# HANNIBAL

IMDL    Summer 1999

by
Marc Poe

# Table of Contents

## Abstract

This report will document the design and construction of a walking robot with leg-contact feedback for stability while walking over rough terrain. The robot will also be able to determine when it can climb over an obstacle and how to maneuver around those obstacles it can not.

## Executive Summary

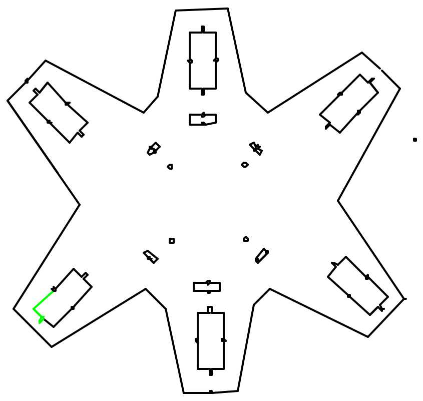
The Design for Hannibal will borrow from the anatomy of spiders, specifically keeping in mind the stability inherent in the octagonal arrangement of their legs. It will also be designed so that the legs rise up over the body when walking. In this way, the robot will be more capable to lift its entire body over large obstacles.

Unlike its eight-legged cousin, Hannibal is a six-legged robot that can maneuver through a cluttered room. The Robot will walk around at full height, enabling it to lift it legs over obstacles without concern for the type of obstacle it is. It will not require a level surface for walking, but will keep its balance based on feedback from micro-switches placed on the feet and a tilt switch on the body. If the robot's leg hits an object, micro-switch is pressed, while it is extending, the robot will know how to keep the leg in that position.. An Infrared emitter/detector pair will be placed on each of the front two legs and will be used to scan for obstacles that it can not maneuver over or around.

## Mobile Platform

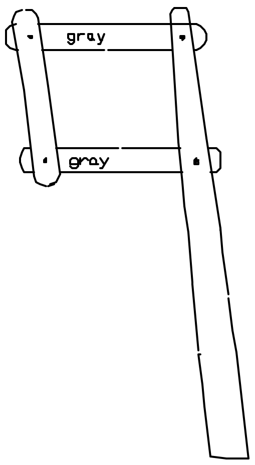
### Body

The body of Hannibal is constructed entirely of 5-ply model aircraft plywood and somewhat resembles a crab in that it has an upper and lower "shell" with the legs arranged around the middle. The two shell halves are separated by 4" supports resembles R.O.U.S. created by Megan Grimm in the Spring of 1999. The space between these supports will house the 6 C cell batteries required for the servos and a pack of 6 AA batteries for processor power supply. The upper body shell will have 1.8"x 1.4" holes cut around the edges for each of the six shoulder servos as shown in the figure below.



## Legs

. To move each leg independently at least two degrees of freedom are required Knowing this, I have designed Hannibal’s legs to allow them to move independently using a servo for each degree of freedom. The figure below illustrates the structure of the leg.

