Six Legged Walking Robot

Supervisor: Dr. Pan Agathoklis

Group Members:
N. Kiel Boyle
Jason Wong
Belinda Zhu
# Table of Contents

1 PROJECT OVERVIEW ..................................................................................................................................................1

1.1 INTRODUCTION .........................................................................................................................................................1
1.2 WALKING ROBOT ....................................................................................................................................................1
1.3 PROJECT GOAL ............................................................................................................................................................1

2 WALKING ROBOT ..........................................................................................................................................................2

2.1 INTRODUCTION ..........................................................................................................................................................2
2.2 TYPES OF WALKING ROBOTS ................................................................................................................................2
  2.2.1 Two-legged Walker ...............................................................................................................................................2
  2.2.2 Four-legged Walker ............................................................................................................................................2
  2.2.3 Six-legged Walker ..............................................................................................................................................2
  2.2.4 Walker with Over Six Legs ................................................................................................................................2

3 MECHANICS .................................................................................................................................................................3

3.1 INTRODUCTION ..........................................................................................................................................................3
3.2 ROBOT PARAMETER SELECTION ..................................................................................................................................3
  3.2.1 Robot’s Leg Number Selection .............................................................................................................................3
  3.2.2 Robot’s Degrees of Freedom Selection ................................................................................................................3
3.3 TYPES OF ACTUATORS ...............................................................................................................................................4
  3.3.1 Pneumatics and Hydraulics ..................................................................................................................................4
  3.3.2 DC motors ............................................................................................................................................................4
  3.3.3 Stepper motors ........................................................................................................................................................4
  3.3.4 RC Servo Motors ....................................................................................................................................................4
3.4 ACTUATOR TYPE USED IN PROJECT ........................................................................................................................5
3.5 MECHANICAL DESIGN ................................................................................................................................................5
3.6 MECHANICAL TESTS, FABRICATION AND ASSEMBLY .................................................................................................7
  3.6.1 Material .................................................................................................................................................................7
  3.6.2 Prototyping ..........................................................................................................................................................7
  3.6.3 Fabrication and Assembly ................................................................................................................................8

4 ELECTRONICS .................................................................................................................................................................11

4.1 INTRODUCTION ...........................................................................................................................................................11
4.2 CONTROL ELECTRONICS ..........................................................................................................................................11
4.3 PCB FABRICATION .......................................................................................................................................................12

5 SOFTWARE .....................................................................................................................................................................13

5.1 INTRODUCTION ..........................................................................................................................................................13
5.2 METHODS OF LEG MOVEMENT ................................................................................................................................13
  5.2.1 Ripple Gait ..............................................................................................................................................................13
  5.2.2 Wave Gait ..............................................................................................................................................................14
  5.2.3 Tripod Gait .............................................................................................................................................................14
  5.2.4 Gait Comparisons ................................................................................................................................................15
5.3 IMPLEMENTATION OF GAIT CONTROLLER .............................................................................................................16
  5.3.1 PWM signal generator ........................................................................................................................................16
  5.3.2 Gait FSM ..............................................................................................................................................................16
5.4 COMMAND PARSING ...............................................................................................................................................20
5.5 OBSTACLE AVOIDANCE ............................................................................................................................................20

6 RESULTS ......................................................................................................................................................................20

7 CONCLUSIONS .............................................................................................................................................................22

8 RECOMMENDATIONS ...................................................................................................................................................23

9 REFERENCES ..................................................................................................................................................................24
List of Figures
Figure 1: RC Servo Operation..............................................................................................................5
Figure 2: Motor Labeling.....................................................................................................................6
Figure 3: The Design of the Walking Robot Created in AutoDesk Inventor.......................................7
Figure 4: Base with Hip Motors.........................................................................................................9
Figure 5: Assembled Legs..................................................................................................................9
Figure 6: Base with One Attached Leg..............................................................................................10
Figure 7: Assembled Control Electronics..........................................................................................12
Figure 8: Software Block Diagram....................................................................................................13
Figure 9: Examples of Hexapod Gaits Completing One Full Cycle..................................................15
Figure 10: FSM of Left Side for Tripod Gait......................................................................................17
Figure 11: Picture of the Completed Robot .......................................................................................20

List of Tables
Table 1: Gait FSM Transitions...........................................................................................................17
Table 2: Serial Command Summary..................................................................................................20

List of Listings
Listing 1: Variable Pulsewidth Generation Algorithm.......................................................................16
Listing 2: FSM Transition Direction Determination..........................................................................17
Listing 3: Obstacle Avoidance Algorithm..........................................................................................20
1 Project Overview

1.1 Introduction

The objective of this CENG/ELEC 499 project was to design and build a six legged robot with the ability to walk over small obstacles and offer redundancy in locomotive capability. The intent was to design and build every component of the system independently and not rely off the shelf kits.

1.2 Walking Robot

Locomotion control in robotics is commonly achieved using wheels. However, wheeled robots have deficiencies moving in complex terrain and can have a catastrophic failure if even a single motor is damaged. Legged robots have discrete contacts with the ground that are good for walking over obstacles and navigating complex terrain. Legged robots are also protected from failure due to the redundancy of multiple legs.

A two-legged robot is very unstable and requires a complicated control design to keep its balance. A four-legged robot can achieve average stability that does not always require fine balancing. However, a four-legged robot does not offer any redundancy. If one leg fails, the robot will lose the ability to walk. A robot with six or more legs offers redundancy and the ability to walk even when a leg has failed or damaged.

1.3 Project Goal

Walking robot kits are readily available from specialty stores. They are available in several different models including robots with two legs and 2 degrees of freedom all the way up to six legs with 3 degrees of freedom and beyond. The cost of these kits is reasonably priced. Unfortunately, purchasing a kit significantly reduces the amount of knowledge and experience that would be gained by designing and building a walking robot from scratch.

