Jerry the Mousebot

Geoff Hahn

December 9, 2002

University of Florida

Department of Electrical and Computer Engineering

EEL5666

Intelligent Machines Design Laboratory

Table of Contents

Abstract

My robot is modeled after a mouse. Its main purpose is to play with a cat. It does this by trying to taunt and then evade the cat. After Jerry is played with for a certain period of time, he goes into a sleep mode to wait until the cat wants to play some more.

Executive Summary

Jerry works as planned initially except for a few modifications. For the taunting behavior, his tail does not spin. Instead, he moves rapidly left and right to shake his tail at the cat and taunt it. When his tail is tugged or the bump sensors along the top are touched, he will begin to evade the cat.

Jerry uses a pyroelectric sensor, IR detectors and bump whiskers to avoid the cat and other obstacles. The combination of these three sets of sensors makes his movements seem erratic and random, so that the cat will remain interested in playing with Jerry.

After a certain amount of time, Jerry will go back to sleep, to help conserve battery power. Upon waking, he will begin again to taunt the cat.

Introduction

Cats are naturally curious and playful animals. Domestication of them does not rid them of these traits. Many people do not have time to play with their cats all the time, due to work, school, and all the normal run-around of everyday life. This leads a curious cat to pursue other things to play with, which usually results in torn furniture and broken lamps. This is where Jerry will be useful. The robot will give the cat something to play with so that its hunting urges can be directed in a non-destructive direction.

Integrated System

Jerry is controlled by an Atmel ATMEGA163 microcontroller board from Progressive Resources. There will be 2 IR detectors for obstacle avoidance, some bump switches on the top to sense when the cat is playing, some whiskers in front to determine if he bumps into something, and a pyroelectric sensor to detect when the cat is nearby. There will also be a tug sensor to indicate when the tail is being pulled. The actuation system will consist of 2 servos that will power the wheels. An 8 pack of NiCd batteries will power the whole thing.

Mobile Platform

The platform is a rectangular shaped box with a wheel on each side. There is an opening in the back for the tail. The "nose" is pyramid-shaped with the top cut off so that there is an opening for the pyroelectric sensor that is mounted inside. On either side of the nose

are mounted whiskers for sensing if Jerry is bumping into an object. At the top of the platform are mounted two ears, each with a Sharp GP2D12 IR sensor mounted on it. The top is a door that opens upward so that the electronics can be accessed. It is closed when the robot is running so that all the components are safe from the cat. Mounted on top are 3 bump switches that sense if the cat is touching them. There is a latch mounted near the front of the door so that it can be locked shut from the cat. At the back on the bottom of Jerry is mounted a plastic low-friction furniture mover so that he will slide easily along the floor. Inside there is a compartment for the battery pack and the microcontroller. The microcontroller is mounted inside via Velcro glued on the bottom of it.

Actuation

The two wheels are controlled by hacked MPI MX-400 servos from Acroname. Black injection molded plastic wheels from MarkIII are mounted onto each of the servos. The wheels are threaded to fit onto the servo shaft and a screw is used to hold them on. The servos are run by a PWM, which is a feature that is integrated into the Atmel board.

Sensors

Bump Sensors

Jerry has bump sensors mounted along the top of the robot to sense when the cat is playing. They are push reset switches from Radio Shack with a piece of wood glued to the top of each one so that there is a wider area that it will detect where the cat is hitting.

Each one of the switches is connected to a voltage divider circuit as shown below in Figure 1.