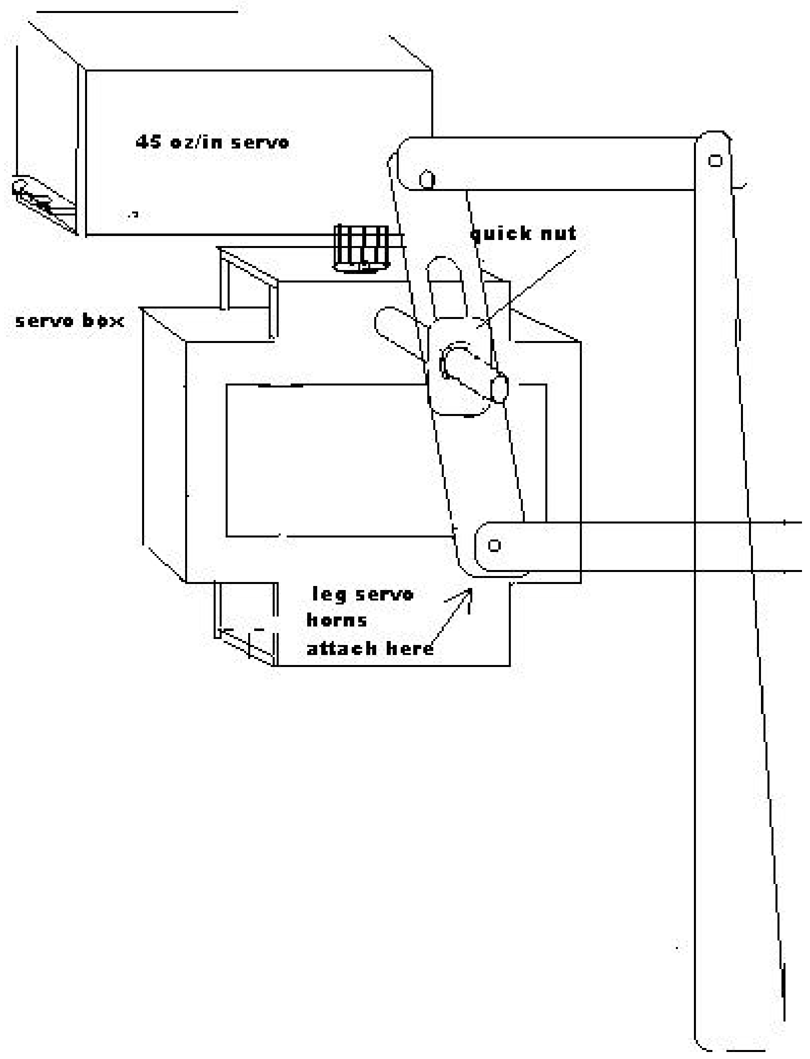


Pantograph Leg Mechanism

Each leg is basically a two dimensional pantograph mechanism, where a small movement at the shorter end translates to a greater motion on the larger end. The Mechanical advantage of the pantograph mechanism is directly related to the length of the gray segments in the figure above. A longer connecting segment enables a leg to have greater freedom of motion. However, this also increases the torque required to lift the leg. Since the servo strength and the weight of the platform, including the servos, limit the design we need the shortest foreleg possible to alloy the robot to maneuver over reasonable obstacles. I have chosen six 130 oz/in Cirrus servos from Tower Hobby Based on my torque calculations, if the platform was to stand on one leg, it would require 280 oz/in of torque. Since I intend to have no less than three legs on the ground at once, and three of these servos together have a max torque rating of 390 oz/in. I am well within the design requirements for leg torque.

I have designed a wooden box to house the leg servos. The upper part of these 'servo boxes' connects directly to the shoulder servos on the upper shell of the body. The lower section of the servo boxes is connected to the lower shell of the body. Nylon spacers are placed underneath the servo box, between it and the lower. Machine screws are inserted through the lower shell, the nylon spacers, and the lower part of the servo box. Lock washers and hex-head nuts are used to keep the box in place and allow the legs to swing.

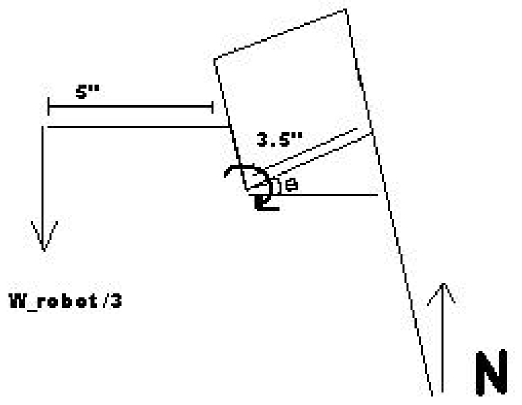
To secure the legs from wobbling, and to guarantee the legs will only move vertically, I drilled a 3/16th hole through the top of these boxes and milled out oval spaces in the vertical leg segments as a leg guide. Then I inserted 3 inch pieces of 3/16th brass tubing through both holes in the boxes and through the guide spaces with about 1/2 inch extending from the inside holes and out of the leg guides. I crimped the tubing on the inner side of the servo boxes and used 3/16th quick nuts to secure the leg from sliding horizontally along the tubing. Below is a diagram of the servo box used in the final design:



