCSR          ; STATUS REGISTER
        BITA  RDRF          ; RDRF
        BEQ   INCHAR

POLL            LDAA  SCDR          ; SCI DATA
        CMPA  #$31
        BEQ   FORWARD1  ; FRONT CLEAR, BACK CLEAR
        CMPA  #$32
        BEQ   FORWARD2  ; FRONT CLEAR, BACK BLOCKED
        CMPA  #$33
        BEQ   BACKWARD1 ; FRONT BLOCKED, BACK CLEAR
        CMPA  #$34
        BEQ   BACKWARD2 ; FRONT BLOCKED, BACK BLOCKED
        RTI


*******************************************
** FORWARD - FRONT CLEAR, BACK CLEAR
*******************************************

FORWARD1        LDAA  $0001         ; BEEN TO FRONT? 1 FOR YES
        CMPA  $01
        BEQ   BACKCHECK ; CHECK BEEN TO BACK
GOFORD  LDAA  #$03          ; SET UP MUX TO BACK LEG
        STAA  PORTB
        LDAA  #$00
        STAA  SCDR          ; TELL BACK LEG TO STOP
        LDAA  #$02          ; SET UP MUX TO FRONT LEG
        STAA  PORTB
        LDAA  #$01
        STAA  SCDR          ; TELL FRONT LEG TO GO FORWARD
        RTI
```

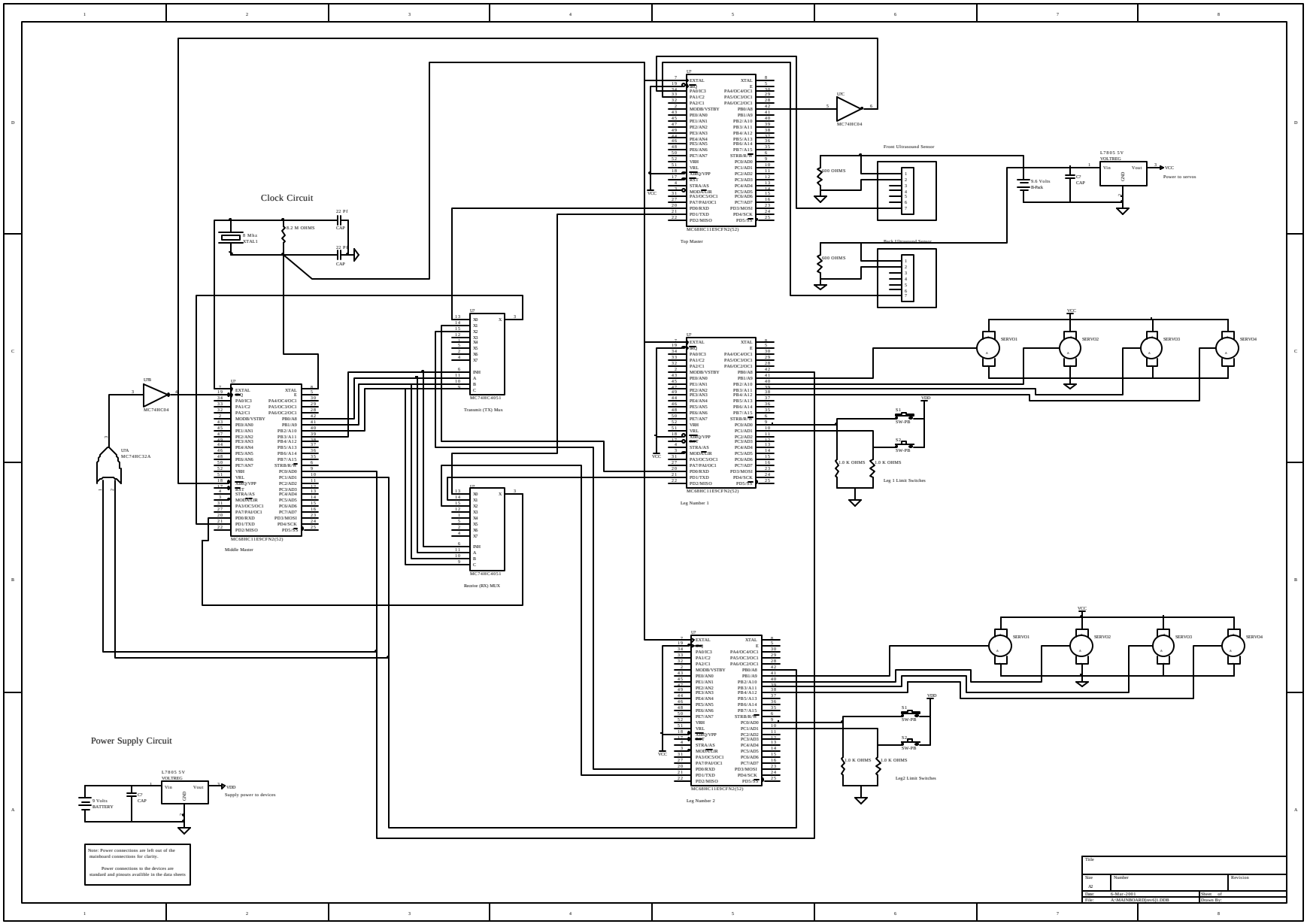
```
*****************************************
** FORWARD2 - FRONT CLEAR, BACK BLOCKED
*****************************************