The goal of this project was to design and create a walking robot without the aid of a pre-made kit. The scope of the project involved the designing and building of the leg mechanisms, motor linkages, and required software to control the motors. The project was considered successful when the robot was able to walk itself in a straight line to the opposing side of a room.

To add more sophistication to the project a remote control tether was connected to the robot. The remote control allowed the walking robot to be manipulated manually by entering commands into a computer terminal window.

Finally, bump sensors were added to the robot. The addition of the above parts allowed the walker to become autonomous. The robot would walk straight until it encountered an obstacle where it would reverse and turn to avoid it.
2 Walking Robot

2.1 Introduction

There are many different types of walking robots from bipeds of different joint configurations and intelligence, to walking animal like quadrupeds, many legged insect styles and even snake emulating robots. The basic components of a robot are:
   1. Moving parts that perform an action like arms and legs.
   2. An actuator to power moving parts and sensors to detect environment.
   3. A control system that makes decisions and overlooks the overall operation.

2.2 Types of Walking Robots

Walking robots can be implemented in various fashions. Each leg can have 2 degrees of freedom or more. In addition, the number of axis is directly related to the maneuverability of a robot’s locomotion.

2.2.1 Two-legged Walker

A two-legged walker/biped is, by far, the hardest to create. Since at the point of stride only one foot will be in contact with the ground, and the centre of gravity must dynamically shift in order to keep the robot from falling over. Whilst neither as efficient nor as simple as the wheeled or a tracked vehicle, the potential for crossing more difficult terrain is paramount. An obstruction can merely be stepped over, pass between the legs. However, there is no redundancy in the legs. If one leg fails, the robot cannot walk at all.

2.2.2 Four-legged Walker

A four-legged walker/quadruped is fairly common. An example of a commercially available quadruped is the Sony Aibo. A quadruped is simpler to control then a bipedal robot and so long as only one leg is ever off the ground no sophisticated balance is required. However, a 4 legged robot does not offer any redundancy in the legs. If a leg fails the robot looses the ability to walk.

2.2.3 Six-legged Walker

The six-legged walker/hexapod is very stable. A designer can design a gait that can take three legs off the floor at any one time leaving a stable tripod. A hexapod is very good for complex terrain, and along with quadrupeds, it is the most common legged form robot.

2.2.4 Walker with Over Six Legs

This form of robot offers excellent stability and redundancy in the mechanisms. The main problem with this robot is simply the extra cost incurred from implementing the additional legs.
3  Mechanics

3.1  Introduction
The first phase of the design was selection of design requirements. Some of the key objectives of the design were:
- Low weight
- Low cost
- Low maintenance
- Short Fabrication time
- Simplicity
- Functionality.

The following sections explain the design decisions made regarding the mechanisms for the walking robot.

3.2  Robot Parameter Selection

3.2.1  Robot’s Leg Number Selection

After analyzing the device cost and the need for extra control circuits and actuators, an eight legged robot was not ideal. Then we looked into a four-legged design. Although, it was lower in cost than the eight-legged robot and it required fewer controls and actuators, it also increased the design complexity and decreased stability. A two legged walker is even more complex and unstable than a four legged walker. After the research, a six-legged/hexapod design was chosen for the project design. A six legged walker was easier to implement than a two-legged walker, it was more stable than the four legged walker and costs less than an eight legged walker.

3.2.2  Robot’s Degrees of Freedom Selection

The number of joints in a robot roughly translates to the degrees of freedom. In the design process three different possibilities were considered. A one degree of freedom leg offers very limited capabilities and produces legs which act similar to uneven wheels. Legs with one degree of freedom also prevent the robot from adjusting its step sizes to compensate for the environment rendering one degree of freedom inadequate. A leg with two degrees of freedom using indirectly driven joints in a shoulder and elbow configuration yields motion at the tip of the leg. The foot of the leg is constrained to a spherical surface. However the up-down and forward-back motion is approximately linear and provides a method to propel forward or backward while adjusting to some uneven terrain. A three degree of freedom design offers superb maneuverability allowing the robot to adjust to many situations. However, the three degree of freedom design significantly increases the cost of the robot. A two degree of freedom leg was chosen since one degree of freedom does not offer the required capabilities and three degrees of freedom is costly.
3.3 Types of Actuators

Actuators in robots are like muscles in the human body. Without the actuators, the limbs of the robot cannot move. There are many types of actuators available but only certain types suit the needs of the project. Below is a comparison of some common actuators. Following the comparison a decision was made on the method of actuation for the robot legs.

3.3.1 Pneumatics and Hydraulics

Pneumatics and hydraulics run on the basis of a pressurized fluid driving a piston or motor mechanism. The system requires a pump or compressor to provide the necessary pressure to drive the piston or motor. This equipment is large, noisy, and expensive.

3.3.2 DC motors

DC motors are commonly found in wheeled robots. DC motors rotate as long as power is applied and stops when power is removed. A continuous rotation at a constant speed can be achieved by applying a constant voltage. However, extra support circuitry including a sensor for feedback is needed for position control.

3.3.3 Stepper motors

A stepper motor is a very simple DC motor. Because it has no brushes or contacts, it operates by having its magnetic field electronically switched to rotate the armature magnet. This setup allows the motor to rotate and halt at specific angles. There are two types of stepper motors bipolar and unipolar. The bipolar stepper motor consists of two coils. The current direction is reversed in each coil to achieve four separate positions. The unipolar stepper motor consists of four coils. When each coil is energized individually and working in proper sequence, the motor shaft turns the inherent number of degrees per step. Regardless of the type of stepper motor used there is no closed loop feedback unless an external position sensor is used.