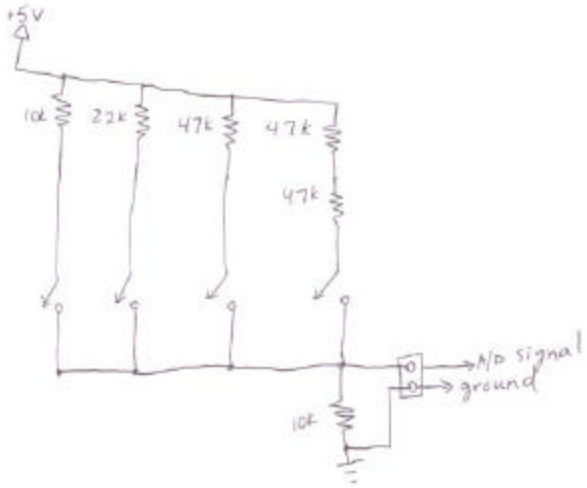

Figure 1

Whiskers

Mounted on the each side of Jerry's nose is a whisker. I purchased a bumper switch kit from Lynxmotion (part # BMP-01). This consisted of two switches, two pieces of PVC tubing, and two rounded rubber end pieces to fit over the hollow pieces of PVC. The switch is pressed when the whisker bumps into an object head on. Both of the whiskers are connected to the voltage divider circuit in Figure 2 below so that Jerry knows which whisker is being touched. One of the problems that I've had with the whiskers is that if Jerry backs up too close to an object, the whiskers will sometimes get caught and pulled on. These bump sensors were designed to be mounted with both pieces of the PVC tubing curving towards the center. I did not mount them that way, however, because I

wanted them to look more like mouse whiskers and it would also have been more difficult to mount it that way.

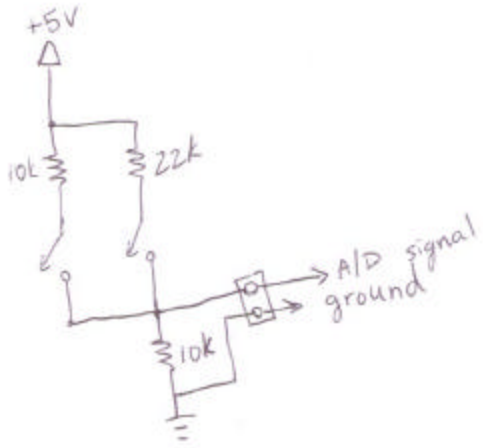

Figure 2

IR sensors

There are IR sensors for obstacle avoidance mounted on the ears on top of Jerry. They are both pointed slightly outward from each other. The IR sensors that I used are Sharp GP2D12 sensors that send an analog voltage to the A/D converter that is proportional to the distance that an object is from the sensor. The greater the value, the closer an object is. The range of values versus the distance (in inches) of an object away from the sensor is shown in Figure 3.

Pyroelectric sensor

I mounted an Eltec 442-3 pyroelectric sensor in Jerry's nose so that he can "smell" where the cat is. It gives an analog voltage to the A/D converter based on the movement of heat

sources. Since people and animals are heat sources, it can detect them. It will have a value of greater than 134 when the cat is moving left and a value less than 122 when the cat is moving right. The value will be about 128 (which corresponds to about 2.5V) when there is no heat source in view.

