

The problem

I have been intrigued with the idea of building a walking robot that can perform a certain task. A walking robot in the future would have the potential to climb over difficult terrain. With this in mind, I am planning to build a prototype robot that could aid in a search and rescue effort. The idea of building a robot that can do this is not new, but the creation of an autonomous walking robot that can aid in a search and rescue effort is unheard of.

What I plan to do

I plan to build a four-legged robot with these objectives in mind:

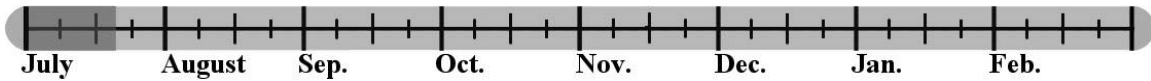
- **Use an Atmel AVR 8-bit processor to control the robot.**
- **The robot should be built out of standard parts (RC servos, easy to fabricate plastic material, micro controller with good development tools)**
- **Be built for under \$1,500**
- **Be small, light and powerful**

Why four legs?

I believe that a four-legged robot that is close to the ground will be able to shift its weight in small amounts to balance its body. To keep the robot close to the ground I will configure the robot's legs like a spider. I will also put the robot's batteries close to the ground to give it a low center of gravity.

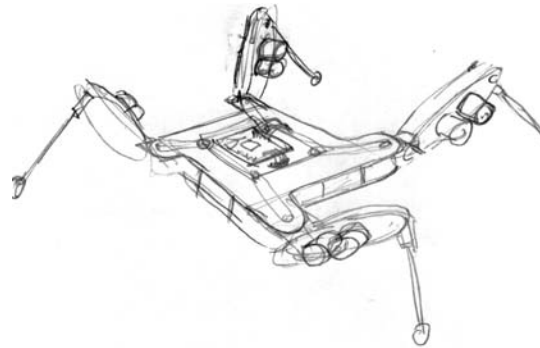
What the robot will do

I plan to build a walking robot that is small enough to fit in a suitcase. This robot will be able to avoid obstacles and also have the ability to use information given by a remote operator. The robot will use a sonar range finder to look at its surroundings. I hope to make a robot that will not replace the robots that are in use today. I plan to change the way we think about robots for the future. I think there is a large potential for the use of walking robots that has not yet been tapped into. Hopefully, this project will help open the doors to the use of new types of robots.



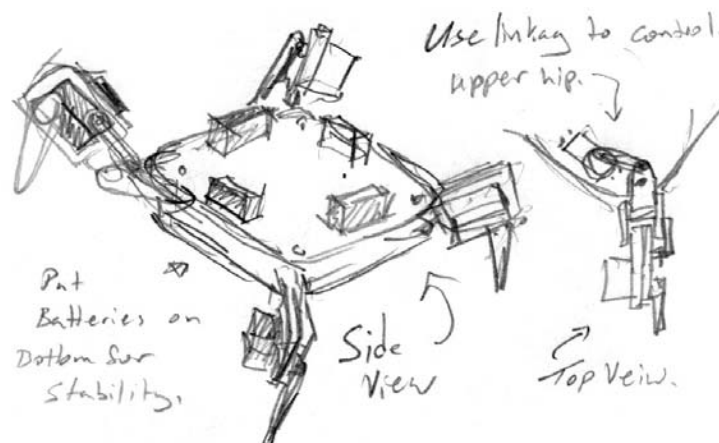
The ideas

Earlier I had been playing with the idea of using cables and springs to attach DC gear motors to the legs of a robot. This might give the robot the ability to perhaps jump. I would have made the robot out of aluminum to make it light and strong. I am not going to pursue this because of time limitations and machining resources. An embedded PC would have controlled the robot. Here is a drawing of what the aluminum robot would have looked like.



I am now starting to design a robot that will use RC servomotors to move its legs. By using RC servos I will be able to keep the cost of the robot down. The servos are also very lightweight, which will be helpful in making the robot walk. I plan to control this robot with an Atmel AVR micro controller. My main goals when designing this robot are lightweight and strong. By making the robot as small as possible I can make the robot light. By making the distance between axes of freedom in the legs as small as possible I can make the legs more powerful.

Here is a picture of what the robot might look like.

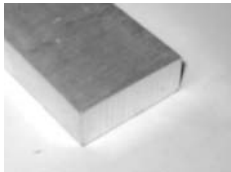




Deciding on a robot frame material

Where to start

My first step is to decide what to build the robot out of. I am quite sure that the robot needs to be built out of some type of plastic. To research different plastics I went to SPI Plastics in Tacoma, WA. I ended up deciding on a lightweight product called expanded PVC, which, unlike the piping, is a high density styrofoam. Here are some of the materials I have considered for the robot.



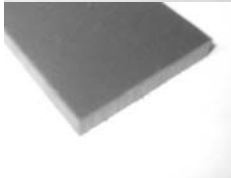
Aluminum

Aluminum, although light and strong, is time consuming to machine.



Acrylic plastic

Acrylic is somewhat easy to machine but it is quite heavy.

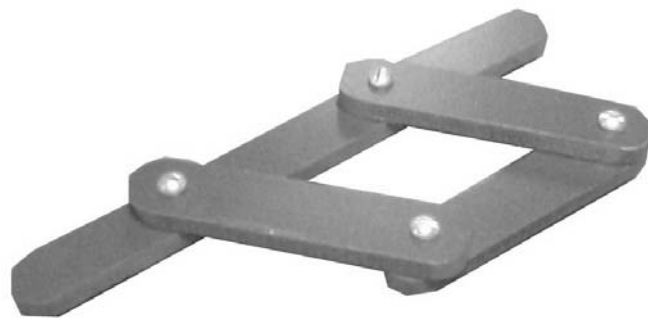


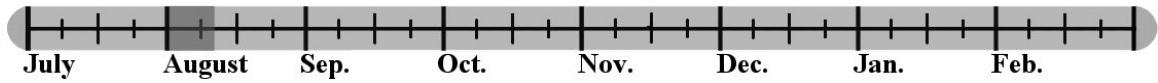
**Expanded PVC
(High density
styrofoam)**

Expanded PVC is very easy to machine and is very light-weight.

After buying two 24 x 48 inch sheets of PVC, I now need to build a couple of test structures to test the plastic's ability to be cut, screwed and used as a bushing. I also need to test the plastic's overall strength.

Here are two structures I have made to test the plastic's ability. What I found was that the plastic can be fabricated easily, screwed together without tapping the material first. It also makes a fair bushing.

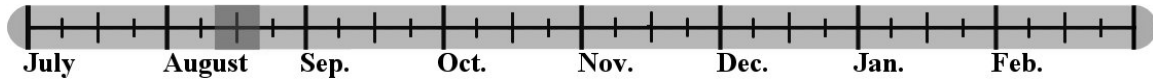




Building a test structure of the robot

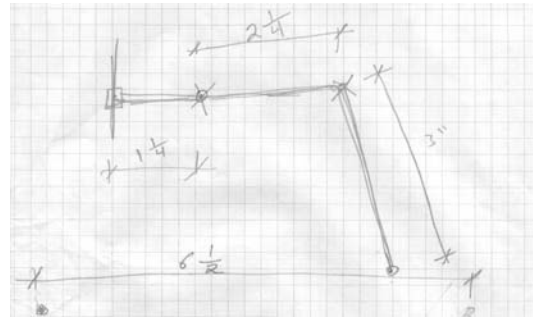
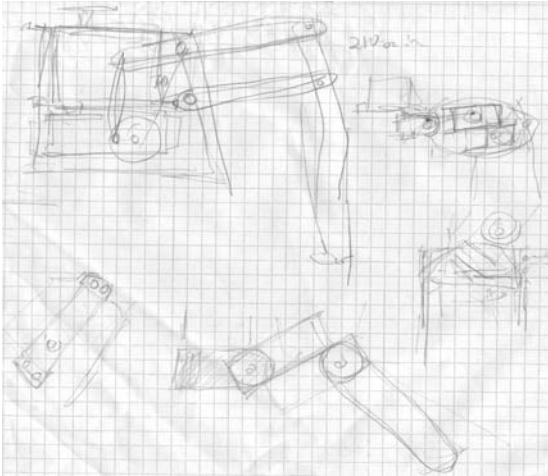
I have built a four-legged model of the four-legged robot. This model allows me to see how the robot will walk. It also helps me understand any potential problems in the general design before I start to build the actual robot. I did not realize the range of motion the robot would have until I built this model. I also realized that the smaller the base of the robot, the less the robot would need to shift its weight to balance.



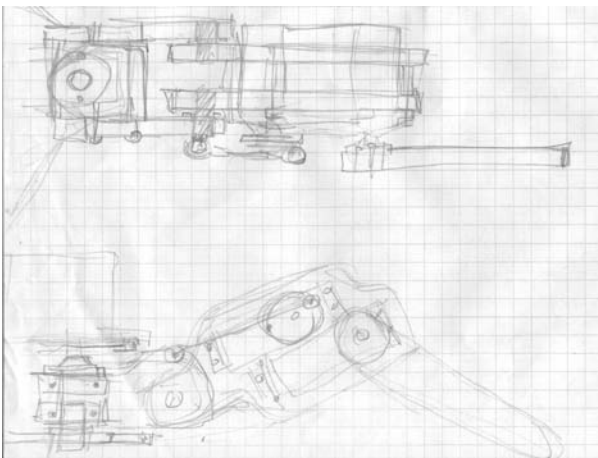


Designing the robots leg

One of the problems that I am faced with is how to configure the legs on the robot. The legs are using standard servomotors for power. When designing the leg, the goal is to have as much range of motion as possible. I also need the distance between the axes of freedom to be as short as possible to give the robot more power. Here are some design drawings I have made.



A large challenge in designing the leg is having enough power to support the robot. Certain joints in the robot will be able to lock by using linkages between the motor and the upper leg. When the joint is locked there is a minimal amount of stress on the motor. The linkage also limits the range of motion in the hip along with giving the leg different amounts of torque at different leg positions.



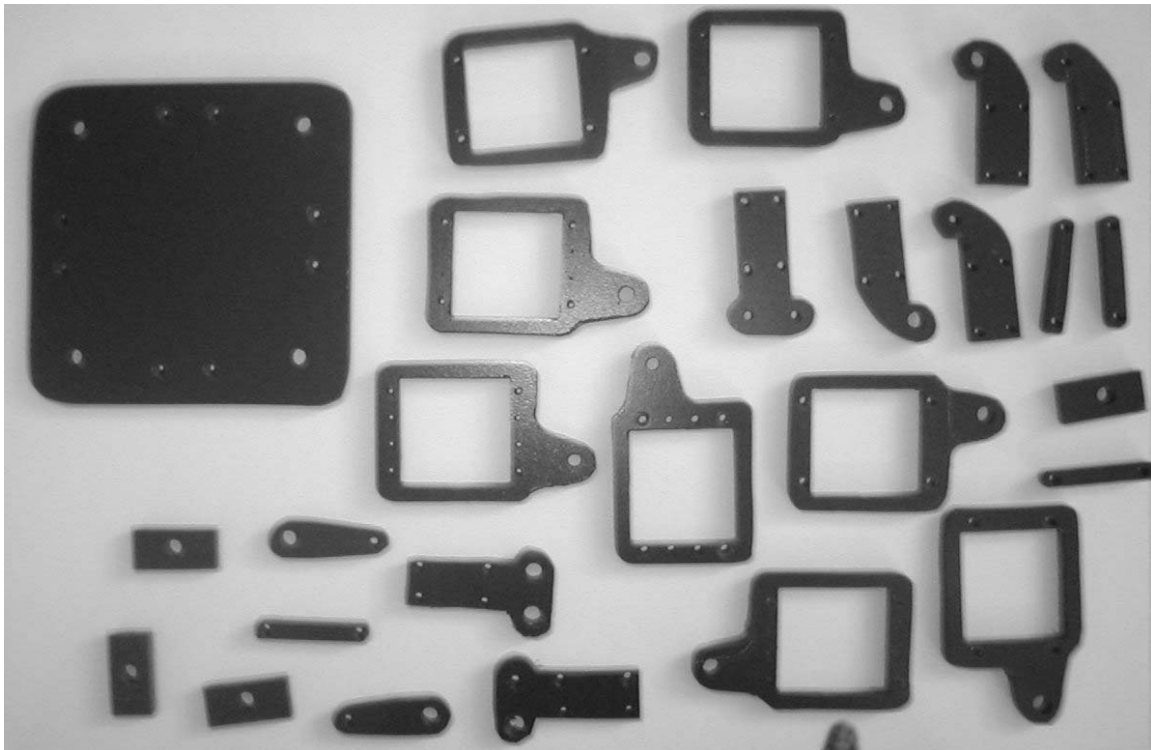
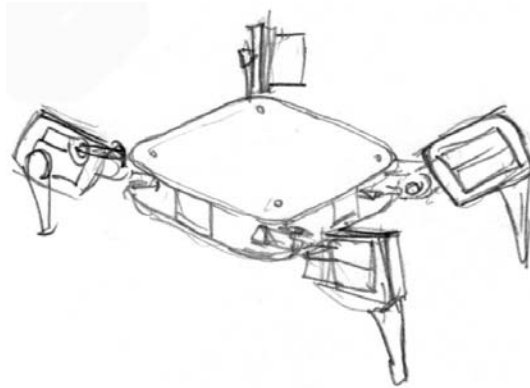
I have settled on this type of design because it is compact and free moving. The linkage should be able to give the robot plenty of power for lifting the robot.



The four legged robot is not looking good

While assembling the robot's legs, I have noticed there is a problem with the linkage that connects the servo to the upper hip. There is too much slop in the leg joints. This might be a problem later.

After working a couple days, I have finished the robot's mechanics. In an attempt to fix the slop problem I noticed earlier, I accidentally destroyed the robot's joints. I now am starting to research different ways of fixing the slop problem. Here is a picture of what is left of the four-legged robot and a drawing of what the robot looked like before it was drilled.





Redesigning the leg hardware

After destroying the four-legged robot's joints, I noticed another problem with the robot's design. The 6-32 screw and nut I was using for the hip of the legs to pivot were causing the bolt to cut away at the hip. Before I rebuild the base of the robot I must find a better way to transfer the energy from the hip servo to the hip. I must also find a better hip pin that does not cut away at the hip. I am starting to research the use of linkages from model airplanes and different pins from the hardware store.

Different designs for the hip linkage

	<p>Old linkage using 6-32 screws to pivot on.</p>	<p>Makes the robot's joints sloppy, making movement in the leg inaccurate.</p>
	<p>New type to linkage from RC airplane systems.</p>	<p>This linkage is extremely accurate and is free moving.</p>

Different designs for the hip pin

	<p>6-32 screw</p>	<p>This screw works as a pin but the screw's threads cut the surrounding material, which causes slop in the joint of the robot.</p>
	<p>Nylon pin</p>	<p>This pin does not damage the PVC but is manufactured at a variation of sizes. The variation in size causes the joint to be either too stiff or too loose.</p>
	<p>Aluminum pin</p>	<p>The aluminum pin is consistent in size and will not damage the surrounding leg material.</p>

Problem:

There is too much slop in the robot's leg

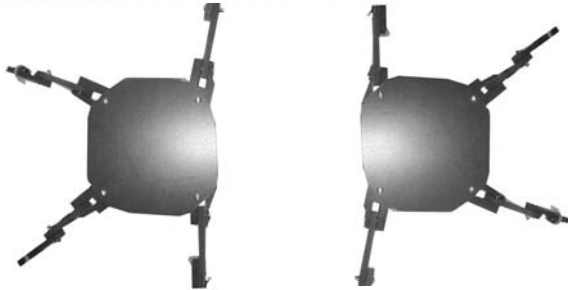
Solution:

Use a linkage made for RC planes and an aluminum pin from the hardware store.



Abandon four-legged base design

I now think that I will not be able to use the four-legged design. The problem with the robot's design is that it needs to shift its weight. For the robot to be able to walk it must shift its weight about 2 inches either way. To fit all of the electronics needed on the robot, the base must be at least 6-7 inches square. The legs also need to be as small as possible to have enough power to support the weight of the robot.



When the robot shifts its weight enough to be able to lift a leg, the robot's legs do not have the range of motion required to take a step. I realized this would be a problem, because the base I would have to build would be too big for the four-legged robot to balance.

I am now considering building a five-legged robot instead of a four. Here are some things I must consider before I do.

Pros of five-legged robot design

- More stability
- The robot does not need to shift its weight to be stable
- The size of base will not affect the performance of the robot
- Five legs is more rare than four
- Gives the robot more ability to walk in different directions

Cons of five-legged robot design

- Will cost more for extra servos
- Will take time to redesign new robot base
- Harder to program because there is not an example in nature

Problem:

The four-legged base is not practical because of stability and base size issues

Solution:

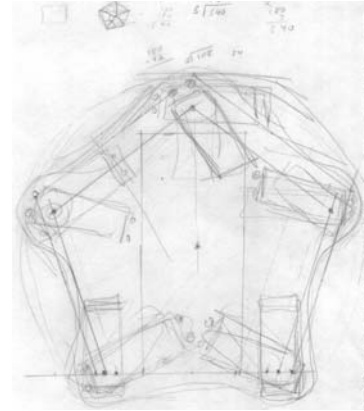
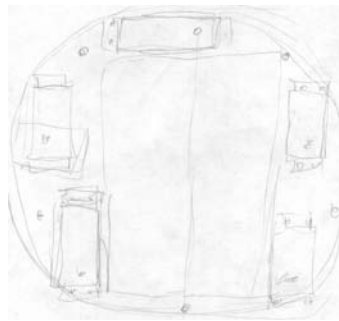
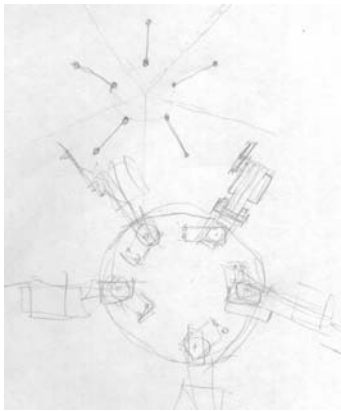
Try a five-legged base design



Design ideas for five-legged robot

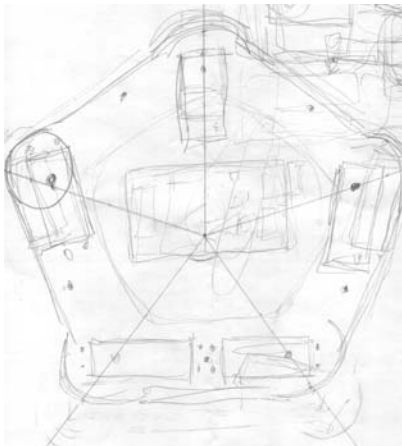
Design objectives

- Make the robot as small as possible
- Have enough room for two RC car batteries on the bottom of the robot
- Have batteries on the bottom of the robot for stability
- Give legs maximum range of motion



Here are some of the design drawings I have been making for the new base design.

Designing this new robot is a bit tricky. The major design challenge I have been solving is the where and how the batteries will fit. The drawing to the right shows how the servos are oriented in different directions. This is to accommodate the batteries. For the final design I will use a base with rounded edges like the drawing on the far right. This system will give more range of motion to the legs.



I have decided to use this configuration for the servos. Two batteries will be able to fit side by side, pointed towards the top of the robot.

I don't think that Velcro would be strong enough to hold the batteries, so I might just make a plate that pins to the bottom side of the robot's base.



Constructing the five-legged design

Each leg is identical in design.

Problem:

The legs of the robot take a long time to make.

Solution:

Build metal template to speed up the time it takes to build the leg.



Here is a picture of the template I made.

The robot is coming together nicely. I have noticed the robot is easier to build the second time around.

Here are some of the tools I am using to build the robot.

Layout tools



The tools shown from left to right are:

- Dial calipers
- Veneer calipers
- Machinist compass
- Razor knife
- Scribe
- Ruler

Fabrication tools



Milling machine



Band saw



Drill press



Sander



I am just about done with the robot's mechanics

I have configured the batteries to fit between the robot's servos.

Problem:

The screw in the upper hip of the robot is cutting the base of the robot because of movement in the leg.

Solution:

Use nylon bushings around the screw to protect the robot's base.



Here is a diagram showing the two nylon bushings I am using to prevent the robot from cutting itself.

One of the things I have done in the design of the robot is to make the corners of the robots base rounded. This gives the legs more range of motion and strength in the hip joint.



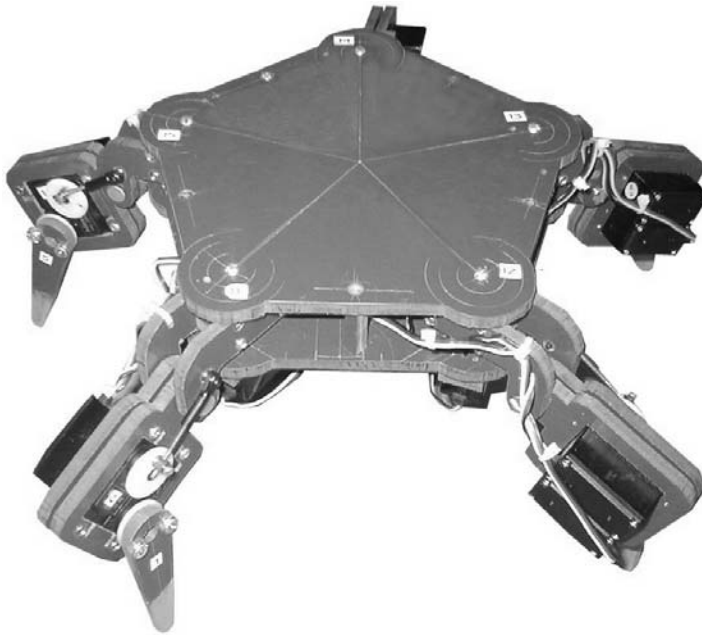
The corners being rounded on the robot's base is the end result of the design. What gives the legs on the robot more range of motion, is the pentagon shape of the robot's base.

While making one of my routine trips to the hardware store, I noticed this can of rubber dip for coating tools. I have used this rubber to put traction on the robot's feet.





I finished the robot's mechanics!



Here is a picture of the robot with all of the main design problems solved. I am confident the mechanics will hold together. It is hard to say how long the robot's servomotors will hold out.

Now I can start putting a brain on the robot. The robot is looking more life-like every time I work on it. I wonder how fast the robot will walk?

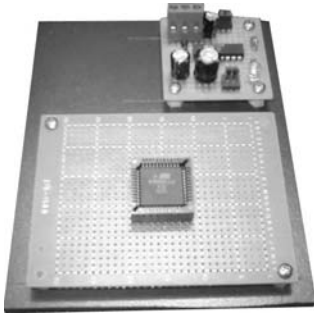
My decision to round the corners of the robot was a good one. The robot now has ten to twenty more degrees range of motion in the hips.

I am happy with the rubber dip that I put on the feet of the robot. It looks like the rubber will stay on the robot well. I might use the rubber in the future to insulate hard to get to wire connections.

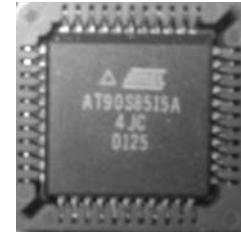


The Atmel AVR is not going to work

I have decided that the Atmel AVR is not going to be powerful enough to control the five-legged robot. Up until this point I have researched the AVR, made a board to test it, and gathered some programming tools on the Internet.



I have built a test set-up for the Atmel AVR processor (shown on right). Although the AVR is an extremely powerful processor, it will not be practical to build the robot with it because of the amount of processing power the robot needs.

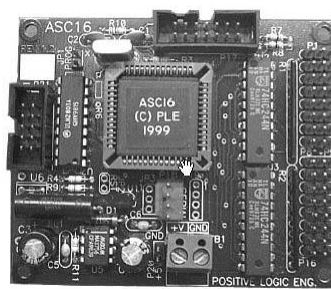


I am considering the use of a Palm Pilot and a commercially available servo controller to control the robot.

Advantages of using a Palm / Servo controller

- The Palm Pilot will not be bogged down if a servo controller is used.
- The servo controller has software to test positions of the servos from a computer.
- There is a compiler available “PocketC” which allows an application to be programmed in C.
- The servo controller has the ability to connect outside information to the Palm Pilot.

I have decided to use the Palm Pilot instead of the Atmel AVR processor



Shown here are the Palm Pilot on the left and the SC16 servo controller on the right.

Problem:

The Atmel AVR processor is not powerful enough to run the robot's complex algorithms.

Solution:

Use a Palm Pilot and servo controller to control the robot's movements.



The robot has a brain

I have interfaced the Palm Pilot, servo controller and the robot's servos together.

Problem:

The servo controller's serial line input voltage must be no greater than five volts.

Solution:

Use a Max232 serial level converter to protect the servo controller.

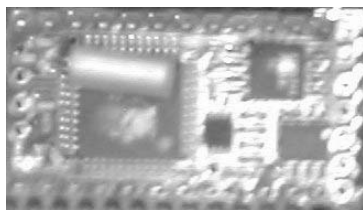
I am using a commercially available level converter board with surface-mount components on it. The board is barely visible in the picture behind the SC16 label.



This picture shows the link between the Palm Pilot and the SC16 servo controller. I have left the serial connectors for the palm and SC16 open so I can easily read values in the robot with a computer. The palm connector lets me download programs into the palm without having to take the Palm out of the robot.

I have also decided to add a third processor to the robot called a BasicX.

The BasicX is a micro controller that will be programmed to receive information from a RC receiver and a sonar range finder. Depending on whether a RC signal is received or not, the BasicX will determine if the robot is in a RC or autonomous mode. If the robot is in the RC mode, the robot will relay the RC information to the Palm Pilot. If the robot is in autonomous mode, the robot's head will look at its surroundings with the sonar range finder, and determine the best direction to pursue.



Here is a picture of the BasicX micro controller that I will be using on the robot to perform various tasks outside of the walking and mapping routines on the robot.



Writing software to run the motion scripts.

I am writing a program for the Palm to send motion scripts to the robot's legs.

Problem:

On the palm I have a program that can compile and then download motion scripts. The program that compiles the motion scripts takes too long to compile the motion scripts every time I want to use them.

Solution:

Break the program into two programs, one to compile the scripts and one to download the motion scripts. The information will be stored in database files by the compiler. The down loader will read the database files.

Problem:

Some of the motion scripts are very large and when sent, the servo controller's serial line buffer gets over loaded.

Solution:

Break the motion scripts into smaller parts that can be sent to the servo controller in packets instead of all at once.

Software description

Compiler:

This is a program that extracts servo motion information from a memo document on the Palm. This information is then translated into an array of numbers that the servo controller can understand. This array is then saved in a database file for later use. The characters in the database file can be sent directly from the database file to the servo controller through a serial line to control the servos.

Down loader:

This program reads information saved in database files by the compiler. After reading this information, the program downloads the file to the servo controller. The down loader also has the ability to wait for the servo controller to be ready for the next packet of information before sending it.



The robot moves by itself

I have made a connection between the Palm Pilot and the servo controller. The robot can now move its motors from commands in a compiled motion script.

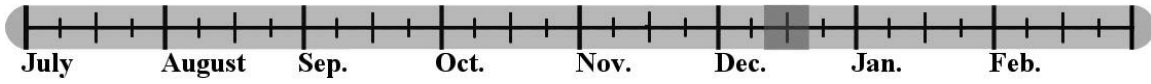
I have just written a working motion script that makes the robot move in a push-up motion.

What this proves:

- The robot's electronics and software have the potential to run complex motion scripts to make it walk.
- The motor configuration on the robot is strong enough to support the weight of the robot.

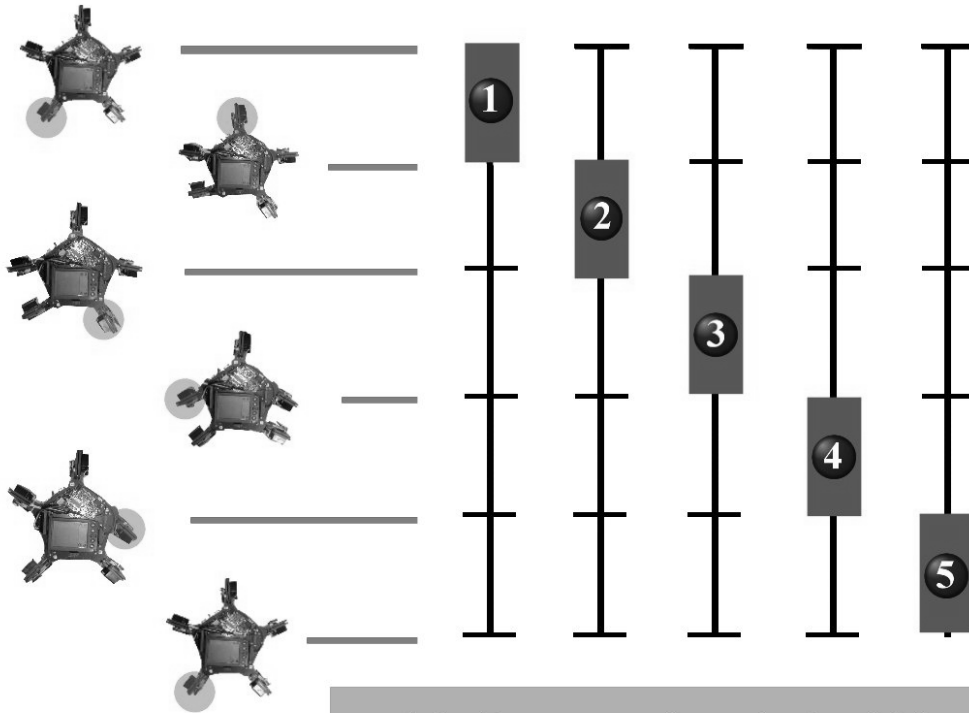


This is a picture of the robot in the down position of the push up routine. Through the motion script, I am able to change the speed that the robot moves. I am impressed by the amount of force the robot can push up with. It feels to be at least a couple of pounds of force being exerted on my hand when I try to push down the robot while doing the push up routine.

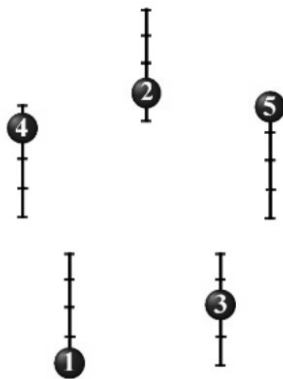


I am now faced with the problem of making the robot walk. Here is a description of the algorithm I have planned to use for the robot.





These diagrams show one cycle of the robot's gait. For the robot to walk continually the gait is repeated.



Walk Forward

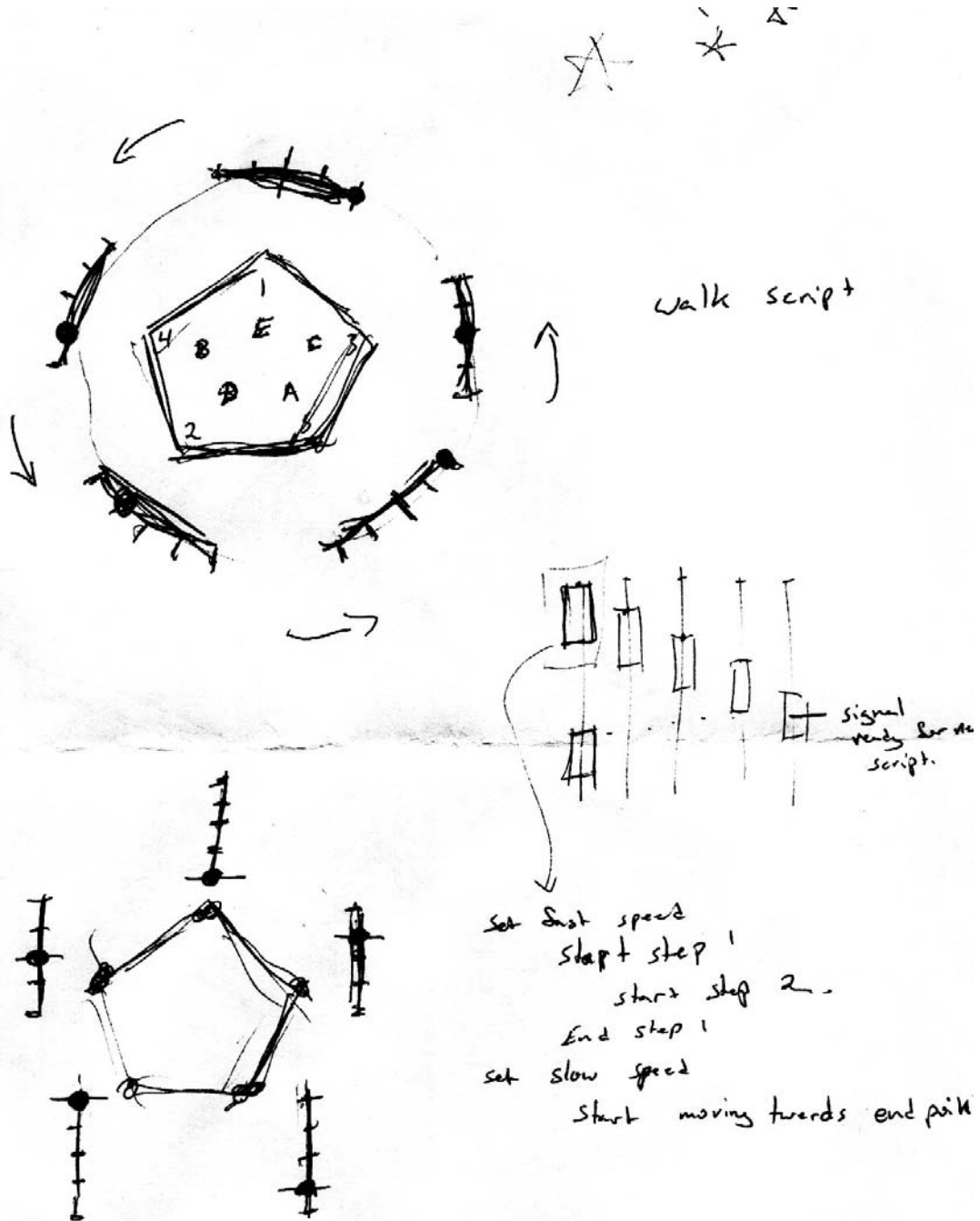


1-5 Represents the order in which the legs take a step.

-  - Represents one of the robot's legs taking a step.
-  - Represents one of the robot's legs sweeping to the end of the legs travel.
-  - Represents the starting position of each leg at the beginning of the gait.
-  - Represents a leg before it takes a step.

By using this type of gait the robot can always have two legs taking steps in succession while having the robot's center of mass supported by the other legs. These legs form a tripod which keeps the robot stable.

Here is a copy of the actual drawing I made before I programmed the robot to walk. I made the drawing during a not too interesting lecture in math class. There is also a diagram for a turning algorithm I might use later.





The robot walks!

After programming the robot for a couple of days, I have been able to make a motion script that makes the robot walk in one direction.

I am going to try making a walking algorithm, which has the robot order its steps in the same manner but waits less time between steps. This should make the robot walk twice as fast as it does right now.

I was able to make the robot walk with the new routine, but the algorithm did not give the robot enough stability to be dependable. The new walking method was not as fast as I thought it might have been.

I am going to use the old walking algorithm for stability reasons.

While programming the robot I have noticed a problem with a couple of servos on the robot jittering. I can't link the problem to the servo controller, so I am going to try replacing the jittering servos.



I have just ordered 4 new servos to replace the servos that are jittering on the robot. I will probably order another five or six servos just in case a few of them start to act up in the future.

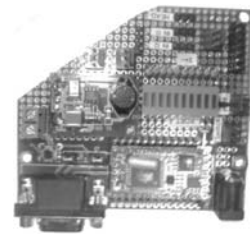
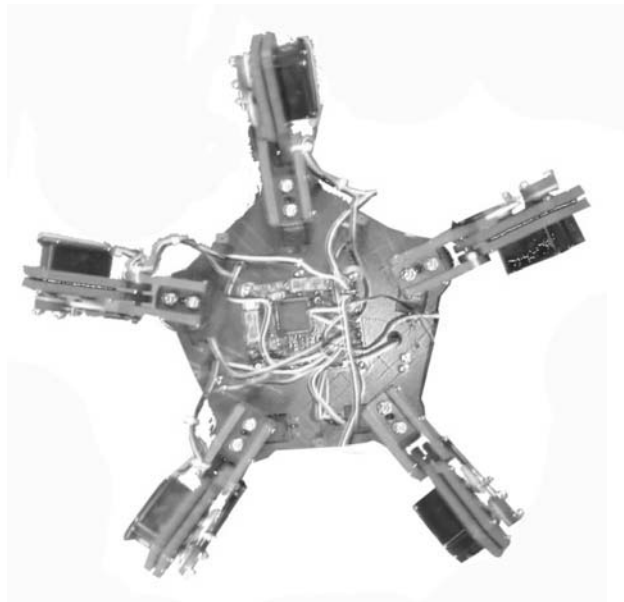
My next step is to write the backwards walking routine. This should be easier since I have written the forward routine.



I have finished the backwards walking algorithm.

The robot's walking algorithms for forwards and backwards are set up to start in the same position. By doing this the robot can change its direction from forwards to backwards easily.

There is now a connection between the BasicX and the SC16 servo controller. This allows the BasicX to send an eight-bit word of information to the Palm Pilot. This information link will be used in the future to send navigational information to the Palm Pilot.



The BasicX is connected to the SC16 servo controller by a 14 line ribbon cable. On the BasicX shown above the cable is connected to the small row of pins on the upper right of the board. On the SC16 shown on robot to the left, the connector is located just above the black chip in the center of the board.

A RC receiver unit is now attached to the BasicX. By reading pulses sent by the RC receiver, the BasicX can determine the control values sent by a RC transmitter. The robot will use this information to determine the direction the robot should travel in.



This receiver was out of an old RC car I had. I will be reading two signals from the receiver, one for turning and one for the forward back movement.



Interfacing the RC receiver and the BasicX

I have written a program on the BasicX that uses information from the RC receiver to distinguish between forward, backward and stop. This information is then sent to the Palm Pilot.

Problem:

The signal from the RC receiver floats up and down, making values inconsistent.

Solution:

Put the incoming RC values into a three layer FIFO software stack. By returning the average of the stack the stream of values are smoothed out.

Interpreting navigational information with Palm

At the end of each motion script sent to the servo control there is a piece of code that reads the value of the port the BasicX is writing to. This value is then sent back to the Palm to use for navigating.

I have written a program for the Palm Pilot that takes the direction information from the BasicX. With this information the Palm will send either the forward, backward or stop routine. The stop routine is just a dummy script that is sent to the servos.

I had trouble at first writing the software to receive the signal from the servo controller. The software is using a serial line buffer to help in the task of reading the stream of information from the SC16. Palm waits for a certain flag in the serial stream. When the flag is seen, the next byte received by the serial line is the intended byte sent by the SC16.

With the Palm and BasicX programs running together, the RC transmitter can control the robot.

My next step is to write a motion scrip to make the robot turn left and right.

The robot will have two starting positions. One is for the walking forward and backward routine. The other starting position is for walking left and right. I need to write a motion script to move from the turning algorithm start position to the walking algorithm starting position. I will also need to write a script for the walking to the turning algorithm.



Building - Programming the head

I am building a head for the robot. The head will have a video camera and a sonar range finder on it. By turning the head, the robot will be able to find different obstacles in front of it.

Problem:

The Sonar range finder and the video camera both need a clean 5volt power supply. The voltage spikes from the motors in the robot's power supply would disrupt the head circuitry.

Solution:

Add a 5volt switching power supply to run the camera and the sonar range finder.



This is the switching power supply I am wiring to the camera and the sonar rangefinder.

With a clean power supply, the BasicX is now receiving readings from the sonar rangefinder.

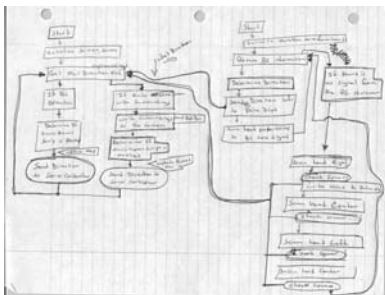
Problem:

The sonar rangefinder returns obscure values once in awhile.

Solution:

Make software condition where two incoming values must be the same for the value to be valid or used for navigating the robot.

I am now writing a software routine that would look at the robot's surroundings and relays the information to the Palm Pilot. If the robot is in RC mode, it will relay the RC information to the Palm Pilot instead. While in RC mode, the BasicX will also turn the head proportional to the RC turning signal.



Here is one of the drawings I have made while laying out the next step of the robot, which is bringing all of the individual systems together. It is hard to say exactly how the robot is going to work before I finish everything.



Finishing the head

As a finishing touch I have added headlights to the robot's head. The headlights are powered by a MOSFET transistor. The gait of the MOSFET is controlled by the BasicX micro controller.



Above is a picture of the robot's head with the new lights on it. The head lights are flashlight lamps with filaments. I decided to use lamps instead of LED's because of the amount of light the lamps can produce. The back sides of the lamps are insulated in the same liquid rubber put on the feet for traction. To the right of the head is a picture of the MOSFET transistor used to control the lamps.

All of the walking scripts are done

I have just finished all of the robot's walking scripts. The robot now has seven different walking scripts it uses to control its legs. Four of the scripts control the robots direction: forward, backward, turn right, turn left. The other three scripts are for changing the start position of the robot's legs and for initialization purposes.

As one of the last steps in building the robot, I have programmed the BasicX to use every piece of hardware it is attached to. The BasicX can now determine if the robot is in RC or autonomous mode, sense objects around it, and send appropriate information to the Palm Pilot through the SC16 servo controller. The BasicX also controls the state of the headlights on the robot.



Building video transmitter system

Up until now the video camera on the head of the robot is not being used for anything. I plan to use a UHF transmitter to transmit the camera's video signal to the operator of the robot.

Problem:

The video transmitter I ordered conflicts with the RC receiver on the robot.

Solution:

Order different video transmitter that operates at 2.5 GHz instead of at UHF frequencies.

Problem:

New video transmitter requires at least a 12volt power supply. The robot only runs on 7.2volts.

Solution:

Give the video transmitter its own power supply by using two 9volt batteries in series.

Problem:

The new transmitter is bigger then the old transmitter and will not fit on the robot.

Solution:

Cut and file off the parts not needed on the transmitter to make it smaller.



This is a picture of the camera that will send the video signal to the transmitter.



Here is a picture of the transmitter after I cut enough of the case away so it would fit on the robot.

Installing the body heat sensor

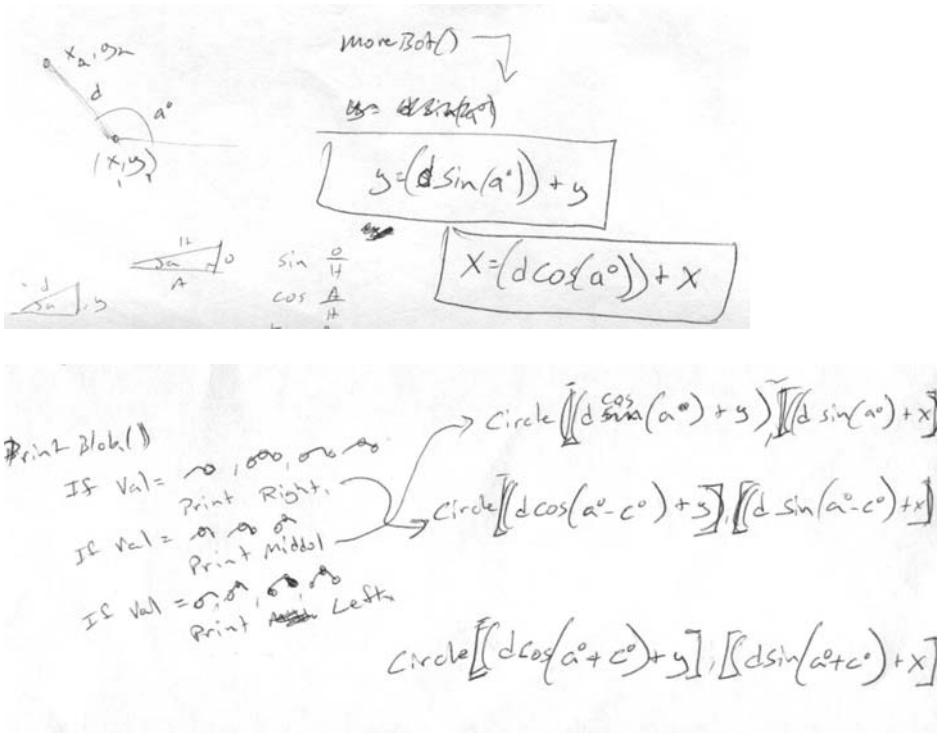
The last step of the robot is to install a body heat sensor.



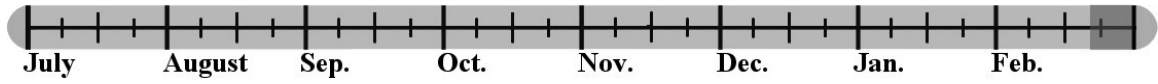
This sensor will alert the operator of the robot when there is a person in the field of view of the robot. This sensor was fairly easy to install, because the sensor uses its own power supply and does not require any use of the robot's processors.

Writing the mapping software

The last software piece left to write is the mapping section on the Palm Pilot. The mapping software uses a few math equations to figure out its relative position. By incrementing variables such as x-position, y-position and angle of orientation, the robot is able to map its position. The program uses right angle trigonometry to determine the amount of x, y incrimination to do.



These are preliminary equations I had made for determining the position of the robot. I ended up using these exact equations in the final mapping program.



The robot is done

The robot is now able to walk on its own and avoid obstacles in its way. A video signal is constantly sent to the operator of the robot. While the body heat sensor is looking for life around it, the robot is mapping its path and surroundings on the robot's onboard screen. If the robot runs into trouble or if the robot finds a person to rescue, the operator can control the robot remotely.

Here is a picture of the robot finished.