3.3.4 RC Servo Motors

An RC servo motor is a DC motor combined with position sensing parts. RC servo motors have 3 wires running out from the motor. Two lines are for power and the third line is for the control input. A pulsewidth signal applied to the input indicates to the motor the desired position. The exact relation of pulsewidth to output shaft location varies by motor model; but as a standard all RC servos moves to the center position with a 1.5 ms input pulse. Inside the RC servo motor, the components consist of a DC motor, a gear train, limits stops, a potentiometer for position feedback and some control circuitry. RC servo motors serve as an easy to implement open loop control system for robots. They remove complexity from the control system hardware and software and also reduce the total part cost and design time. The diagram shown in Figure 1 serves as reference to the control of an RC servo:
3.4 Actuator Type Used in Project

Pneumatics and hydraulics increase the cost and weight of the robot without offering any functional benefits. To reduce weight the pump or compressor can be taken off the robot and a tether can be extended to the robot to provide the pressurized fluid. Unfortunately this means the robot would have a tether and a limited walking range. Although DC motors are inexpensive they lack position control and are not directly suitable for this robotic application. Stepper motors have an open loop position control and can easily skip steps resulting in poor correlation between expected and actual position. RC servos were chosen because of their relatively low cost and their ease of control. The main drawback of an RC servo was the loss of feedback position control to the external device providing the control pulse.

3.5 Mechanical Design

A design was drafted in AutoDesk Inventor after the decision to create a six legged walking robot with two degrees of freedom per leg was made. Since the design required
six legs a total of 12 servo motors were used. Each leg was designed to use two servo motors. The first motor drives a hip joint giving front to back motion. The second motor drives a knee joint creating an up and down motion. The twelve motors were grouped into six legs labeled A through F and numbered as shown in Figure 2. The knee and hip joints are connected to the servo motors by use of linkages. The linkages are connected with a pin to a plastic disk mounted to the shaft of the motor. The rotation of the disks, spacing of the joints and the placement of the holes were all calculated to create a desired motion of the legs. This is what made inventor such a useful tool. By simply changing the design parameters in the software we were able to ensure the motion of the system worked as required. The design is shown in Figure 3.

Figure 2: Motor Labeling
3.6 Mechanical Tests, Fabrication and Assembly

3.6.1 Material

The initial choice for construction material was Sintra®, a moderately expanded, closed-cell PVC sheet. The material was chosen initially for its strength to weight properties and ease of fabrication. However, upon recommendation from people in industry the construction material was switched to aluminum. The decision would add some extra weight but would greatly increase strength of the robot.

3.6.2 Prototyping

An initial prototype leg was built to reveal design flaws and provide an opportunity to recover from mistakes made in the design. The initial construction material, Sintra, was used to manufacture the first prototype leg mechanism. The prototype design used Sintra because it was easier to work with. During this phase it was observed Sintra would not have been a good option for the final assembly. The Sintra was too soft and risked buckling of the legs with any added weight. Instead the design needed to use a stronger material like aluminum.

A single leg was constructed out of aluminum to again verify operation with the manufacturing material and processes used for the leg. This prototype also included
connecting the RC servos to the prototype leg. Tests were run on the single prototype leg to verify the motion and the estimated load requirements. The prototype tests were successful and proved the design would work. The aluminum proved to be lightweight, strong and not too difficult to machine showing decision to use aluminum was a good one.

3.6.3 Fabrication and Assembly
Fabrication and assembly of the final parts began after the prototype design had been built and verified. The hexapod was made from raw materials and manufactured in a machine shop. While some parts were computer numerical control (CNC) machined, other parts were crafted by hand. The exact methods used to fabricate the parts are detailed below.

Bushings and Spacers
Many bushings (108) and some spacers (12) with an inside diameter of 2mm and outside diameter of 5mm were required for construction of the robot. The bushings were required so that periodic maintenance may be performed by replacing worn bushings instead of machining a new leg. These parts were not available off the shelf and needed to be custom manufactured. Hollow aluminum rod was purchased meeting those requirements and the only tool found to cut without damaging the stock was a small pipe cutter. This proved to be a very time consuming process. Belt sanding was used to shape the spacers and bushings to an exact size since the pipe cutter was not as accurate as needed and did not leave a flush surface. Bushings were press-fitted into the robot using a pair of vice-grip pliers. This process produced some damage to the bushing and surrounding aluminum. Any damage was easily repaired using some sandpaper to refinish the surface and a drill to ream out the hole.

Leg segments
Drawings of the leg segments were glued onto the 2mm thick sheet of aluminum. First the bushing hole centers were marked with a punch for future drilling. The individual pieces were then cut out using the jigsaw. In the next step bushing holes were drilled out on the drill press using a bit matching the outer bushing diameter. The bushings were then press fitted into the legs. Finally the legs were finished and cleaned on the belt sander. The hexapod’s legs were then assembled using nuts and machine screws as pins. The nuts were secured to the screw with Loctite medium strength thread locker to prevent loosening over time.

Body
The body layout was cut using a 3 axis CNC mill. The Inventor drawing was exported to Master CAM and a g file created for programming the CNC mill. The body was cut from the same 2mm sheet of aluminum material used for the legs.
*Leg brackets*

The initial build of the leg brackets was a complete failure, the scribe marks on half the leg brackets were not done in the correct location setting us back an entire day worth of work. However, more stock was found and instead of hand milling the pieces a CNC program was made to reduce the chance of measurement error and prevent more fabrication setbacks.

The individual components were assembled with the RC servo motors to produce what is shown in Figures 4, 5 and 6.