Nylon spacers are used between the ends of the fore leg and the main outer leg to allow easy movement rotation within the joints. The screws used to attach the lower foreleg segment to the servo horns interfered with the range of motion in the initial design. The interfering screw head reduced the over-all leg clearance to 5.5 inches. An extra set of nylon washers were used between the inner leg and the two fore-legs to allow full unrestricted a range of motion close to 180°. With a 9" main leg and two 3.5" fore legs, the overall range of motion of each leg is now approximately 7 inches. Although the leg clearance allows Hannibal to clear large obstacles, it also raises the center of gravity further from the ground, making it less stable

## Torque Calculations

For the robot to be statically stable, least three legs must be on the ground at all times. Each leg servo must therefore be capable of producing enough torque to counterbalance one third the entire weight at the end of a leg. The lower foreleg segment is connected directly to the servo. This fore leg is a 3.5 inch torque arm. In static equilibrium, 1/3rd of the total normal force will push against the end of the leg. Since the servo is approximately 5 inches from the robot's center of mass, the torque required to balance this force is Torque= (W_robot/3)* (3.5*cos $\theta$), where $\theta$ is as indicated in the figure below:

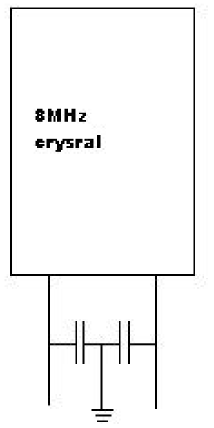


Since the weight of the robot is 4.5 lbs. , or 92oz, the maximum torque required is Torque =92/3 *(3.5*cos θ) = 107.3 oz/in. Since the leg servos can output 130oz/in of torque, they are sufficient to balance the robot against gravity. There are 23 oz/in of extra torque available. With this extra force, one servo can tilt the platform off balance. If Hannibal is walking over rough terrain and a leg servo is hard-coded with a PWM value to send it to its maximum position, and it strikes an object closer than the maximum position, the servo has enough force to make the platform unstable. Therefore, we must use a closed loop feedback system to sense when a leg has struck an object, and to stop moving it when this occurs.

## Actuation

Each of Hannibal’s legs is driven by two servos to provide two degrees of freedom. One 42 oz/in servo provides a sweeping rotation motion from front-to-back and back-to-front. The other leg servo (130 oz/in) lifts the leg in an up-and-down manner.

Due to the complexity of controlling all 12 of these servos, a separate processor is required for generating the pulse-width-modulated signals. A 68HC11 single-chip board from Mekatronix will be used as Hannibal's servo controller since ports B and C are available on this board. This board also has a power and ground bus surrounding each port which can either be used with the regulated 5V powering the chip itself, or an external power supply can be used. This is well suited for driving servos since they typically require 6 Volts for full torque and they draw so much current that the microprocessor will reset if both servos and the processor are powered from the same battery pack. I will use the 68HC11E2 since it has the highest, and readily accessible, internal memory of any processor in the E-series. A Motorola 68HC11(A1) evaluation with 32Kbytes of external RAM, two output ports at $2000 and $3000, and an input port memory mapped at $2000 will be used to incorporate the sensors and generate the signals required to control the servos. At first I intended to use the MTJPro board produced by Mekatronix, since it has a everything the EVBU board does plus a built-in 40kHz generator for modulating the IR LED's and is much smaller than the EVBU. However, I accidentally shorted power to ground on the first board I built and fried all of the circuitry. When I built the second one, Mekatronix was out of the low-profile 8MHz crystals the MTJPro boards were designed for and I had to resort to modifying a regular sized crystal  based on the following diagram:



Where the capacitors are 22pF.  Since I experienced such trouble with the prefabricated boards, I had to build my own 40kHz generator using a  74HC390 to step down the processor's 8 MHz clock to 40kHz for the Infrared LED's.