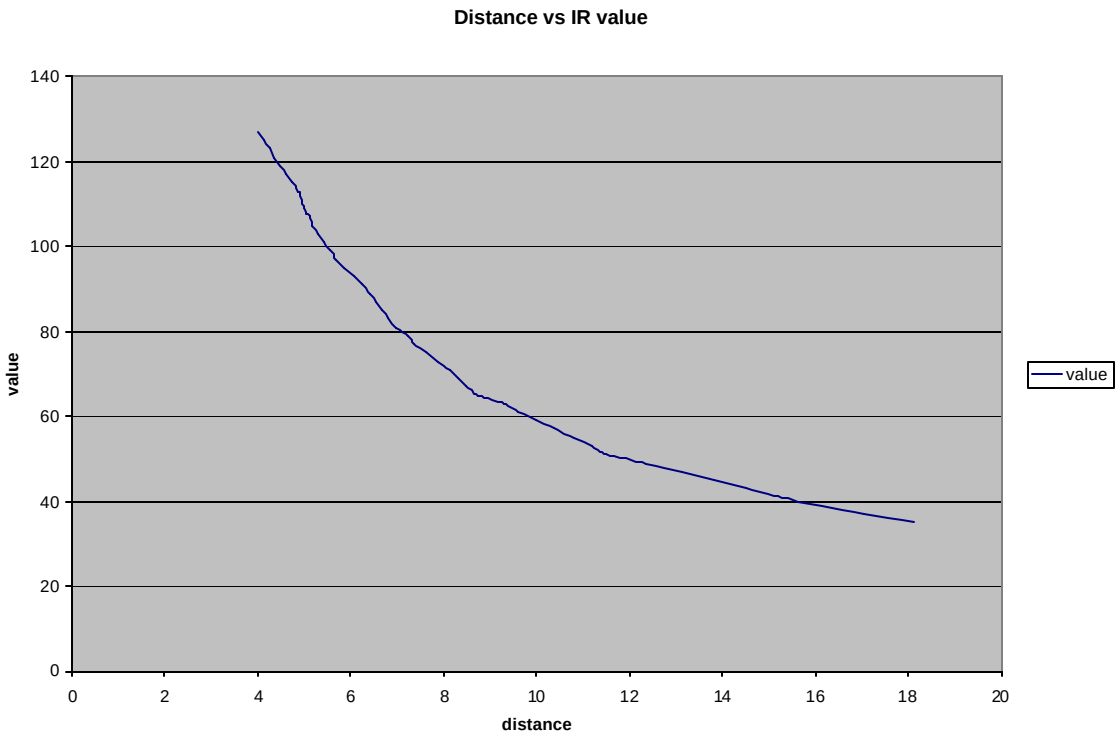

Figure 3

Flex Sensor

My special sensor is a flex sensor that is mounted to the tail. The tail is made from a toy rubber snake that is naturally bent. I taped a flex sensor (Jameco part # 150551) to the tail. The flex sensor is a variable resistor that changes its resistance based on how much it is bent. It has an optimal resistance of about 10 k-ohms and a maximum resistance of

about 30-40 k-ohms. The tail is connected to a voltage divider circuit. When the tail is tugged on by the cat, the flex sensor is straightened and the resistance changes, which changes the value measured by the A/D converter.

Behaviors

The first behavior Jerry will have is playing with the cat. He will move away from the direction that the pyroelectric sensor determines the cat is in and at the same time avoid other obstacles with his IR sensors and whiskers.

After a certain amount of time of the playing behavior, he will enter a sleep behavior. Upon waking, Jerry will taunt the cat and if the cat either pulls on his tail or hits one of the bump switches on top, he will enter into his playing behavior. In the taunting behavior, Jerry moves rapidly left and right to shake his tail at the cat.

Conclusion

Jerry does everything that he was created to do. I had a few problems while building him, however. I had a lot of trouble designing the platform in Autocad before I knew exactly what I was doing. I made a few mistakes in that area, the biggest of which was a put the hole for the serial port to plug into too close to one of the wheels, so I had to take off that wheel whenever I was downloading the program to it, which was quite often. I also made Jerry too big because it was the only way I could fit everything inside.

I'm not really sure how to make it smaller, but there must be a better way. It's the same size as a cat, which would probably scare a lot of them. With better servos, I could have made Jerry faster, so that a cat would be even more interested in playing with him. I originally wanted to make the tail spin by mounting it onto a servo. Unfortunately, I hacked that servo and didn't think about the cord that is attached to the tail. If it were to spin, the cord would get wrapped up and probably damage something in the process.