![Figure 4: Base with Hip Motors](image)

![Figure 5: Assembled Legs](image)
Figure 6: Base with One Attached Leg
4 Electronics

4.1 Introduction

The servo motors were controlled by a central microcontroller. Each motor required a dedicated control line to control its position. Power is supplied by two separate battery banks, one for logic components and one for motors. An RS-232 level shifter was used to bridge between the microcontroller and a PC serial port. All the components were then mounted onto a custom manufactured PCB.

4.2 Control Electronics

The main controller of the six-legged walker was a PIC microcontroller. The robot required the following breakdown of I/O and functionality:
- 1 Timer
- 12 Outputs for motor control
- 1 input for mode selection
- 2 lines for transmit and receive of a serial data connection
- 2 inputs for bump sensors

There is a large variety of PIC microcontrollers available from Microchip ranging from six pin devices with four I/O pins to large packages with over 70 I/O pins and a generous amount of memory. After some research the PIC 16F876 was selected as the main controller. The PIC 16F876 has 22 general purpose I/O pins, 3 internal hardware timers and 14,336 bytes of flash program memory. In addition to supplying the requirements outlined above the 16F876 offers in circuit debugging and programming functionality to streamline the development process.

The controller was clocked using a 20 MHz resonator connected to the oscillator pins on the PIC 16F876. The MAX 233 level converter chip was used to convert TTL logic to RS-232 signal and vice versa to facilitate communication between the remote control and the robot. Two bump sensors were installed on the front of the robot. The bump sensors were made using micro switches with wire extensions to provide adequate reach. By using RC servos as actuators for the system the need to design and implement motor driving circuits and motor position sensors was removed.

The six legged walker was powered by Nickel Metal Hydride (NiMh) batteries. The NiMh batteries were designed for high current drain devices. Due to the fact that the motors were drawing a lot of current, the batteries drained quickly. Also it was discovered as the battery voltage was reduced it caused the microcontroller to execute a brown out reset. To remedy this problem two battery banks were used. One battery bank was for logic control and the other battery pack was for motor and electronics control of the robot.
4.3 PCB Fabrication

The robot walker circuitry was built onto a printed circuit board as shown in Figure 7. The PCB layout was performed using schematic capture software with the ability to route traces. The PCB traces were printed onto a transparency for masking. A pre-sensitized single-sided copper-clad board was exposed, developed, and etched to create the custom PCB. The required component holes were drilled and individual devices soldered into their appropriate locations.

Figure 7: Assembled Control Electronics
5 Software

5.1 Introduction

The software for the PIC microcontroller was written in PIC-C. It was compiled and transferred onto the chip via a USB in circuit debugger module. The six legged walker had two modes, automatic and manual. If the robot was set to manual mode the robot responds to user typed commands from the terminal window. If automatic mode was selected the walker will walk forward until a bump sensor is activated. When either bump sensor is activated, the robot will attempt to avoid the obstacle encountered by the active bump sensor. The main components of the software were the gait controller, command parsing and obstacle avoidance. A block diagram of the software is shown in Figure 8. The source code of the robot controller is contained in Appendix C.

![Figure 8: Software Block Diagram](image)

5.2 Methods of Leg Movement

There are several different methods of moving the six legs of a walker robot to create locomotion. These types include ripple, wave and tripod gait.

5.2.1 Ripple Gait

The ripple gait consists of two independent wave gaits. The two waves are opposite each other and 180 degrees out of phase. The power stroke time of the ripple gait is twice the lift time. The ripple gait takes three beats to complete one cycle and each side consists of a local wave with non-overlapping lift phases. The two opposite side waves are exactly 180 degrees out of phase with each one another. An example of the ripple gait is shown in Figure 9.
5.2.2 Wave Gait

The wave gait is the most stable gait among all three types of gaits. It allows smooth forward leg movements and five legs are always on power stroke. The wave gait takes six beats to complete one cycle that makes it the slowest gait. When all legs on one side moved forward at same time, motion starts at the rear leg and progresses forward. The same procedure is repeated on the other side. The robot always remains very stable, because only 1 leg is lifted at a time. The other 5 legs are stationary when the sixth leg is moving. Unfortunately the wave gait cannot increase in speed; if the swing time is shorter the steps become shorter. An example of the wave gait is shown in Figure 9.

5.2.3 Tripod Gait

The tripod gait is both statically and dynamically stable where three robot legs are always on the ground. It takes two beats to complete one cycle, which makes it the fastest gait among all three gait types.

Tripod gait is the most popular gait for hexapods. The hexapod consists of two tripods. The front-back legs on left side and the middle leg on the right side makes up one tripod and vice versa for the other side. For each grouped tripod the legs are moved forward, up and back at the same time. It uses the 2 tripods as a biped would use two legs for walking. An example of the tripod gait is shown in Figure 9.
5.2.4 Gait Comparisons

If the step size is held constant, the Tripod gait will be fastest, completing a cycle in two steps. The wave gait is the slowest among all three types at six steps. By overlapping the local side waves, the ripple comes in at a fast three time beats per cycle, although the phasing-offset actually produces six mini-beats overall.

The wave gait will be most stable, since it keeps the most legs on the ground at all phases of the stride. It is also the easiest to adjust during movement over uneven terrain because only the one moving leg needs to be adjusted. The next most stable is the ripple gait since four legs are always contacting the ground. At any given time only one leg is being lifted with a maximum of two legs off the ground at any moment in time. Although the tripod gait produces the fastest motion, it is also the least stable, since it always has three legs off the ground. Nevertheless the tripod still maintains three legs on the ground at any time and does remain stable enough to not require additional balance control to be implemented.
5.3 Implementation of Gait Controller

The gait controller consists of a PWM signal generator and a gait control FSM. The signal generator is required to output 12 independent channels each with a variable pulse width for RC servo control. Based on inputs from the other modules the FSM decides what the leg positions should be to produce desired motion.