## Description of Software for Actuation.

In early July I received a copy of the  library files developed by Ivan Zapata for the base platform of RoboBug.  It included functions specifically designed to control up to 16 servos using Ports B and C on  the single chip board.  It also included a program designed to allow the user to calibrate the minimum, middle, and maximum positions for particular servo and generates the integer value ( between 1000 and 4900) corresponding to these servo positions.  Upon  execution of this calibration program, the following table was produced:

| Servo # | Middle | Up/ Forward | Down/ Back | Acronym for servo |
|---|---|---|---|---|
| 2 | 2260 | 1000 | 3360 | RFL-- right front leg |
| 3 | 2730 | 1930 | 3570 | RFS --right front shoulder |
| 4 | 2430 | 1200 | 3870 | RML-- right middle leg |
| 5 | 2430 | 1730 | 3570 | RMS--right middle shoulder |
| 6 | 2500 | 1070 | 4800 | RRL-- right rear leg |
| 7 | 2400 | 1700 | 3610 | RRS-- right rear shoulder |
| 8 | 2200 | 3870 | 1000 | LFL--left front leg |
| 9 | 3280 | 4170 | 2460 | LFS--left front shoulder |
| 10 | 2200 | 3940 | 1000 | LML--left middle leg |
| 11 | 3180 | 3940 | 2400 | LMS--left middle shoulder |
| 12 | 2200 | 4000 | 1000 | LRL--left rear leg |
| 13 | 3280 | 4170 | 2200 | LRS--left rear shoulder |

Since the code for integrating all the sensors and controlling servo actuation will most certainly require more than 2K of memory, and the MTJPro is only capable of controlling 5 servos, some type of inter-processor communication is necessary. The serial communications interface is the simplest method to implement this since. The program on the single ship will obtain the necessary signals to control the servos, through the SCI interface, from the EVBU board. A dummy character will be sent to initialize communication, then EVBU board transmit the number of the servo to move and the position to send it to. The single-chip board will, in turn, receive the servo number, initialize that servo and transmit the proper pulse width. However, since the final pulse widths can range from 1000-4900, and the largest integer size is 255, the proper pulse widths must be calculated using a general algorithm. . Since this range of integers also corresponds to the maximum range of the servos (180° ) there is approximately (180°) /

(3900 $\Delta$PWM)= 0.046°/ PWM. If we want the error to be within 2° of the actual value, the value of the PWM sent must be $\pm$ 44 of the desired value. If we simply multiply the PWM value sent through the serial port by 44 we obtain the desired pwm with a 98% accuracy. With a 3.5 inch torque arm this corresponds to an overshoot of $ds=r*d\theta(rad)=$ 3.5 in *2.02° *($\pi$*/180°)= 0.125inches.

## Walking Gaits

To maneuver across rough terrain, Hannibal must not only be able to vary its leg position based on terrain. However, this if its walking gait is not stable it can't possibly maneuver over smooth terrain. The most difficult aspect of walking robots is designing a dynamically stable gait because the center of gravity shifts throughout the gait. The key to designing a stable gait is the position of the shoulder servos before and after leg placement.

### Tripod gait

This gait is by far the fastest insect-like walking gait. For stability reasons the bug must have three legs on the ground at all times. In the tripod gait, the middle leg on one side and the front and rear legs on the other side lift and sweep forward, while the other legs extend and sweep backwards. Of all the different gaits, this one was most difficult to implement. The robot's weight is easily supported by three legs, but when it would propel itself with only three legs on the ground, stability was lost and Hannibal tended to tip forwards.

## Metachronal Wave Gait

This is the most stable of the gaits I experimented with. During a wave gait, the robot always has at least 5 legs on the ground at all times. The walk starts at the rear leg on each side. The leg lifts, swings forward, and after it is placed on the ground, the leg in front of it is swung forward and planted on the ground. The wave moves up each side, and when it reaches the front leg, the gait switches to the rear leg on the opposite side.

