

Date: 4/24/12
Student Name: Jonathan Krieger
E-mail: kriegerj@ufl.edu
TAs : Ryan Stevens
Tim Martin
Josh Weaver

Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

University of Florida
Department of Electrical and Computer Engineering
EEL 4665/5666
Intelligent Machines Design Laboratory

Final Report

Table of Contents

Abstract	3
Executive Summary	3
Introduction	3
Integrated System	4
Mobile Platform	5
Actuation	7
Sensors	8
Behaviors	11
Experimental Layout and Results	12
Conclusion	15
Documentation	15
Appendix A – Circuits Used	16
Appendix B - Source Code	19

1. Abstract

The purpose of this report is to explain the development that went into building an autonomous robot. This robot's main purpose is two-fold. First, it will play with a cat using a laser pointer and moving the laser pointer across the ground. Second, the robot will punish the cat if it is caught scratching. It will accomplish these tasks through a combination of infrared thermometer, sonar sensors, and a microphone. It will also be capable of collision avoidance to prevent damage to the robot.

2. Executive Summary

Zobot is a fully autonomous robot that interacts with my cat and was designed in my Intelligent Machine Design Lab at the University of Florida. During the day Zobot will play with my cat, Zoey, by having her chase a red laser on the ground.

At night, everything changes as Zobot will habituate Zoey against scratching at the door. The robot will activate upon the sound of scratching, and then it will locate the source using a non-contact thermometer. It then drives over to her and spray her with compressed air.

Zobot uses an Xmega64a1 processor for controlling the actuators and decision-making. For the infrared temperature sensor, the robot uses an Arduino Mega 2560 that communicates with the Xmega.

Overall, Zoey enjoys playing with Zobot and the Guard Mode has successfully passed all its tests. I have learned a tremendous amount this semester and I am especially thankful for all that the professors and teaching assistants have done for me.

3. Introduction

The cat in Figure 3.1 is Zoey. She is a 3 year old Lynx-point Siamese that my wife and I adopted. Like most Siamese cats, her mischievousness is only matched by her curiosity.

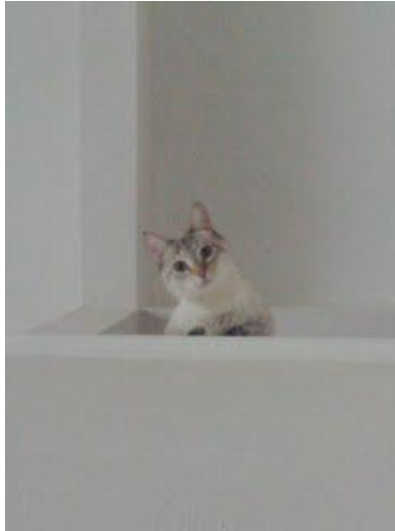


Figure 3.1. A picture of Zoey on the balcony.

As you can see, she likes to flirt with danger by jumping up on the balcony ledge in my loft. To prevent this, we keep the door to the loft closed. Naturally, she hates any door being closed. So now she tries to tunnel her way underneath the door by scratching furiously at the carpet at the foot of the door. This obnoxious noise wakes my wife and me up constantly. Therefore, I wanted to design a robot that not only prevented her from scratching but also habitually trained her to not scratch at the door.

But I didn't want her to be afraid of robots all the time. So I equipped Zobot with a laser pointer, a cat's natural enemy. During the day, when my wife and I are gone, the robot will move the mystifying red dot around the room while Zoey chases it. As soon as Zoey pounces on the laser, the robot will move the laser to a new location, offering a cornucopia of endless fun!

In order to do this, the robot will need to be able to:

- Avoid obstacles
- Determine day or night setting
- Locate my cat upon scratching
- Stop my cat from scratching
- Entertain my cat through the use of a cat toy

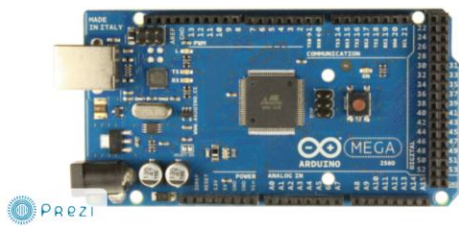
4. Integrated System

The ATxmega64A1 processor is the used to control the robot and is included in the Epiphany DIY microcontroller board. This board was chosen due to the plethora of peripherals available and because of the seller's availability to service the board. For instance, this board comes pre-equipped with motor drivers for my 12 volt motors, a 5 volt rail for my laser, and 24 servo ports.

Due to compatibility issues with Atmel's Two Wire In drivers and the Melexis infrared temperature sensor, I used an Arduino Mega 2560 to communicate with the sensor and then relay it to the Xmega.