Appendix

```
/*
Title: bigOne.c
Purpose: alternate between taunting the cat, running from the cat,
and going to sleep
```

```
A/D interface and servo control modified from
code originally written by Kristen Allen
*/

#include <io.h>
#include <math.h>
#include "bigOne2.h"

int twoFeetLeft, twoFeetRight, oneFootRight, oneFootLeft;
int oldLeftValue = 102;
int newLeftValue;
int oldRightValue = 102;
int newRightValue;
int wait, count, start, stop, x, z;


unsigned short servoSpeed; //variable that holds the servo speed


void motorInit(void) {

        outp(servoStop, OCR1AL); //set PWM to 10% duty cycle on channel A
        outp(servoStop, OCR1BL); //set PWM to 10% duty cycle on channel B
        sbi(TCCR1A, COM1A1);
        sbi(TCCR1A, COM1B1);
        sbi(TCCR1A, PWM10);
        sbi(TCCR1A, PWM11);

        sbi(TCCR1B, CS11);
        sbi(TCCR1B, CS10);
        sbi(DDRD, PD4);
        sbi(DDRD, PD5);

}

void calibrate(void) {

        ADCset(bump); //set ad system to bump

        start = runADC(bump); //get ad reading

        int p, wait3;

        while(start < 22) { //wait till button is pressed

                wait3 = 0;

                start = runADC(bump);

                for(p = 0; p < 100; p++) {

                        wait3 = wait3 + 1;

                }//end for

        }//end while
```

```
ADCset(leftIR);
oneFootLeft = runADC(leftIR);
ADCset(rightIR);
oneFootRight = runADC(rightIR);
sbi(PORTB,PC6); //turn on IR led to indicate that it has that reading
while(start >= 20) { //wait until button is released
        wait3 = 0;
        ADCset(bump);
        start = runADC(bump);
        for(p = 0; p < 100; p++) {
                wait3 = wait3 + 1;
        }
}
cbi(PORTB, PC6); //turn off IR led
start = 0;
while(start < 22) { //wait till button is pressed again
        wait3 = 0;
        ADCset(bump);
        start = runADC(bump);
        for(p = 0; p < 100; p++) {
                wait3 = wait3 + 1;
        }
}
ADCset(leftIR);
twoFeetLeft = runADC(leftIR);
ADCset(rightIR);
twoFeetRight = runADC(rightIR);
sbi(PORTB, PC6); //turn on IR led to indicate that it has that reading
```

```
        while(start >= 20) { //wait until button is released

                wait3 = 0;

                ADCset(bump);

                start = runADC(bump);

                for(p = 0; p < 100; p++) {

                        wait3 = wait3 + 1;

                } //end for

        } //end while

        cbi(PORTB, PC6);

        while(start < 22) { //wait till button is pressed final time

                wait3 = 0;

                ADCset(bump);

                start = runADC(bump);

                for(p = 0; p < 100; p++) {

                        wait3 = wait3 + 1;

                }

        }

} //end calibrate method

void init_ADC(void) {

        sbi(ADCSR, ADEN); //enable ADC
        sbi(ADMUX, ADLAR); //set ADC to left-adjusted to make ADC 8-bit
        //set the clock divider by 16
        sbi(ADCSR, ADPS2); //ADPS2 set to 1
        cbi(ADCSR, ADPS1); //ADPS1 set to 0
        cbi(ADCSR, ADPS0); //ADPS0 set to 0
        sbi(ADMUX, REFS0); //set to 2.56V reference voltage
}

void ADCset(int port) {

        if(port == 0) {

                cbi(ADMUX, MUX2); //set to channel 0
                cbi(ADMUX, MUX1);
                cbi(ADMUX, MUX0);
```

```
	}else if(port == 1) {

		cbi(ADMUX, MUX2); //set to channel 1
		cbi(ADMUX, MUX1);
		sbi(ADMUX, MUX0);

	}else if(port == 2) {

		cbi(ADMUX, MUX2); //set to channel 2
		sbi(ADMUX, MUX1);
		cbi(ADMUX, MUX0);

	}else if(port == 3) {

		cbi(ADMUX, MUX2); //set to channel 3
		sbi(ADMUX, MUX1);
		sbi(ADMUX, MUX0);

	}else if(port == 4) {

		sbi(ADMUX, MUX2); //set to channel 4
		cbi(ADMUX, MUX1);
		cbi(ADMUX, MUX0);

	}else if(port == 5) {

		sbi(ADMUX, MUX2); //set to channel 5
		cbi(ADMUX, MUX1);
		sbi(ADMUX, MUX0);

	}else if(port == 6) {

		sbi(ADMUX, MUX2); //set to channel 6
		sbi(ADMUX, MUX1);
		cbi(ADMUX, MUX0);

	}else if(port == 7) {

		sbi(ADMUX, MUX2); //set to channel 7
		sbi(ADMUX, MUX1);
		sbi(ADMUX, MUX0);

	}

}

//runs the AD conversion,
//returning the ADC value
int runADC(int portNum) {

	int i, value, n;

	sbi(ADCSR, ADSC); //start the conversion

	for (n=0; n<4000; n++) { //wait for the conversion to be complete
		i=i+1;
```

```
	}

	value = inp(ADCH); //stores value measured by the ADC into value

	return value;

}

void servoStart(int motNum, int speed) {

	if(speed > fullRight) {
		speed = 100;			//set speed to 100 if greater than 100

	}else if (speed < fullLeft) {
		speed = -100;			//set speed to -100 if less than -100

	}

	if (motNum == 0) {

		speed = -speed;

		if (speed == 0) {
			servoSpeed = servoStop;

		}else if(speed >= fullRight) {
			servoSpeed = servoRight;

		}else if(speed <= fullLeft) {
			servoSpeed = servoLeft;

		}else {
			servoSpeed = servoStop - speed;

		}

		newLeftValue = servoSpeed;

		if(oldLeftValue > newLeftValue) { //if the speed is being decreased

			while(servoSpeed > newLeftValue) {

				wait = 0;

				servoSpeed = ( (K * oldLeftValue) + newLeftValue) / (K + 1);

				outp(servoSpeed, OCR1AL);

				for(count = 0; count < 100; count++) {

					wait = wait + 1;

				}//end for

			}//end while
```

```
                servoSpeed = newLeftValue;
                outp(servoSpeed, OCR1AL);
        }else if(oldLeftValue < newLeftValue) { //if speed is being increased
                while(servoSpeed < newLeftValue) {
                        wait = 0;
                        servoSpeed = ( (K * oldLeftValue) + newLeftValue) / (K + 1);
                        outp(servoSpeed, OCR1AL);
                        for(count = 0; count < 100; count++) {
                                wait = wait + 1;
                        } //end for
                } //end while
                servoSpeed = newLeftValue;
                outp(servoSpeed, OCR1AL);
        }else if(oldLeftValue == newLeftValue) { //if speed is staying the same
                servoSpeed = newLeftValue;
                outp(servoSpeed, OCR1AL);
        }//end if-else
        oldLeftValue = servoSpeed;
} else if(motNum == 1) {//if right motor
        if(speed == 0) {
                servoSpeed = servoStop;
        }else if(speed >= fullRight) {
                servoSpeed = servoRight;
        }else if(speed <= fullLeft) {
                servoSpeed = servoLeft;
        }else {
                servoSpeed = servoStop - speed - 5;
        }
        newRightValue = servoSpeed;
        if(oldRightValue > newRightValue) { //if the speed is being increased
```

```
                while(servoSpeed > newRightValue) {
                        wait = 0;
                        servoSpeed = ( (K * oldRightValue) + newRightValue) / (K + 1);
                        outp(servoSpeed, OCR1BL);
                        for(count = 0; count < 100; count++) {
                                wait = wait + 1;
                        } //end for loop
                } //end while loop
                servoSpeed = newRightValue;
                outp(servoSpeed, OCR1BL);
        }else if(oldRightVa lue < newRightValue) { //speed is being decreased (full right is 1)
                while(servoSpeed < newRightValue) {
                        wait = 0;
                        servoSpeed = ( (K * oldRightValue) + newRightValue) / (K + 1);
                        outp(servoSpeed, OCR1BL);
                        for(count = 0; count < 100; count++) {
                                wait = wait + 1;
                        } //end for
                } //end while
                servoSpeed = newRightValue;
                outp(servoSpeed, OCR1BL);

        }else if(oldRightValue == newRightValue) {