5.3.1 PWM signal generator

Twelve motor control channels were implemented using an interrupt timer with a resolution of 150µs and 12 digital output pins on the microcontroller. Motor position resolution is not paramount; however the position must be repeatable. This is because the walking motion is always done in full steps and the steps must always be the same size. A timing resolution of 150µs is more than adequate yielding a precision of 13.5º. Hardware timer interrupts on the PIC are used to generate 12 variable pulsewidth signals. Each signal’s pulsewidth can be independently varied from 0ms to 21ms in steps of 150µs. the period of each signal is a constant 21ms. The listing below is a pseudo-code implementation of the algorithm used to produce the desired outputs with a single hardware interrupt.

```plaintext
interruptCount = 0
forever
    wait for timer interrupt
    if (interruptCount = 0) then
        set all outputs high
    end if
    for each output
        if stopCount >= interruptCount
            set output low
        end if
    end for
    interruptCount = (interruptCount + 1) mod 140
Loop
```

**Listing 1: Variable Pulsewidth Generation Algorithm**

5.3.2 Gait FSM

To produce the selected gait two Moore FSMs, one per side, were implemented with their outputs defining which joints should be moved. A complete step of the robot leg consists of four joint motions. Thus the number of states in the FSM is steps/cycle × 4. The FSM for the tripod gait requires eight states and is shown below in Figure 10. The transition between the states occurs after sending 25 identical pulses to the RC servos, approximately 0.5 seconds. The turning and walking motions are generated by specific FSMs transitions as outlined in Table 1. In order to preserve synchronization, motion type may only switch between walking and turning when left and right sides are in the same state. The number of steps in any direction for the robot to take must also
controlled. The algorithm in Listing 2 ensures that the change of motion only occurs when it will maintain synchronization and that the robot stops after the specified number of steps.

if (motion = walking and new motion = walking) or (motion = turning and new motion = turning) then change direction to the requested new direction

else if state of left FSM = state of right FSM then change direction to the requested new direction

if robot has completed requested number of steps then change direction to stop

Listing 2: FSM Transition Direction Determination


5.4 Command Parsing

The command parsing module is only active when the mode is set to manual control. The built in hardware UART was used to communicate with the computer remote control. The six legged walker acts as DCE and provides a prompt at the DTE. Commands were received from the terminal and checked for correct syntax. If the syntax is incorrect the user is shown an error message on the terminal and prompted again for a new command. If the command is correct it is executed and prompts for the next command.

Each command consists of two arguments. The first argument is a letter specifying direction and the second argument is a natural integer specifying the length of motion. Table 2 summarizes valid commands.

Table 2: Serial Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>f{X}</td>
<td>Walk forward {X} steps</td>
</tr>
<tr>
<td>b{X}</td>
<td>Walk backward {X} steps</td>
</tr>
<tr>
<td>l{X}</td>
<td>Turn left {X} steps</td>
</tr>
<tr>
<td>r{X}</td>
<td>Turn right {X} steps</td>
</tr>
<tr>
<td>s{X}</td>
<td>Stop motion, {X} is ignored</td>
</tr>
</tbody>
</table>

{X} represents a one digit number ranging from 0 to 9, and 0 executes indefinitely

5.5 Obstacle avoidance

When the robot is operating in autonomous mode the obstacle avoidance module is active. The obstacle avoidance logic was designed as the algorithm shown in Listing 3. The behavior of the obstacle avoidance is as follows. The six legged walker will start by walking forward, if it encounters an obstacle it will backup and then turn away from the object before walking forward again in an attempt to avoid it. If at any point during avoidance the sensors indicate an obstacle the robot will backup and turn again. Once the avoidance algorithm is complete the robot begins walking forward again.

```java
if a sensor is hit then
    remember which sensors are hit
    backup until all sensors deactivated
    continue backing up for one full cycle
else if finished backing up then
    turn to the opposite side of the activated sensor for one cycle
else if not turning and not backing up then
    walk forward until a sensor is activated
```

Listing 3: Obstacle Avoidance Algorithm
6 Results

A six legged walker was designed and built from raw materials as shown in Figure 11. There was lots of labor involved in the machining and maintenance of the robot. The screws and nuts used as pins in the legs of the robot proved to be quite problematic. Even with thread locker applied the joints often loosened introducing slop in the mechanisms. With the increased looseness in the joints the legs began to operate poorly. The bushings also proved problematic. The inside diameter of the bushings was not well matched to the outer diameter of the pins. This introduced more give in the mechanism than what was expected. A more suitable and better designed system for the joints would have produced a much more capable and robust robot. Off the shelf robot kits have taken this into account and provide well designed joints which do not degrade with usage. Although two degrees of freedom per leg does provide locomotion, two degrees of freedom does not provide sufficient control and maneuverability necessary to navigate uneven terrain. Also the lack of tactile feedback from the legs does not permit the controller to adapt to the terrain. Having a walking robot with greater leg maneuverability and tactile feedback would aid the robot in its ability to navigate hazardous terrain.

![Figure 11: Picture of the Completed Robot](image)

Obstacle avoidance was implemented into the robot. The sensors used were rudimentary and physical contact had to be made before an obstacle was detected. More sophisticated
sensors like computer vision and sonar would enable the robot to avoid obstacles prior to contact. During obstacle avoidance the walker backs up and turns blindly because there were no sensors in the rear or sides. For the robot to more effectively navigate the terrain more information would need to be provide to the controller. The additional information could come from using more sophisticated sensors or using additional sensors. With an increased amount of available information a more sophisticated obstacle avoidance system could have been implemented.