Hardware

Epiphany DIY from
Out of the Box, LLC



Mega 2560 from Arduino

Figure 4.1. This figure shows the Epiphany board (upper right) and the Arduino (lower left) as they were used. This image was produced in Prezi.

5. Mobile Platform

Zobot's platform is made mostly out of a 1/8th inch thick piece of balsa wood. This offers a lightweight and nonconductive surface to mount the mechanical and electrical systems. The platform itself is cut into an octagonal shape to eliminate sharp corners. This was done to lessen the chance of the robot getting caught on a wall when it rotates. The robot's platform can be found in Figure 5.1



Figure 5.1. The completed version of Zobot. This image was produced in Instagram.

A PVC coupling was used to house the cleaning duster to allow it to be replaced. This coupling was glued to the platform to save space, and a spacer was placed inside to duster the duster a press fit into the coupling. To give the laser pointer ample surface area to play on, the pan and tilt servo was mounted 10.5" high on a 1"x3.5" piece of wood. To increase the stability of the post and coupling, they were glued together as well. This helped increase their moment of inertias, and rely less on the screws.







Zobot uses two-wheel steering with a plastic ball caster on the back. The plastic ball caster was chosen to help eliminate the friction between the caster and the carpet it will be operating on. A single level platform was chosen to keep the center of gravity low, and the robot more stable.

The stainless steel cover and sonar brackets were chosen purely for aesthetic reasons. Careful care was taken to make sure the metal does not short out any of the electronics.

6. Actuation

The actuators used in Zobot are described in Table 6.1. The reasoning behind their selection can be found after the table.

Table 6.1. A description of the Actuators used in Zobot.

Photo	Sensor	Description
	TTL Laser Module	This magical device shines a mystifying red dot at the ground. This red dot captivates Zoey to the point she must attack it. It will only be on during the Play Mode sequence.
	Pan and Tilt Servo from TrossenRobotics.com	This system forms a two degrees of freedom joint. It is used to point the laser pointer at the ground in varying locations. It will also allow the thermometer to sweep back and forth horizontally while searching for the cat.
	Ativa Cleaning Duster	This air canister will be discharged when the cat is scratching. Zobot was designed to interchange air canisters easily. Therefore, the only modification to the air canister was the addition of rubber bands to help the firing servo generate the necessary force to discharge it.
	Parallax Standard Servo from Radioshack	This servo is used to fire the compressed air canister when the cat is scratching. The air canister must be pre-loaded with springs to aid the servo in firing.
 www.pololu.com	67:1 Metal Gearmotor 37Dx54L mm from Pololu.com	These are the motors used to drive the robot. The wheels are attached directly to the motor because they are already gear to an optimal ratio.
	16x2 LCD - Amber on Black from Sparkfun.com	This LCD is used to display information from the robot to the user.

In order to entertain the cat, the robot will be equipped with a pet-friendly laser pointer. This laser pointer will be attached to pan and tilt servo motors so that it can be rotated in both directions orthogonal to its line of action.



In order to stop the cat from scratching, the robot will be equipped with a cleaning duster canister. When the cat begins to scratch, the robot will drive over to the cat and expel compressed oxygen. This will not only force the cat to retreat, but it will also relate scratching to a negative consequence. The trigger will be squeezed by using a simple servo motor that is attached to the robot's body.

The 67:1 gear ratio motors were chosen to increase the control of the robot when it is turning to face the cat. If the motors did not have enough torque, there would be more error introduced due to unequal wheel conditions. Also, if the robot turned too slowly, the increase spinning time would allow more error due to inconsistent flooring.

7. Sensors

Table 7.1 shows the sensors used in Zobot as well as a short description of their purpose. After the table, the reasoning behind choosing these sensors is described. Zobot has a total 8 sensors, 5 of which are unique.

Table 7.1. Description of the sensors used and the behaviors they are used in.

Picture	Sensor	Behavior	Description
	Melexis Non-Contact Thermometer Product Number: MLX90614ESF-ACF	Guard Mode Play Mode	This sensor determines the temperature of an object using infrared imaging. It is used on my robot to determine where Zoey is in relation to my robot. This particular sensor is special because it uses Two-Wire-In interface to communicate with my processor.
	Inex ZX-Sound	Guard Mode	The microphone is used to detect when the cat scratches. Once my cat scratches the microphone will tell the robot to start looking for her.
	Photoresistor provided to me by Joseph Osentowski	Guard Mode Play Mode	The photoresistor is used to determine whether it is day or night. If it is day, the robot will enter Play Mode and if it is night the robot will enter Guard Mode.
	Ultrasonic Range Finder - Maxbotix LV-EZO	Guard Mode Play Mode Obstacle Avoidance	There is one Sonar Sensor in the middle of the robot to determine how far the cat is when the robot is driving toward. This prevents the robot from running over my cat. It also prevents the robot from going underneath my bed and damaging the pylon that holds up the infrared temperature sensor and laser actuator. It is also used in Play Mode to determine if there is adequate space to play with the cat. If there is not enough space, the robot will move around until a good area is found. During this moving around process, the robot will use these sonar sensors to determine the presence of obstacles and avoid running into them.
	Bump Switch with Roller from Radioshack	Obstacle Avoidance	This sensor is used as a last resort in obstacle avoidance. It is used to prevent the motors from stalling in case the robot runs into an obstacle and does not realize it.