FORWARD2    LDAA  $0001        ; BEEN TO FRONT? 1 FOR YES
            CMPA  $00
            BEQ   GOFORD       ; GO FORWARD
SET1        LDAA  #$01
            STAA  $0002        ; SET BEEN TO BACK TRUE
            CLRA
            STAA  $0001        ; SET BEEN TO FRONT FALSE
            BRA   POLL

*****************************************
** BACKWARD1 - FRONT BLOCKED, BACK CLEAR
*****************************************

BACKWARD1   JSR   SET2         ; SET FRONT=1, BACK=0
            BRA   BACKCHECK
GOBACK      LDAA  #$02         ; SET UP MUX TO FRONT LEG
            STAA  PORTB
            LDAA  #$00         ; TELL FRONT LEG TO STOP
            STAA  SCDR
            LDAA  #$03         ; SET UP MUX TO BACK LEG
            STAA  PORTB
            LDAA  #$01         ; TELL BACK LEG TO GO FORWARD
            STAA  SCDR
            RTI

*****************************************
** BACKWARD1 - FRONT BLOCKED, BACK CLEAR
*****************************************
```

```
BACKWARD2       BRA     SET1

*****************************************
** SET BEEN TO FRONT = 1
** SET BEEN TO BACK  = 0
*****************************************

SET2            LDAA #$01           ; SET BEEN TO FRONT
                STAA $0001
                CLRA                ; SET BEEN TO BACK FALSE
                STAA $0002
                RTS


;*****************************************
;** BACKCHECK  - BEEN TO BACK?
;*****************************************
BACKCHECK       LDAA  $0002
                CMPA  #$00          ; 0 FOR NO
                BEQ   GOBACK
                BRA   SET1
```

Appendix C

Main Board Schematic




Clock Circuit
8 Mhz
XTAL1
2.2 M OHMS
22 PF
CAP
22 PF
CAP
Top Master
MC68HC11ESCFN2(52)
MC74HCB4
Front Ultrasound Sensor
Back Ultrasound Sensor
100 OHMS
100 OHMS
L7805 5V
VOLTREG
Vin
Vout
GND
9.6 Volts
B-Pack
CAP
VCC
Power to servos
SERVO1
SERVO2
SERVO3
SERVO4
S1
SW-PB
S2
SW-PB
1.0 K OHMS
1.0 K OHMS
Leg 1 Limit Switches
Leg Number 1
MC74HC4051
Transmit (TX) Mux
Receive (RX) MUX
Middle Master
MC74HC32A
Leg2 Limit Switches
Leg Number 2
Power Supply Circuit
9 Volts
BATTERY
Supply power to devices
VDD
Note: Power connections are left out of the mainboard connections for clarity.
Power connections to the devices are standard and pinouts available in the data sheets.
Title
Size
A2
Number
Revision
Date:
6-Mar-2005
Sheet
of
File:
S:\MAINBOARD\Rev5\J-DOB
Drawn By: