Denise Mendez

Report on Robot (Sneaker)

Intelligence Machines Design Lab

Spring 1999

Dr Arroyo

April 18, 1999

Table of Contents

1. **Abstract**
2. **Executive Summary**
3. **Introduction**
4. **Integrated system**
5. **Mobile Platform**
6. **Actuation**
   - 6.1 **Motion Activation**
   - 6.2 **Suction Cup Activation**
   - 6.3 **Motion**
   - 6.4 **Failures**
7. **Sensors**
   - 7.1 **IR**
   - 7.2 **Bump**
   - 7.3 **Mercury**

8. **Behaviors**
   - 8.1 **Dancing**
   - 8.2 **Collision Avoidance**
   - 8.3 **Incline Walking**
   - 8.4 **Surface Clinging**

9. **Experimental Layouts and Results**
10. **Conclusion**
11. **Documentation**
12. **Appendices**
    - A. Programming
    - B. Pictures

Abstract

Sneaker walks using two suction cups as feet, it avoids obstacles, sticks both its legs on to a surface and waits until it is told to release, walks up inclines, and dances. Each behavior is controlled by a button. Sneaker on initializations puts itself into walking pose, that is with its left leg forward, and waits for a command given by a button activation. Sneaker uses bump sensors, IR transmitters and receivers, and mercury switches to implement its behaviors. Each bump sensor (there are four) pertains to a behavior. The mercury switch is to tell Sneaker to turn off if it has hit a certain degree in an incline. And the IR sensors are used for obstacle avoidance.

Sneaker uses four motors to make it move. Two of the motors activate the suction cups and the other two activate the forward motion. Two cams made out of plexi-glass enable Sneaker to lift one leg off the floor so that I would not have to worry about the friction cause by rubbing one suction cup on the floor. But, the use of cams causes sneaker to have a limited degree of freedom—he can only move at approximately 45° to 90° of each leg. This also limits the versatility of its programming, because each motion is relative to where each leg is, Sneaker has to complete a forward or backward motion before it is able receive stimuli from a sensor and complete another task.

2. Executive Summary

Sneaker is a two footed walker that, climbs up inclines and tries to walk around on flat surfaces. It has an okay success rate inhibited by a shift in the center of gravity. Sneaker uses LED IR transmitters and hacked Sharp detectors to avoid obstacles, bump sensors to determine which behavior it is in, and a mercury switch to determine whether the slope of an incline is too great. It has four behaviors: dancing, clinging to a surface and releasing it, walking up inclines, and avoiding obstacles. It contains four servos. The two 42 oz/in torque servos that actuate the suction cups using Lego's to couple the inside lever in the suction cup to the servo horn. The other two large servos (300 oz/in of torque) are used to actuate its walking motion. There are two cams; their function is to keep the free foot (the one that is not stuck on the surface) from rubbing against the surface and burdening the large servos with friction. Since the center of gravity is not towards the center of the suction cup, Sneaker has some trouble walking. This imbalance is slightly alleviated by two lead fishing weights I acquired from Patrick O'Malley. But it still does not walk perfectly. In the future more work can be done to ensure that the suction cups have better center of gravity and there is an actual platform to mount all the sensors in.

3. Introduction

Sneaker can walk on smooth flat or inclined surfaces. This paper describes the four behaviors it has, the three different sensors it will use to implement them, and a description on how everything is connected and how it all works.

4. Integrated system

Sneaker walks on the ground using suction cups. It has two suction cups as feet and behind it drags a little cart with the microprocessor (a TJ Pro board, chosen for its small size) and its batteries since there was no where on the platform to place them. The suction cup is activated using a 42 oz/in torque motor attached by the use of Lego's. One foot will clamp onto the floor while the other foot will be free to pivot. The feet are attached to each other using a bar of steel bought at Lowe's that is attached each servo horn by two small door hinges with removable pins. The 300 oz/in torque servo of the clamped foot will pivot the opposite leg 45° to 90° depending on the position that Sneaker is in at the time. The sensors are mounted on the platform wherever there was room to place them.

5. Mobile Platform

Each suction cup requires at lease 8oz of weight in order to ensure suction. The large 300 oz/in torque servos apply this necessary weight. These servos are glued to pieces of wood that were glued to the plastic in the suction cup. The wood extends over the side of the suction cup so that the smaller servos that actuate the suction cup s can be mounted. This helps the motion of the robot, as well as provide a place to put the servos, because it helps distributes the weight of the free leg so that not all of it is forced on the suction cup.

A cam moves the arm being pivoted slightly upward so that the free foot will not drag on the ground. Arfath also thought of this idea and we were thinking about how to implement it. I tried building a clay model of what I thought it should be and I built several molds which turned out to be a bad idea. I talked with an engineer who came up with a much better model of what I should do. He shaped for me two pieces of square plexi-glass and I filed them down to the contour that I wanted. But since I had to do this by hand there is inaccuracy so that the angles aren't exactly 45°. These cams are screwed to the side of the large servos.

6. Actuation

Sneaker uses four servos. Two to actuate the suction cups and two to pivot the free feet. The ones that will pivot the free feet will need anywhere between 180 oz/in torque to 280 oz/in torque.

6.1 Motion Activation

Sneaker will have the ability to turn approximately 90° to the right or left. The turning is limited to the cams which are described in the next paragraphs. Sneaker uses two large servos to move it forward an have it turn. These servos have 300 oz/in of torque. They will each be attached to either of the of the two suction cups using epoxy .

Attached to the horns on the servos are two door hinges with removable non-rising pins, which I bought at Lowe's. I also bought a steel metal bar in which I drilled two holes so that the removable pin would fit through and then be attached to the hinges. There are two pieces of rubber on either side of the metal bar on the pin. These pieces of rubber keep the bar from sliding. I used the pieces of rubber that came with the 42 oz/in servos that are supposed to be used to fix the servos onto a platform. The bar attaches the two suction cups together. Since the bar is shaped like a square, it would be hard for it to travel along the contours of the cam described. I found a pen (by Bic) which the casing of it just happened to be the right size to allow it to roll along over the contours and fit loosely on the metal bar. The metal bar also had to be trimmed down in the area above the screw that attach the servo horn to the servo, if not the screw would keep the bar from being level with the other servo and the ground. I also had to shave off the top of the servo horn where to screw went because of the same reason with the screw, but since this happened first it didn't occur to me to just shape the steel bar instead of having to file down the top of the servo horn. The servo horn itself had to rounded down because it kept hitting the cam edge. This would cause friction which would cause the servos to work harder.

Screwed into the side of the motor are two cams (which Arfath gave me the idea to build). These cams have two hills and three valleys (see figures attached). The purpose of the hill is so that once the suction cup has been stuck to the ground and the servo turns the free leg forward the steel bar will rise over the hill so that the free suction cup does not have to drag on the floor causing friction. The other motor will also turn in mid air in the opposite direction so that the “feet” are always facing forward. These cams were cut by an engineer friend and are made out of plexi-glass. I received two squares with holes for screws and I had to file down the edges so that I could get the contour I wanted (it is advantageous to use the Drummel tool to shape the contour). Since the contours were shaped using a metal file the angles won’t be perfect 45° angles, but after some testing with IC the angles turned out to be off by 1° on two of the hills and 3 to 4° on the other ones. But these can be taken care of in the programming to make sure that servo has moved the bar over the hill.

6.2 Suction Cup Activation

Each of the two suction cups is activated by a 42 oz/in torque servo. The servo is coupled to the metal arm inside of the suction cup by Lego parts. I shaved down the round metal bar inside the suction cup so that it looks like a flat screw driver. This fits into half of a gray Lego piece. The other half is attached to a black bar that looks like a big 3-D plus sign. This plus sign fits into a Lego gear which is screwed onto the servo horn and attached to the servo. Both servos are attached to the sides of the suction cups and placed in the middle of the two feet (see appendix figures). They will act as supports for the robot when it has one foot in the air.

Each servo is attached to the suction cup by the Lego parts, but it has to be fixed to something or else it won't function. Two small square pieces (the size of the servo) of the wood used for the T-Tek machine and glued together and then to the bottom of the servo are used to keep the servo level with the coupling to the arm inside the suction cup. A long piece of wood is attached (super glued) to the top of the suction cup, this wood will act as the top support for the servo. Extra wood is used to fill the gap between the servo and this wooden support. This is an added feature to the stability of the robot. The added support toward the middle of the robots feet will help it from falling on its side when on of the legs is picked up. This might even be useful for a two legged robot without the use of suction cups. Arfath helped me realize this after he showed me the wind up robot toy that is in the lab. The wind up toy walks on two feet but the feet are elongated so that as one foot is picked up, the toy is supported by the elongated foot on the floor and it will ensure that it won't fall on its side.

6.3 Motion

The heavy duty servos will be epoxied on to the top of the suction cup, but for now there will be zip ties used to fasten the servo onto the suction cup so that further testing can be done.

Approximate weight of robot: 50 oz

Necessary Torque: 5.57'*50= 278.5 oz/in

Torque of servos: 300 oz/in

Weight necessary to ensure suction: 8 oz

(1 and a half six pack of batteries)

6.4 Failures

I had built several versions of the cam by using molds made out of clay and resin. The first version was not thought out really well, and the second version was too big and bulky, but it would have had the same effect as the final version. The second cam was a large piece of resin into which were screwed two legs (they were metal shelf supports that were cut to the size necessary and bought at Lowe's). These legs would touch the floor for added stability to the robot. I was also going to use them to attach the suction cup activation servos, but they were too big and bulky. After talking to a friends father who worked at a company that deals with bullet proof materials, he cut out the clear plexi-glass pieces for the cam. I shaped them to the contour necessary using a large metal file.

I found that cold temperatures delay the hardening time of resin from a couple of hours to a couple of days. Also use well ventilated areas. The mold that I used was bought at Micheal's. It is a powder mold that you add water to. It is hard to work with because it doesn't take the shape of small complex objects very well and it is also grainy and thick when you mix it. I wasn't able to find the silicon stuff that Scott talked to me about.

When I originally screwed the hinges onto the servo horns I had placed them near where the screw was to be set. When I had to shave off part of the steel bar because of the screw in the servo horn, I realized that I had to move the hinges further away from the center of the horn. This was due to the fact that I would have eliminated the hole where the pin goes on the steel bar since it was very close to the screw in the servo horn.

My original method for coupling the suction cup activation servo involved using rubber fuel tubing that would be attach (by means of super glue) to the metal arm inside the suction cup and a screw placed on a servo horn which was screwed to another servo horn that was attached to the motor. This method did not work because some of the turning motion was lost due to the flexibility of the fuel tubing. It also left me with very sticky fingers.

7. Sensor Report

7.1 IR

Sneaker uses the standard IR LED emitters and the Sharp hacked detectors. These are used for collision avoidance. The Sharp detector and IR emitter are connected to each other and then two of them are glued to the front, one on the right and left, and one on the back. Sneaker turns to the left after he detects and obstacle the following method:

1. Detects an obstacle
2. Left leg must be forward
3. Activate left suction cup
4. Move right leg 90° without adjusting it to face forward
5. Release left suction cup, ensure right suction cup

6. Move left leg forward 22°
7. Adjust left suction cup so that it is facing forward
8. Bring left leg forward 45°

The routines for this are in the programming section in the appendix.

7.2 Bump Switch

The bump switches are used to put Sneaker in different modes: dancing, forward motion –for walking up inclines, collision avoidance, obstacle clinging. Since I am using the TJ Pro board the analog 0 port had already been multiplexed for the use of 4 bump switches.