The special sensor used in this robot is the Melexis MLX90614ESF-ACF Infrared Temperature. This Inter-Integrated Circuit based sensor will need careful software consideration. This sensor requires a supply voltage of 5V and has a 10^0 field of view. Due to the narrow field of view, this sensor will be placed on

the pan and tilt servo bracket to obtain the angular direction of the cat relative to the robot. The thermometer will be hooked up to the Arduino first and then sent to the Xmega, as outlined in the figure below.

Melexis Non-Contact Thermometer

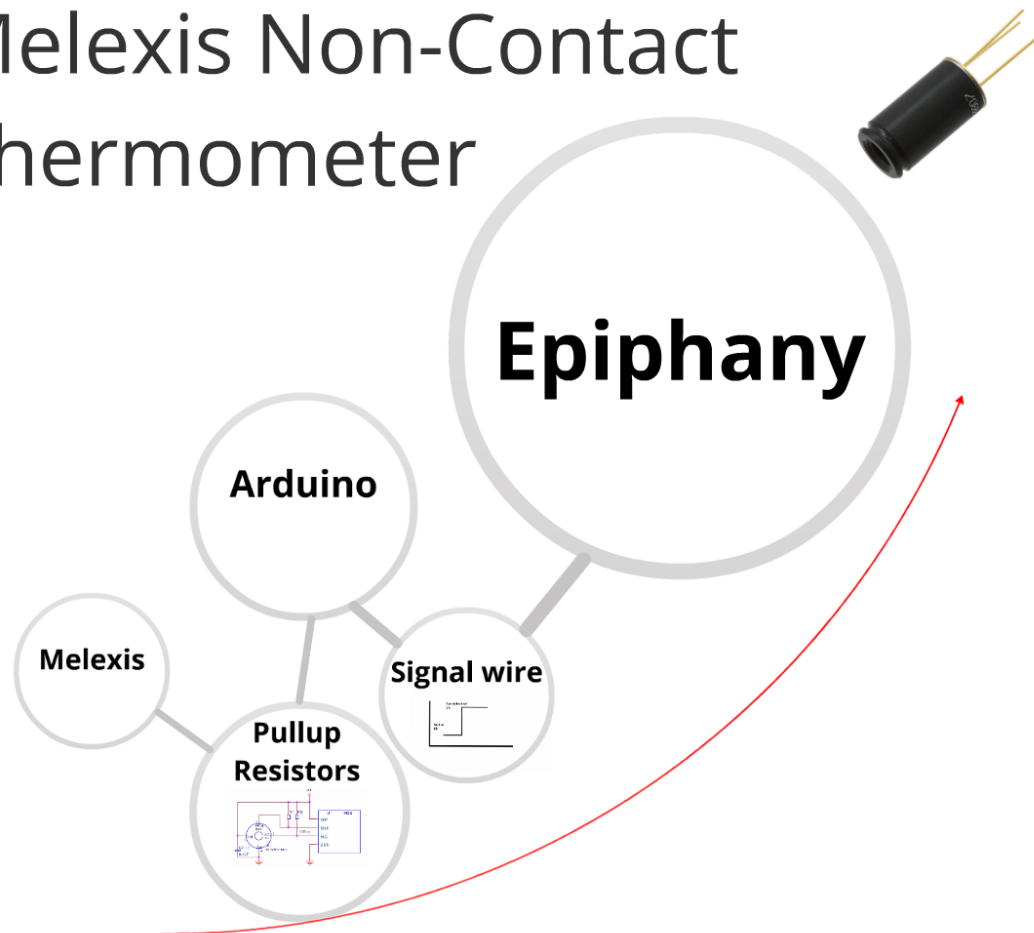


Figure 7.1. Interfacing Melexis temperature sensor with Ephiphany board.

The Arduino sends a high signal when it detects the presence of a warm object. Therefore, the Xmega looks for this signal when it is looking for the cat. The pullup resistor diagram used for this sensor can be found in Appendix A.

The microphone used is the Inex ZX Sound. It will be used in conjunction with the breakout board outlined in Appendix A. This sensor was chosen because it has operating conditions that fall in range with the scratching noise. In addition, it offered a stronger differentiation between ambient noise and the sound of the cat scratching than my original electret microphone.

The XL-Maxsonar EZ0 will be used in obstacle avoidance sequence to detect obstacles before collision. This sensor was chosen because of its precise field of view and optimum range of 6.5 inches to 25 feet. When these sensors fail, simple limit switches will be used to prevent the motors from stalling out. When these limit switches are activated, the robot will know that it has hit an immovable object and it

will stop the motors and turn around. The wiring diagram used with the bump switches can be found in Appendix A.

A simple photoresistor was used to determine which mode will be used. Because the resistance across the photoresistor varies with light, it is implemented in series with another resistor. The design of this circuit can be found in Appendix A.

8. Behaviors

As mentioned previously, the robot will feature two different modes selected based on the photoresistor's data. The first mode will be Play Mode, and it will be activated during the daytime. If the robot perceives that there is light, the robot will activate the on-board laser pointer. It will then entice the cat to play by wiggling the laser pointer on the ground. Once the infrared thermometer senses that the cat detected has attacked the dot, the robot will move the dot to a new location. A flowchart of this behavior generated in Microsoft Visio can be found in Figure 8.1.

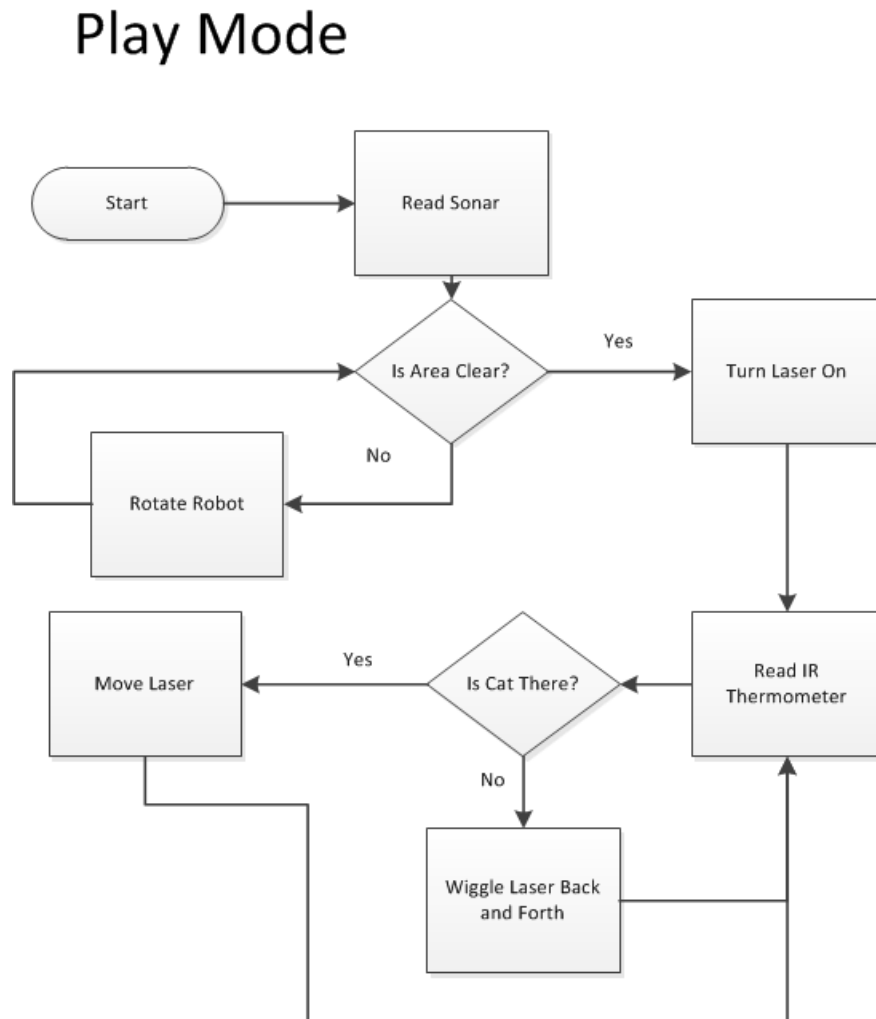


Figure 8.1. Flowchart of Play Mode behavior.

The second mode will be a Guard Mode. It will wait by the cat’s favorite scratching zone. After it hears a noise some percentage above the ambient noise level, it will activate the infrared thermometer to seek out the cat. Upon locating the cat, it will drive over to it, and when it comes in range, it will fire a burst of compressed oxygen. A flowchart of this behavior generated in Microsoft Visio can be found in Figure 8.2.