The joint and sensor issue became apparent towards the end of the project. Due to time constraints these problems were not addressed. However much has been discovered from the design, manufacture and testing of the six legged walker.


7 Conclusions

The goal of this project was achieved successfully. A six legged walking robot was designed and constructed without a kit. The scope of the project involved the designing and building of the leg mechanisms, motor linkages, and required software to control the robot autonomously and manually. Although the original requirements were met some design flaws were uncovered during testing. Time constraints prevented the rectification of these deficiencies.
8 Recommendations

The robot was successfully made to walk. However there were some deficiencies, including:

- Poor joints
- Limited maneuverability of legs
- Primitive environmental sensing
- Elementary obstacle avoidance.

To remedy the poor joints it is recommended that a proven, perhaps more complicated, joint design should be used. A higher degree of freedom can be used to increase the maneuverability in the legs. These mechanical issues could be resolved by using an off the shelf robot kit. Sonar could be used as a more sophisticated method of detecting obstacles removing the need for physical contact. To facilitate a more sophisticated obstacle avoidance system, sensors should be added to the back and sides of the robot.
9 References

Analysis of Multi-Legged Animal + Robot Gaits, Jan 2003, available at:
http://www.oricomtech.com/projects/leg-time.htm#Hex2

http://www.tic.ac.uk/micromouse/guide/index.asp

PIC 16F876 Datasheet, 2001, available at:

MAX 233 Datasheet, 2004, available at:

Frank Scott's Hexapod Robots, available at:
http://www.frasco.demon.co.uk/

D.J. Todd, Walking machines : an introduction to legged robots, 1985, Kogan Page Ltd.

Hong-Sen Yan, Creative Design of Mechanical Devices, 1998, Springer-Verlag

Appendix A – Logistics

Division of Tasks

The project tasks were defined and divided amongst the group members as shown in Table 1.

Table 1: Group member project tasks assignment

<table>
<thead>
<tr>
<th>Task</th>
<th>Details</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanics</td>
<td>- Design of mechanical legs and overall implementation of robot</td>
<td>Kiel, Jason</td>
</tr>
<tr>
<td></td>
<td>- Creating and assembling mechanical parts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Implementing motors into robot chassis</td>
<td></td>
</tr>
<tr>
<td>Electronics</td>
<td>- Designing control electronics for servo motors</td>
<td>Belinda, Jason, Kiel</td>
</tr>
<tr>
<td></td>
<td>- Prototyping electronics and building circuit board</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td>- Developing routines to actuate motors in the proper sequence</td>
<td>Kiel</td>
</tr>
<tr>
<td></td>
<td>- Creating autonomous routines for obstacle avoidance</td>
<td></td>
</tr>
<tr>
<td>Report</td>
<td>- Documentation of project</td>
<td>Jason, Belinda, Kiel</td>
</tr>
<tr>
<td></td>
<td>- Schematics and flowcharts</td>
<td></td>
</tr>
<tr>
<td>Webpage</td>
<td>- Design and creation of report contents into a web site</td>
<td>Jason, Kiel</td>
</tr>
</tbody>
</table>

Task Completion Timetable

The project tasks were scheduled to be completed by the dates listed in Table 2.

Table 2: Completion Dates of Project Tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project definition</td>
<td>Jan 14, 2005</td>
</tr>
<tr>
<td>Hardware design</td>
<td>Jan 7, 2005</td>
</tr>
<tr>
<td>Hardware creation</td>
<td>Feb 11, 2005</td>
</tr>
<tr>
<td>Electronics design</td>
<td>Jan 28, 2005</td>
</tr>
<tr>
<td>Electronics creation</td>
<td>Feb 4, 2005</td>
</tr>
<tr>
<td>Software design</td>
<td>Feb 18, 2005</td>
</tr>
<tr>
<td>Software creation</td>
<td>Feb 25, 2005</td>
</tr>
<tr>
<td>Connecting hardware and software components</td>
<td>Mar 4, 2005</td>
</tr>
<tr>
<td>Testing and calibration</td>
<td>Mar 25, 2005</td>
</tr>
<tr>
<td>Project completion and poster presentation</td>
<td>Apr 1, 2005</td>
</tr>
<tr>
<td>Final report and webpage submitted</td>
<td>Apr 8, 2005</td>
</tr>
</tbody>
</table>
Project Part Sources

The project parts are listed in Table 3.

Table 3: Major Parts Sources and Costs

<table>
<thead>
<tr>
<th>Qty</th>
<th>Description</th>
<th>Part#</th>
<th>Price</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIC Microcontroller</td>
<td>PIC16F876</td>
<td>11.16</td>
<td>Digi-Key</td>
</tr>
<tr>
<td>1</td>
<td>RS-232 line driver</td>
<td>MAX233</td>
<td>9.73</td>
<td>Digi-Key</td>
</tr>
<tr>
<td>1</td>
<td>20MHz resonator</td>
<td>ECS-SR3-20.00-B</td>
<td>1.08</td>
<td>Digi-Key</td>
</tr>
<tr>
<td>12</td>
<td>RC Servo motors</td>
<td>GWS S03N</td>
<td>198.00</td>
<td>Solarbotics</td>
</tr>
<tr>
<td></td>
<td>Misc material</td>
<td>---</td>
<td>15.00</td>
<td>Stock</td>
</tr>
<tr>
<td></td>
<td>Misc resistors</td>
<td>---</td>
<td>---</td>
<td>Stock</td>
</tr>
<tr>
<td></td>
<td>Misc fasteners</td>
<td>---</td>
<td>12.00</td>
<td>Canadian Tire</td>
</tr>
<tr>
<td>1</td>
<td>Copper Clad Board</td>
<td>---</td>
<td>10.00</td>
<td>Queale</td>
</tr>
</tbody>
</table>

| Total Cost | 256.97 |

Appendix B - Schematic
# Appendix C – Source Code

## Controller.c

---

Walking Robot Control System v1.0

firmware component of 499B project

This software offers autonomous and tethered control of lil'bugger.

Left and right bump sensors for autonomous control is connected to C5 and C4 respectively.

12 RC servo motors are connected on A0-A5 and B0-B5

Mode select is connected to C3 high=>auto, low=>tethered

RS-232 is used for accepting user commands, formatting is specified below.

---

```c
#include "controller.h"
#include <ctype.h>
#include <stdlib.h>

const int8 kneeCCW = 13;
const int8 kneeCW  = 7;
const int8 HM  = 10;
const int8 hipCCW = 14;
const int8 hipCW  = 6;

//                         fwd hip, fwd knee, mid hip, mid knee, rear hip, rear knee

const char errorString[] = "\r\nError: Invalid Input!";

const int8 fwdSequenceLeft[8][6] = {
    {hipCW, kneeCW, hipCCW, HM, hipCW, kneeCW } //Lift a,c
, {hipCCW, kneeCW, hipCCW, HM, hipCCW, kneeCW } //Fwd a,c
, {hipCCW, kneeCW, hipCW, HM, hipCCW, kneeCW } //Rev b
, {hipCCW, HM, hipCW, HM, hipCCW, HM } //Down a,c
    //----------------------------------------//
[hipCCW, HM, hipCW, kneeCW, hipCCW, HM ] //lift b
[hipCCW, HM, hipCCW, kneeCW, hipCCW, HM ] //Fwd b
[hipCW, HM, hipCCW, kneeCW, hipCW, HM ] //Rev a,c
[hipCW, HM, hipCCW, HM, hipCW, HM ] //Down b
};

const int8 fwdSequenceRight[8][6] = {
    {hipCW, HM, hipCCW, kneeCW, hipCW, HM } //Lift e
, {hipCW, HM, hipCW, kneeCW, hipCW, HM } //Fwd e
, {hipCW, HM, hipCW, kneeCW, hipCCW, HM } //Rev d,f
    //----------------------------------------//
[hipCCW, kneeCW, hipCW, HM, hipCCW, kneeCW ] //Lift d,f
[hipCW, kneeCW, hipCW, HM, hipCW, kneeCW ] //Fwd d,f
[hipCW, HM, hipCCW, HM, hipCW, HM ] //Down d,f
};

int8 state=0;
int32 count=0;
int16 countDown;
int8 infiniteTime;
int8 moveTimeup;
signed int8 leftSideDir=0;
```
signed int8 rightSideDir=0;
signed int8 newLeftSideDir=0;
signed int8 newRightSideDir=0;
signed int8 leftPosIndex=7;
signed int8 rightPosIndex=7;
int8 directionReady=1;
int8 rightLatch=0;
int8 leftLatch=0;

////////////////
startAllPulses()
{
    output_bit(motorLeft1,0);
    output_bit(motorLeft2,0);
    output_bit(motorLeft3,0);
    output_bit(motorLeft4,0);
    output_bit(motorLeft5,0);
    output_bit(motorLeft6,0);
    output_bit(motorRight1,0);
    output_bit(motorRight2,0);
    output_bit(motorRight3,0);
    output_bit(motorRight4,0);
    output_bit(motorRight5,0);
    output_bit(motorRight6,0);
}

///////////
endPulses()
{
    output_bit(motorLeft1,((motorLeftStops[0]>=state)?1:0));
    output_bit(motorLeft2,((motorLeftStops[1]>=state)?1:0));
    output_bit(motorLeft3,((motorLeftStops[2]>=state)?1:0));
    output_bit(motorLeft4,((motorLeftStops[3]>=state)?1:0));
    output_bit(motorLeft5,((motorLeftStops[4]>=state)?1:0));
    output_bit(motorLeft6,((motorLeftStops[5]>=state)?1:0));
    output_bit(motorRight1,((motorRightStops[0]>=state)?1:0));
    output_bit(motorRight2,((motorRightStops[1]>=state)?1:0));
    output_bit(motorRight3,((motorRightStops[2]>=state)?1:0));
    output_bit(motorRight4,((motorRightStops[3]>=state)?1:0));
    output_bit(motorRight5,((motorRightStops[4]>=state)?1:0));
    output_bit(motorRight6,((motorRightStops[5]>=state)?1:0));
}

///////////
#int TIMER2
TIMER2_isr()
{
    //this should occur every 150us
    //first perform output logic
    if (state == 0){
        startAllPulses();
    }
    else {
        endPulses();
    }
    //now update state
    if (state >=140) //on cycle is finished
    {
        state = 0;
        count++;
    }
    else
    {
        state++;
    }
}
if (count >= 25) //have we sent 25 cycles (25 cycles -> 150us * 140 * 25 = 0.525s)
{
    if ((leftSideDir == rightSideDir && newLeftSideDir == newRightSideDir) || //first
        (leftSideDir != rightSideDir && newLeftSideDir != newRightSideDir) ) //second
    
    else if ((leftPosIndex == 7) && (rightPosIndex == 7))
    {
        leftSideDir = newLeftSideDir;
        rightSideDir = newRightSideDir;
        directionReady = 1;
    }

    if ( !((leftSideDir == 0) || (rightSideDir == 0) || (moveTimeUp==1)) )
    {
        leftPosIndex  = (leftPosIndex  + leftSideDir );
        if (leftPosIndex <= -1) leftPosIndex = 7;
        else if (leftPosIndex >= 8) leftPosIndex = 0;
        rightPosIndex = (rightPosIndex + rightSideDir);
        if (rightPosIndex <= -1) rightPosIndex = 7;
        else if (rightPosIndex >= 8) rightPosIndex = 0;
        memcpy(motorLeftStops , fwdSequenceLeft[leftPosIndex], 6);
        memcpy(motorRightStops, fwdSequenceRight[rightPosIndex], 6);
    }