## Oar Gait

This gait, though somewhat slower than the tripod gait, was fastest and most stable gait I experimented with. Four legs are on the ground at all times to keep it level. The gait begins as the middle legs lift swing forward, touch ground and swing back. Then the left front and right rear legs lift, swing forward and touch down, and swing back. Next, the right front and left rear legs swing forward, down and back.

## Turning

To turn in place, a walking routine similar to the Oar gait described above is used. To spin counterclockwise in place, all legs will move similar to the Oar gait routine except the legs on the right side will lift, swing back, place, and swing forwards. To spin clockwise in place, all legs will move similar to the Oar gait routine except the legs on the left will lift, swing back, place, and swing forward.

## Sensors

### Contact Switches

To keep Hannibal stable while walking over rough terrain, contact feedback from the legs is necessary to keep the leg in its current position when it steps on an obstacle. Otherwise, the robot will attempt to keep pushing against the obstacle, which could possibly tip it over. Every servo is designed with a specific gear ratio, which determines how long it takes for the servo to reach a certain position. Knowing this, we can design a contact feedback sensor that will alter the PWM signal sent by the processor to the servos and keep the servo in its current position.

To keep the robot stable, it must be able to sense when one of its legs solid ground. To accomplish this, I must be able to change the Pulse-width modulated signal being sent to the servos to keep them at the position when the feet hit the ground.

The servos I am using for leg extension are 130 oz/in with a 60°/.22s transfer rate. Since a servo requires a 20ms PWM signal with time high between .7ms and 1.7 ms and will rotate only 180° in either direction, we need to determine the time elapsed from the beginning of the leg extension until the foot hits a surface. The angular distance traveled by the leg is given by the following equation :

distance[d] =angular velocity[w] * time [t].

Since the torque arm is 3 inches long, the vertical distance traveled by the leg with respect to time is d=[60° /.22sec]*t. Therefore, the ration of the distance traveled to the total possible distance is [(4*π/3)*t]/π. The required PWM signal to send the servo to one of its maximum positions way has a time high of 1.7ms and the other has a time high of .7ms. This signal is equivalent to the ratio of the distance traveled to the total possible distance. Therefore, the new PWM must have a high time of 1.7ms*[(4*π/3)*t]/π, where t is the elapsed time elapsed during the leg extension. Simplification produces the following equation:

New time high (T) =(4/3)* 1.75ms * [1E clock/500ns] * elapsed time (t).

Some of the servos will require a signal with approximately a .7ms high time. For these, the following equation will be used :

New time high (T) = [ 2.1 ms *1E clock/500ns] /(4* elapsed time(t)

The bump switches will be connected directly to the microprocessor using the memory mapped digital input port at $2000. The figure below is a diagram of the bump switches as they were connected to the digital input port:

Implementation.

The above description for altering the PWM signals being sent to the servos to control leg position was so difficult to implement that a different approach was needed. In mid July I was given a set of libraries developed for RoboBug by Ivan Zapata. These libraries included a routine for controlling up to 16 servos. The function takes in two variables, a servo number and pulse width, and generates the PWM signal for that servo continually until it receives a different pulse width. To implement the touch sensor, I created a loop that polls the foot switch of each leg as it extends towards the floor incrementing the PWM value sent to the servo controller by 2 each time through the loop. Each time through the loop the program waits for approximately 1ms for the switch bouncing to settle. Though this delay means the legs will extend somewhat slower than if the loop was running wide open, a delay is absolutely necessary for debouncing the switch. Without it, the loop program will miss the switch feedback entirely. Whenever the switch closes, the program exits the loop and the servo position is held. The servo controller function requires that the PWM sent to the servos be within 1000 and 4900. Since this range of integers also corresponds to the maximum range of the servos (180° ) there is approximately $(180°) / (3900\ \Delta PWM)= 0.046°/\ PWM$. Since the servo controller automatically multiplies the integer it receives for the PWM signal by 44, an error of $\pm 2.02°$ will always be present. Also, the leg extension loop increments the signal it sends to the servo controller by 2 each time through the loop. Thus, when the foot switch is pressed the leg will have a minimum overshoot of 4.04°. With a 3.5 inch torque arm this corresponds to an overshoot of $ds=r*d\theta(rad)= 3.5$ in $*4.04° *(\pi*/180°)= 0.25$inches. This quarter of an inch error, also ensures that extra pressure is applied to a leg while it is in contact with the ground and is so slight that it would not significantly affect the robot's balance.