Guard Mode

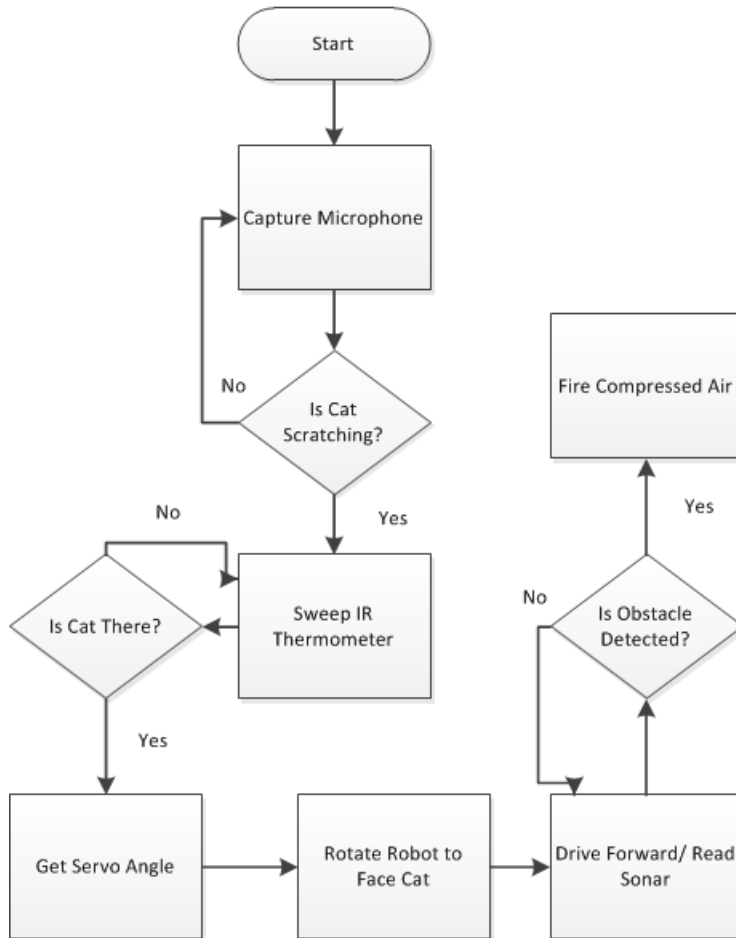


Figure 8.2. Flowchart of Guard Mode behavior.

9. Experimental Layout and Results

To use the photoresistor to determine the Robot’s mode, a threshold is used. This threshold was set by measuring the signal under various conditions. After the ADC values were determined at both daytime and nighttime, the square root of their product was used as the governing threshold.

In order to convert the sonar's input signal to inches, a series of tests were conducted. The first test was to find the minimum recordable distance that the sonar could pick up. After this was determined, 12 ADC signals were recorded at varying distances inside the operational zone of the robot. These were then averaged and plotted against their corresponding distances. A linear regression was then fitted to this graph in Excel to find the slope and offset needed to convert the signal to inches. The results for the three sonars are shown in Figure 9.1.

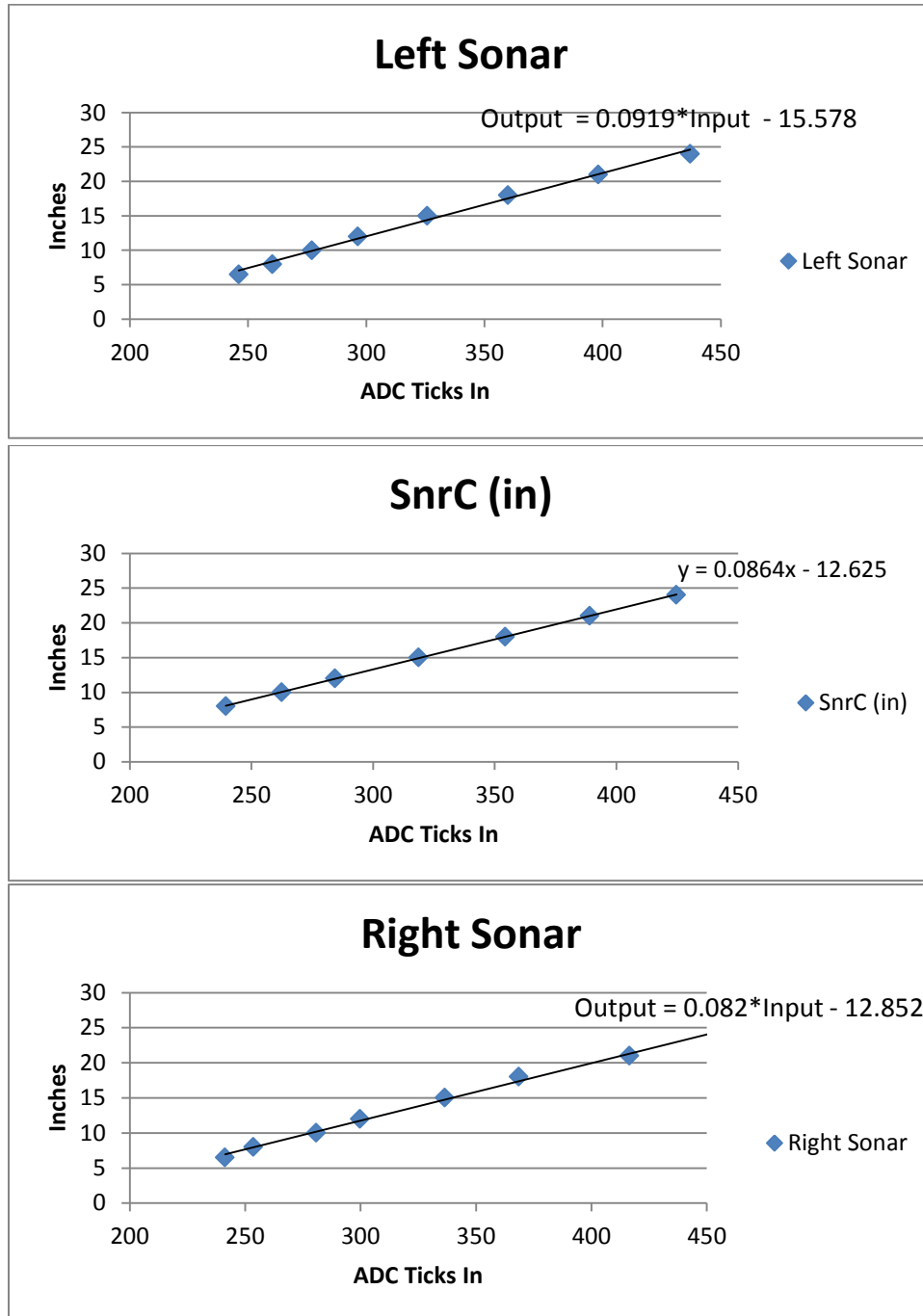


Figure 9.1. The linear representation of the Left and Right sonar sensors.

It should be noted that the slopes of the two sonar sensors vary significantly from the manufacturer specified 0.125 inches per ADC ticks in.

In order to determine the delay time to rotate the robot to face the cat, the turning rate was measured. This was done by setting up a series of lines corresponding to different angles and finding the appropriate delay time to reach each angle. After testing, it was determined that for every degree the robot needed to turn, it must turn for approximately 30 milliseconds.

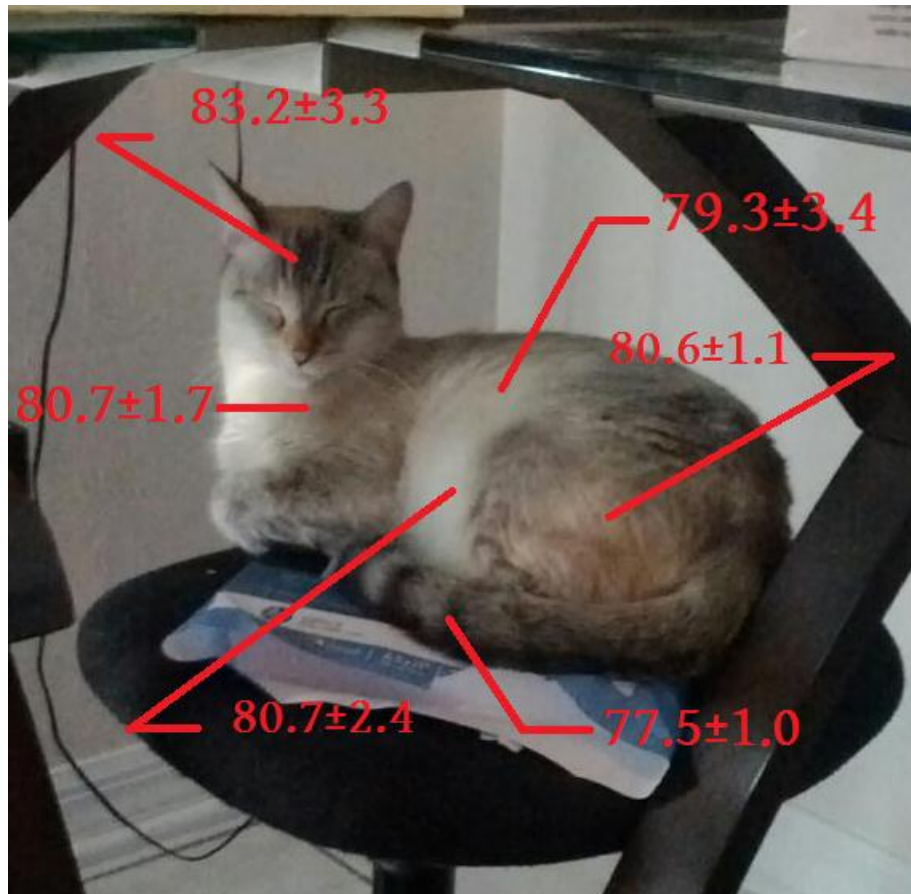


Figure 9.2. Temperature recordings of the cat in Fahrenheit.