7.3 Mercury Switch

The mercury switch was not implemented on Sneaker but it could have been used to detect the height of the incline, or whether it was climbing up the incline.

8. Behaviors

8.1 Dancing

The dancing behavior was found by mistake. When I started the programming using Preston Faiks code to control the servos, I had the robot move forward. But it moved so fast that I knew I had to put in a delay routine in order to slow down the servo(); routine. And that is how I found the dancing behavior.

8.2 Collision Avoidance

Collision avoidance is done by using the IR detectors. If the robot sees and obstacle within a foot (a reading greater that 110) it will move away from that object: move left if right sensor reads high, right if left sensor reads high, and back if front sensor reads too high.

8.3 Incline walking

To get the robot to go up an incline I just have it continuously move forward. Although due to inaccuracy forward is not always perfectly straight.

8.4 Object pick up

Sneaker clings to an object when a button is pushed and releases the object when the button is pushed again. I thought that this would make an interesting behavior and Jose Garcia gave me the idea as an application for a worker that need to pick up glass, takes the robot pushes it on the surface, presses the button, and moves the glass.

9. Experimental Layouts and Results

In order to figure out the exact positions the servos needed to be in, I used a program that Preston Faiks wrote. The program got input from the keyboard and let me interact with the servo positions so that I would be able to find the range of motion that I would need in the program. Middle position for the large servos was 0° (relative to the middle position on the cams. Because of the inaccuracy of the cams on each leg middle position +45° and -45° were different on each servo. On one servo 0° corresponded to a pulse width of 3300 and on the other it was 3150; +45° was 2200 on one and 4300 on the other; -45° was 2100 on one and 4200 on the other. This is why I was limited as to the versatility of the program.

10. Conclusion

This semester I learned a lot about how to integrate some of the knowledge that I acquired at the university, but since my project was both mechanical that electrical the mechanical aspect got most of my attention. The robot is not perfect but we are where we are because we stand on the shoulders of giants. I couldn't have done most of this without the TA's help or the students help. And my robot has plenty of limitations. It can only move at certain angles and the programming is long because of the complexity of how the legs move with relation to each other.

I did not get done as much as I wished, but given more time I am sure I would have gotten Sneaker to do more. I hadn't realized what an important role the center of gravity would play to ensure that the suction cups worked. I found that Sneaker worked better walking up an incline than on flat ground and Arfath helped me realize that is was because the center of gravity was off, it was towards the front as opposed to the middle. This was good on a incline, but bad on horizontal surfaces.

This semester has been a sobering experience for a graduating senior, maybe that is not a good thing, but sometimes we just do not want to grow up. I entered the semester with dreams of walking on walls and flying without wings, only to realize that falling is very painful from such a height, and it teaches us to make the walls smaller so that they are easier to climb. I hope that if given a challenge in the future I will snatch it up, and not shy away because of the greatness of the distance.

10.1 Future work

1. Build own platform to help center of gravity including suction cup housing and allow room for onboard processor and batteries.
2. Work on a mechanism that helps counter act center of gravity when on an incline, this would increase the maximum height the robot can walk on an incline, and help it climb down an incline.
3. Position the large servos in that same direction so as in increase the versatility of the programming.
4. Work on integrating the mercury switch.
5. Try to solve why left suction cup activates better than the right one