    if (infiniteTime == 1)
    {
        moveTimeUp = 0;
    } else if (countDown == 0)
    {
        moveTimeUp = 1;
    } else if (directionReady==1)
    {
        countDown -= 1 ;
    }
    count = 0;
}

void main()
{
    char command     = 'n';
    char data        = '0';
    int8 isValidInput = 0;
    char dataString[2];
    int8 walkState;

    port_b_pullups(TRUE);
    setup_adc_ports(NO_ANALOGS);
    setup_adc(ADC_OFF);
    setup_spi(FALSE);
    setup_counters(RTCC_INTERNAL,RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DIV_BY_1,47,16);//gives a 150us interrupt period
    enable_interrupts(INT_TIMER2);
    enable_interrupts(global);

    directionReady = 0;
    leftSideDir = 1;
    rightSideDir = 1;
    newLeftSideDir = 1;
    newRightSideDir = 1;
    isValidInput = 1;
walkState=0;

if (input(runMode))
{
    while (true)
    {
        if ( (input(leftAnt) || input(rightAnt)) /*&& walkState==0*/ )
        {
            // backup 2 steps
            directionReady=0;
            newLeftSideDir = -1;
            newRightSideDir = -1;
            countDown = 8 * 1;
            infiniteTime = 0;
            moveTimeUp = 0;
            rightLatch = input(rightAnt);
            leftLatch = input(leftAnt);
            walkState=1;
        }
        else if (rightLatch && leftLatch && moveTimeUp && walkState==1)
        {
            // turn 2 steps right
            directionReady=0;
            newLeftSideDir = 1;
            newRightSideDir = -1;
            countDown = 8 * 1;
            infiniteTime = 0;
            moveTimeUp = 0;
            rightLatch = 0;
            leftLatch = 0;
            walkState = 2;
        }
        else if (rightLatch && !leftLatch && moveTimeUp && walkState==1)
        {
            // turn 2 steps left
            directionReady=0;
            newLeftSideDir = -1;
            newRightSideDir = 1;
            countDown = 8 * 1;
            infiniteTime = 0;
            moveTimeUp = 0;
            rightLatch = 0;
            leftLatch = 0;
            walkState = 2;
        }
        else if (!rightLatch && leftLatch && moveTimeUp && walkState==1)
        {
            // turn 2 steps right
            directionReady=0;
            newLeftSideDir = 1;
            newRightSideDir = -1;
            countDown = 8 * 1;
            infiniteTime = 0;
            moveTimeUp = 0;
            rightLatch = 0;
            leftLatch = 0;
            walkState = 2;
        }
        else if (moveTimeUp && walkState==2)
        {
            // continue walking forward
            directionReady=0;
            newLeftSideDir = 1;
            newRightSideDir = 1;
            countDown = 0;
            infiniteTime = 1;
            moveTimeUp = 0;
            rightLatch = 0;
            leftLatch = 0;
            walkState=0;
        }
    }
}
else
{
  while (true)
  {
    puts("\n\r>");
    
    /*
    command examples:
    f0  : walk forward until a new command arrives
    r9  : turn right 9 walk cycles
    s0  : stop forever
    available commands:
    'f' : forward
    'b' : backward
    'l' : left
    'r' : right
    's' : stop
    */
    command = getc();
    putc(command);
    isValidInput = 0;
    switch(command)
    {
      case 'f':
        directionReady=0;
        newLeftSideDir = 1;
        newRightSideDir = 1;
        isValidInput = 1;
        break;
      case 'b':
        directionReady=0;
        newLeftSideDir = -1;
        newRightSideDir = -1;
        isValidInput = 1;
        break;
      case 'l':
        directionReady=0;
        newLeftSideDir = -1;
        newRightSideDir = 1;
        isValidInput = 1;
        break;
      case 'r':
        directionReady=0;
        newLeftSideDir = 1;
        newRightSideDir = -1;
        isValidInput = 1;
        break;
      case 's':
        directionReady=0;
        newLeftSideDir = 0;
        newRightSideDir = 0;
        isValidInput = 1;
        break;
      default:
        isValidInput = 0;
        puts(errorString);
        continue;
        break;
    }
    data = getc();
    putc(data);
    isValidInput &= (isdigit(data)?1:0);
    if (isValidInput)
    {
    
    }
puts("\r\nCommencing Motion!");
 sprintf(dataString,"%c",data);
  countDown = 8 * atoi(dataString);
  infiniteTime = (data == '0') ? 1:0;
  moveTimeUp = 0;
}
else{
  puts(errorString);
  continue;
}
}

Controller.h
#include <16F876.h>
#define motorLeft1 PIN_A0
#define motorLeft2 PIN_A1
#define motorLeft3 PIN_A2
#define motorLeft4 PIN_A3
#define motorLeft5 PIN_A4
#define motorLeft6 PIN_A5
#define motorRight1 PIN_B0
#define motorRight2 PIN_B1
#define motorRight3 PIN_B2
#define motorRight4 PIN_B3
#define motorRight5 PIN_B4
#define motorRight6 PIN_B5
#define debuggerConnection1 PIN_B6
#define debuggerConnection2 PIN_B7
#define leftAnt PIN_C5
#define rightAnt PIN_C4
#define runMode PIN_C3
#define SER_OUT PIN_C6
#define SER_IN PIN_C7
#define fuses NOWDT, HS, PUT, NOPROTECT, BROWNOUT, LVP, NOCPD, NOWRT, NODEBUG
#define rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)