In order to determine if the cat will be tracked by an infrared thermometer, a preliminary test was conducted to determine the cat's body temperature. Using a pre-calibrated infrared thermometer, a series of temperature data were collected in 6 different body zones of the cat. In every zone, not including the tail, the cat was at least three degrees warmer than the room temperature of 76° Fahrenheit as seen in Figure 9.2.

Since this is a suitable temperature difference, the next test will involve tracking the change in ADC ticks when sweeping the infrared thermometer sensor over the cat. In order to ensure the robustness of the sensor, this procedure will need to be done over an array of possible operating-temperature conditions.

10. Conclusion

Despite many close calls with batteries frying, LCD screens shorting, and wheels falling off, Zobot was successful in playing with Zoey and has passed all the testing requirements in Guard Mode. Throughout this lab, I have grown in my understanding of intelligent machines. Despite starting out as a mechanical engineer that did not know the difference between a wire crimper and a wire stripper, I have been able to successfully build a robot that plays and teaches my cat.

If I had more time this semester, I would have rewritten the drivers to get my infrared temperature sensor to work on my Xmega instead of having to use the Arduino. I would have also used multiple infrared thermometers in an array pattern in order to increase sensitivity mode in Play Mode and enable tracking in the attack sequence in Guard Mode.

Zobot's success with playing with Zoey was caught on camera and can be found on Zoey.Krieger's Youtube channel along with videos of the Guard Mode.

No cats were harmed in the making of this robot. Unfortunately, the same cannot be said about me.

11. Documentation

Epiphany DIY Board

<https://sites.google.com/site/epiphanydiy/>

Arduino Mega 2560

<http://arduino.cc/en/Main/ArduinoBoardMega2560>

Melexis Infrared Thermometer

<http://www.melexis.com/Infrared-Thermometer-Sensors/Infrared-Thermometer-Sensors/MLX90614-615.aspx>

Inex ZX Sound Microphone

http://www.inexglobal.com/downloads/ZX-sound_e.pdf

Sonar Sensor

http://www.maxbotix.com/documents/MB1000_Datasheet.pdf

LCD Screen

<http://www.sparkfun.com/datasheets/LCD/ADM1602K-NSA-FBS-3.3v.pdf>

TTL Laser Guide

<http://www.sparkfun.com/tutorials/260>

12. Appendices

Appendix A – Common Circuits Implemented

Photoresistor circuit used:

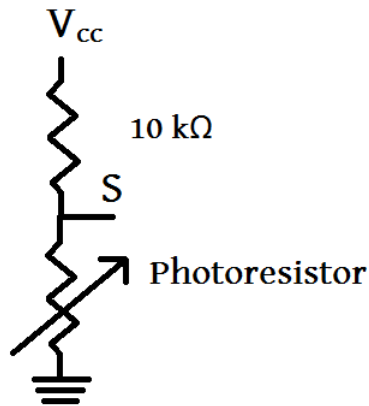


Figure A1. The circuit used for the Photoresistor, provided by the societyofrobots.com.

The resistor value was determined by the following equation:

$$R = \sqrt{R_{dark} \cdot R_{light}} \quad (A1)$$

Where R_{dark} was the measured resistance when the photoresistor was covered ($30.33\text{ k}\Omega$), and R_{light} is the measured resistance when a light was shown on the photoresistor ($2.65\text{ k}\Omega$).

Bump switch circuit used:

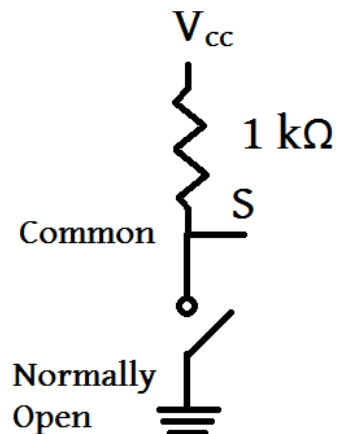


Figure A2. The circuit used for the bump switch, provided by teaching assistant, Josh Weaver.