## Infrared

Two Sharp 40 kHz IR detectors hacked to produce analog signals will output analog ports 1 and 2 used for collision avoidance. One infrared detector and emitter pair will be mounted on each of the front two legs. As the legs sweep forward, Hannibal will scan for obstacles. Since we are only concerned with obstacles that we can not climb over, scanning will only take place while a leg is at its full height. Therefore, if we detect an object close to the leg, we know that Hannibal can not possibly climb over it.

However, after it has climbed a step, Hannibal may end up in a crouched position and will be unable to climb over successive objects unless it can return to its full stance. To solve this problem, I have placed one hacked Sharp IR detector directly beneath the center of the robot and two high-power IR LED one in forward and one aft, each pointing at a ° down from the body. When Hannibal's body is too close to a large obstacle, it will know to rise up to full stance position--thus enabling it to climb over successive obstacles. With all of the sensors working properly, we will be able to maneuver over rough terrain without knowing exactly what kind of terrain it is. To the robot, stairs would just look like a series of obstacles, and as long as there is enough space on the ledge of each stair for the robot, Hannibal will be able to climb onto it.

## Behaviors

Hannibal will exhibit general obstacle avoidance, but with special consideration to its own ability to maneuver across rough terrain.  In other words, if it can climb over an obstacle, it will; otherwise, it will maneuver around the obstacle.   It will also vary its gait based on the number of rough obstacles it has stepped on. The more frequent Hannibal has to stop a leg short of its maximum extension the gait will become slow then change. As it moves across smooth terrain to rough terrain, Hannibal will change from Tripod gait to the Oar gait.  If the terrain is very rough, it will slow to a Metachronal Wave gait.

## Murphy's Law

Nothing could have prepared me for the complexity of this project. Although I finished the mobile platform within the first 6 weeks, I spent almost two weeks during the second semester working out bugs in the Mekatronix MTJPRO11 boards I built. The first time I built one of these boards, it tested correctly. I set it aside until the platform was finished. Once, I accidentally connected the power and ground to both the single chip board and the MTJPRO11board backwards. This blew the 5V regulator on the single chip board, and both the 74HC573 and the 74HC138 chips on the MTJPRO board. I replaced these components and the single chip was back in working order. I spent a week developing and testing different gaints using the single chip alone. However, it did not have enough memory required to integrate the sensors, gaits, and the corresponding servo actuation. I required the 32K of the TJRO board to incorporate the system.

At one point I realized that the crystal was not working properly--possibly broken after dropping it on the ground At this point, Mekatronix ran out of the low profile 8MHz crystals used on their TJPRO boards. I modified a standard crystal with two 22pF capacitors, as discussed above, but this still did not fix my problems. At this point I decided to assemble another one to replace it. I spent three days building and testing the new board. The new board never worked properly and, as a last ditch attempt and with a week to go in the second semester, I switched to using an EVBU board since I knew it worked. With all these hardware problems, I was left with a week to complete all the necessary software. I was able to incorporate the feet sensors and the robot walked, but I ran out of time and was not able to implement the IR sensors in software.

## CONCLUSION

Despite the time, money, and Murphy-inspired migraines, I believe that I received considerably more from this class than I put in. The mechanical design of my robot was much too complicated for my first robot design.  Though I was able to implement most of my plans for Hannibal, time caught up with me and I was unable to integrate all the sensors with the proper behaviors.  The complexity of the software for controlling all the different gaits and sensors overwhelmed me.   After dealing with faulty electronics for most of Summer B, I was left with just a week to develop all the software needed for the different behaviors. Ultimately, my design did not work properly because Hannibal remains unfinished. The Intelligent Machine Design Lab not only gave me design experience, it also made me realize the real-world limitations to any project--namely time, and Murphy's Law.