                servoSpeed = newRightValue;
                outp(servoSpeed, OCR1BL);
        } //end if-else
        newRightValue = servoSpeed;
} //end else if
return;
```

```
}//end servoStart
//turns around to the right
void turnAroundRight(void) {

	motorSpeed(0, 0);

	motorSpeed(100, 0);

	for(z = 1; z < 25000; z++) {

		wait = 0;

		for(x = 1; x < 10; x++) {

			wait = wait + 1;

		}

	}

	motorSpeed(0, 0);

}

//turns around to the left
void turnAroundLeft(void) {

	motorSpeed(0, 0);

	motorSpeed(0, 100);

	for(z = 1; z < 25000; z++) {

		wait = 0;

		for(x = 1; x < 50; x++) {

			wait = wait + 1;

		}

	}

	motorSpeed(0, 0);

} //end turnAroundLeft method


//turns right
void turnRight(void) {

	int wait = 0;
	int z, x;
```

```
        motorSpeed(0, 0);

        motorSpeed(100, 0);

        for(z = 1; z < 25000; z++) {

                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }

        }

        motorSpeed(0, 0);

} //end turnRight method

//turns right
void turnLeft(void) {

        int wait = 0;
        int z, x;


        motorSpeed(0, 0);

        motorSpeed(0, 100);

        for(z = 1; z < 25000; z++) {

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }

        }

        motorSpeed(0, 0);

} //end turnRight method

void motorSpeed(int leftSpeed, int rightSpeed) {

        servoStart(leftMotor, leftSpeed);
        servoStart(rightMotor, rightSpeed);

}

//wiggles his  tail to taunt the cat
void wiggleTail(void) {

        for(x = 0; x < 5; x++) {
```

```
                count = 0;
                while(count < 5000) {
                        motorSpeed(0, 20);
                        count++;

                }//end while
                motorSpeed(0, 0);
                count = 0;
                while(count < 5000) {
                        motorSpeed(20, 0);
                        count++;
                }//end while
        }//end for
        motorSpeed(0, 0);
}//end wiggleTail

//taunts cat to initiate play
void taunt(void) {
        int bumpValue = 0, tailValue = 0;
        cbi(PORTC, PC3);
        for(;;) {
                wiggleTail();
                motorSpeed(15, 15);
                for(z = 0; z < 300; z ++) {
                        ADCset(tail);
                        tailValue = runADC(tail);
                        if(tailValue > 140) {
                                sbi(PORTB, PC1); //turn on tailLED
                                wait = 0;
                                for(x = 0; x < 10000; x++) { //delay so that LED is visible
```

```
                                        wait = wait + 1;
                                }
                                goto End;
                        }//end if
                        ADCset(bump);
                        bumpValue = runADC(bump);
                        if(bumpValue > 40) {
                                sbi(PORTB, PC0); //turn on bumpLED
                                wait = 0;
                                for(x = 0; x < 10000; x++) {//delay so that LED is visible
                                        wait = wait + 1;
                                }
                                goto End;
                        }//end if
                }//end for
        } //end for loop
End: cbi(PORTB, PC0); //turn off tailLED
        cbi(PORTB, PC1); //turn off bumpLED
        motorSpeed(0, 0);
}//end taunt

//run from the cat
void run(void) {
        int whiskerValue, pyroValue, rightValue, leftValue, tailValue;
        for(x = 0;x < 200; x++) {
//                      turnRight();
                        motorSpeed(100, 100);
                        ADCset(whiskers);
                        whiskerValue = runADC(whiskers); //stores value measured by the ADC into leftValue
                        ADCset(pyro);
```

```
pyroValue = runADC(pyro);

ADCset(leftIR);

leftValue = runADC(leftIR);

ADCset(rightIR);

rightValue = runADC(rightIR);

if(whiskerValue > 64 && whiskerValue <= 96) { //if right whisker is bumped

        motorSpeed(0, 0); //stop
        motorSpeed(-100, -100); //back up from obstacle

        for(z = 1; z < 25000; z++) {

                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }//end for

        }//end for

        turnLeft();

} else if(whiskerValue > 128 && whiskerValue <= 154) {

        motorSpeed(0, 0); //stop
        motorSpeed(-100, -100); //back up from obstacle

        for(z = 1; z < 25000; z++) {

                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }//end for

        }//end for

        turnRight();

} else if(whiskerValue > 154 && whiskerValue <= 162) {

        motorSpeed(0, 0); //stop
        motorSpeed(-100, -100); //back up from obstacle

        for(z = 1; z < 25000; z++) {
```

```
                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }//end for

        }//end for

        turnAroundLeft();

} else if(pyroValue > 150 && pyroValue <= 255) { //if cat is moving left (on the right)

        sbi(PORTB, PC7); //turn on pyroLED

        motorSpeed(0, 0); //stop
        motorSpeed(-100, -100); //back up from obstacle

        for(z = 1; z < 25000; z++) {

                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }//end for

        }//end for

        turnAroundLeft();

        cbi(PORTB, PC7); //turn off pyroLED

} else if(pyroValue > 0 && pyroValue <= 105) { //if cat is moving right(on the left)

        sbi(PORTB, PC7); //turn on pyroLED

        motorSpeed(0, 0); //stop
        motorSpeed(-100, -100); //back up from obstacle

        for(z = 1; z < 25000; z++) {

                wait = 0;

                for(x = 1; x < 40; x++) {

                        wait = wait + 1;

                }//end for

        }//end for

        turnAroundRight();
```

```
                cbi(PORTB, PC7); //turn off pyroLED

            } else if(leftValue >= twoFeetLeft && rightValue >= twoFeetRight) { //if an object is too
close to both IRs

                sbi(PORTB, PC6); //turn on IR LED

                turnAroundRight();

                cbi(PORTB, PC6); //turn off IR LED

            } else if(leftValue >= twoFeetLeft && rightValue < twoFeetRight) { //if an object is
close to leftIR

                sbi(PORTB, PC6); //turn on IR LED

                turnRight();

                cbi(PORTB, PC6); //turn off IR LED

            }else if(rightValue >= twoFeetRight && leftValue < twoFeetLeft) {//if an object is close
to rightIR

                sbi(PORTB, PC6); //turn on IR LED

                turnLeft();

                cbi(PORTB, PC6); //turn off IR LED

            } //end if-else

        } //end for loop

}//end run

void sleep(void) {

        cbi(PORTC,PC4);

        motorSpeed(0, 0);

        cbi(TCCR1A, COM1A1);
        cbi(TCCR1A, COM1B1);

        for(z = 1; z < 25000; z++) {

            wait = 0;

            for(x = 1; x < 300; x++) {

                wait = wait + 1;

            }//end for

        }//end for
```

```
        sbi(TCCR1A, COM1A1);
        sbi(TCCR1A, COM1B1);

}//end sleep



int main(void) {

//      int leftValue = 0, rightValue = 0, whiskerValue = 0, pyroValue = 0, tailValue = 0;

        init_ADC();  //initialize the ADC

        outp(0xff,DDRC);        // use all pins on PortC for output

        outp(0xff,PORTC);

        outp(0xff,DDRB);        // use all pins on PortB for output

        outp(0x00,PORTB);                       //turn off all the LEDs

        calibrate(); //calibrate the IRs

        motorInit();

        for(;;) {

                taunt();

                run();

                for(z = 1; z < 4000; z ++) {

                        sleep();

                } //end for

        }//end for

        return 0;

}
```