11. Documentation

12. Appendices

A. Prgram

```
#include "vectors.c"
#include<analog.h>
//#include "serial2.c"
```

```
//#include "servo.c"
//	This program starts with the left foot
//    and moves forward and then skips over the middle
void waste(int);
void move(int, int, int);

void forward_l(void);
void forward_r(void);

void back_l(void);
void back_r(void);
void back_turn(void);

void right_front(void);
void forward_long_l(void);
void forward_long_r(void);

void back_long_l(void);
void back_long_r(void);

void turn_l(void);
void turn_r(void);

void middle(void);
#define IR	*(unsigned char *)(0x7000)

//*************globals************************//
	int ana_0,rf_ana2,bm_ana3;
	int rrs_ana7;
	int lf_ana5,lls_ana6;  //These are analog ports

int main(void)
{
	int loop,q,r,change;		//These are loop vars

	IR=0xff;
	init_serial();
	write("init\n\r");
	init_servos();
	init_analog();

	INTR_ON();
	write("Servo power\n\r");
	loop = 0;
	change = 0;
```

```
        power(3,1);
            power(4,1);
            power(5,1);
            power(6,1);
   write("Servo on\n\r");

            middle();
            waste(6);
            waste(6);
            middle();
            forward_l();

 while(1)
    {
//collision avoidance if button is pushed
            ana_0 = analog(0);
            if ((ana_0 > 63) && (ana_0 < 72))
            { while ((ana_0 <= 15) || (ana_0 >= 29))
               {
//Obstacle in  the right front
             rf_ana2=analog(2);
                if ( rf_ana2>105)
                        {back_turn();
                        forward_l();
                        }
            rrs_ana7=analog(7);
//obstacle to the right
                if ( rrs_ana7>110)
                        {turn_l();
                        forward_l();
                        }

//Obstacle to the left front
            lf_ana5=analog(5);
                if (lf_ana5>110)
                        {
                        back_turn();
                        forward_l();
                         }
            lls_ana6=analog(6);
                if (lls_ana6>110)
                        {right_front();
                        turn_r();
                        forward_l();
                        }
```

```
                ana_0 = analog(0);
              }       //end of next button condition
          }                //end of collison avoidance
//Dance if button pushed
                  ana_0 = analog(0);
             if ((ana_0 > 123) && (ana_0 < 132))
                  {
                   q=0;
                    while (q<9)
                         {        write("inSIDEWHILE\n\r");
                         servo(5,2200);
                         waste(14);
                         servo(6,4300);
                         waste(14);
                         servo(5,4200);
                         waste(14);
                         servo(6,2100);
                         waste(14);
                         q++;
                         }
                         waste(4);
                         middle();
                         waste(9);
                         forward_l();
                  }
          ana_0 = analog(0);
          if ((ana_0 > 40) && (ana_0 < 51))
            { if (change == 0)
                  {middle();
                   move(4,1200,4500);
                   move(3,1200,4500);
                   change = 1;
                  }
              else
                  {move(4,4500,1250);
                   move(3,4500,1200);
                   forward_l();
                   change = 0;
                  }
            }          //end of behavior 3
          ana_0 = analog(0);
          if ((ana_0 > 18) && (ana_0 < 25))
           { while ((ana_0 <= 64) || (ana_0 >= 72))

              {forward_long_l();
```

```
                ana_0 = analog(0);
                 }
            }        //end of behavior 4 (incline)
} //end of while
} //end of main
void middle(void)
 {
       servo(3,1200);
          servo(4,1200);
          servo(5,3150);
          servo(6,3300);
 }
void forward_l(void)
{
//From mid position

          move(4,1200,4500);
          move(6,3300,3700);
          move(5,3150,4100);
          move(6,3700,4400);
          move(6,4400,4300);
          move(4,4500,1200);

}
void forward_r(void)
{
//From mid position

          move(3,1200,4500);
          move(5,3150,2500);
          move(6,3300,2000);
          move(5,2500,1900);
          move(5,1900,2000);
          move(3,4500,1200);

}
void back_l(void)
{
// back left leg to forward position

          move(4,1200,4500);
          move(6,3300,3700);
          move(5,3150,4100);
          move(6,3700,4300);
          move(4,4500,1200);
```

```
}
void back_r(void)
{
// From right leg forward from mid to start

        move(3,1200,4500);
        move(5,3150,3700);
        move(6,3300,4300);
        move(5,3700,4100);
        move(3,4500,1200);
}

void right_front(void)
{
        move(3,1200,4500);
        move(5,4100,2500);
        move(6,4300,2000);
        move(5,2500,2000);
        move(3,4500,1200);
}
void forward_long_l(void)
{
// This routine assumes that the legs are already
//spaced apart left leg forward.  Executing this routine
//leaves sneaker with its left foot forward

        move(3,1200,4500);
        move(5,4100,2500);
        move(6,4300,2000);
        move(5,2500,2000);
        move(3,4500,1200);

//right leg forward now moving left leg

        move(4,1200,4500);
        move(6,2000,3700);
        move(5,2000,4100);
        move(6,3700,4300);
        move(4,4500,1200);
}
void forward_long_r(void)
{
// This routine assumes that the legs are already
//spaced apart right leg forward.  Executing this routine
//leaves sneaker with its right foot forward
```

```
//right leg forward now moving left leg

        move(4,1200,4500);
        move(6,2000,3700);
        move(5,2000,4100);
        move(6,3700,4300);
        move(4,4500,1200);

        move(3,1200,4500);
        move(5,4100,2500);
        move(6,4300,2000);
        move(5,2500,2000);
        move(3,4500,1200);


}
void back_long_l(void)
{
        move(4,1200,4500);
        move(6,4300,2500);
        move(5,4100,2000);
        move(6,2500,2000);
        move(4,4500,1200);

//left leg back now moving right leg back so that
//left leg is now facing front

        move(3,1200,4500);
        move(5,2000,3700);
        move(6,2000,4300);
        move(5,3700,4200);
        move(3,4500,1200);
}
void back_long_r(void)
{
//left leg back now moving right leg back so that
//left leg is now facing front

        move(3,1200,4500);
        move(5,2000,3700);
        move(6,2000,4300);
        move(5,3700,4200);
        move(3,4500,1200);

        move(4,1200,4500);
```

```
        move(6,4200,2500);
        move(5,4100,2000);
        move(6,2500,2000);
        move(4,4500,1200);


}
void back_turn(void)
{
        move(4,1200,4500);
        move(6,4300,2500);
        move(6,2500,2000);
        move(4,4500,1200);
        move(3,1200,4500);
        move(5,4100,3700);
        move(6,2000,3300);
        move(5,3700,3150);
        move(3,4500,1200);
}
void turn_l(void)
{
//This routine assumes that the left leg is again
//forward turns the robot perpendicular and places it
//in mid position.
//Right or left wil be relative to forward or backward.

        move(3,1200,4500);
        move(5,4100,2500);
        move(5,2500,2000);
        move(3,4500,1200);
        move(4,1200,4500);
        move(6,4300,3700);
        move(5,2000,3150);
        move(6,3700,3300);
        move(4,4500,1200);

}
void turn_r(void)
{
//This routine assumes that the right leg is
//forward turns the robot perpendicular and places it
//in mid position.
//Right or left wil be relative to forward or backward.
//These turn routines also assumes obsacle clearance in
//front of the robot.
```

```
            move(4,1200,4500);
            move(6,2000,3500);
            move(6,3500,4300);
            move(4,4500,1200);
            move(3,1200,4500);
            move(5,2000,2500);
            move(6,4300,3300);
            move(5,2500,3150);
            move(3,4500,1200);
}


void waste(int a)
{
    int n,b;
            for (n=1 ; n<500 ; n++)
            { for (b=1; b<a; b++);
            }
}

void move(int servo_num, int old, int new)
{           int k,temp;
            temp = 3;
            if ((ana_0 > 123) && (ana_0 < 132))
                        {temp = 3;}
            if (servo_num == 3)
            {temp = 1;}
            if (servo_num == 4)
            {temp = 1;}


            if (old > new)
            {   for (k = old; k >= new; k=k-50)
                        {
                        servo(servo_num, k);
                          waste(temp);
                        }
            }
            else
            {  for (k = old; k <= new; k=k+50)
                        {
                        servo(servo_num, k);
                         waste(temp);
                        }
```

```
    }
}
```