Appendix A

Breakout board diagram for microphone used:

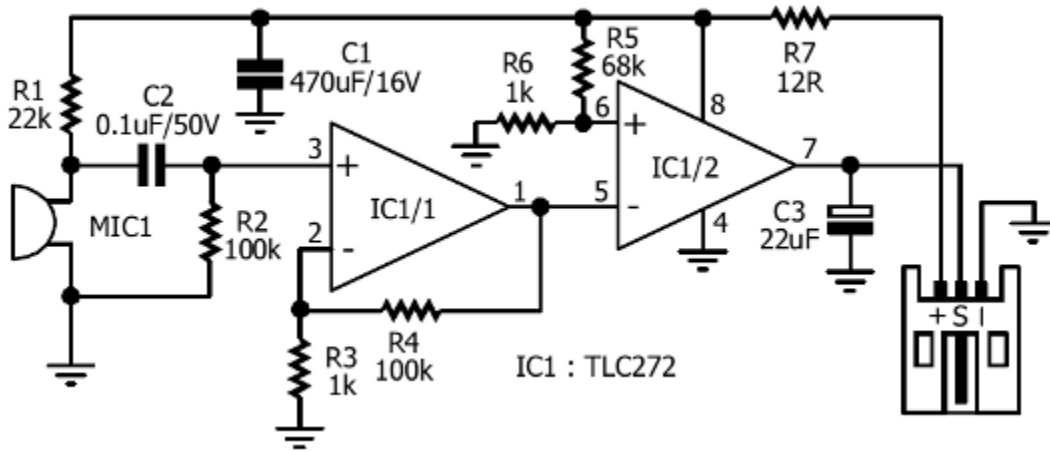


Figure A3. The circuit used in the breakout board to amplify the microphone voltage, provided by robosavy.com.

Appendix A

Infrared Thermometer circuit implemented:

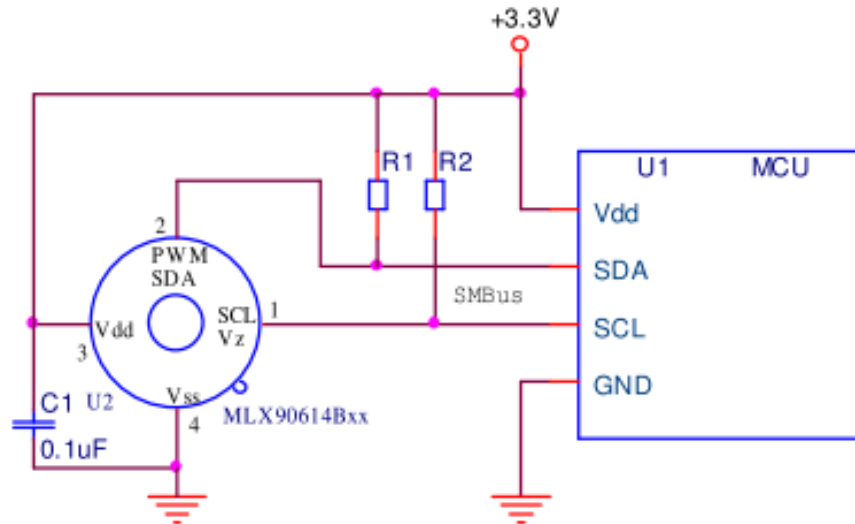


Figure 34: MLX90614 SMBus connection

Figure A4. The circuit used in the implementation of the infrared thermometer, provided by pololu.com.

Appendix B – Source Code

In an effort to save space, only the main functions are shown. Code that came prepackaged with the Epiphany DIY board has been omitted below.

Special Sensor

```
/******  
Reads in from the Melexis MLX90614ACF  
using i2C aka TWI aka SMBus protocol  
and sends a 5V signal from the digital  
I/O pin when the read in temp is over  
the ambient temperature threshold  
  
Uses Peter Fleury's i2cmaster library  
Thanks Peter!  
*****/  
  
#include <i2cmaster.h>  
  
void setup()  
{  
  pinMode(13, OUTPUT); //enables LED output control  
  pinMode(26, OUTPUT); //enables digital pin26 output control  
  i2c_init(); //initializing i2C  
  Serial.begin(9600); //initializing the serial monitor  
}  
  
void loop()  
{  
  //Declaring i2C variables  
  int slave = 0x5A<<1; //Shifting Slave Address one bit to the left, library-specific.  
  int cmd = 0x07; //Command corresponding to Tobj1 location in RAM on the MLX  
  int dataLo = 0; //The lower byte of the temperature data is sent first over the i2c  
  int dataHi = 0; //The upper byte (only 7 bits) is sent after the low byte  
  int PEC = 0; //Error byte  
  
  //Begin i2C transmission for the temperature from the Melexis, Tobj1  
  i2c_start_wait(slave+I2C_WRITE); //Tells arduino to send start command (when SCL is high, bring SDA low) and  
  begin writing something  
  i2c_write(cmd); //Write the RAM address corresponding to Tobj1 in the Melexis  
  
  //After writing the command, the device is now ready to send data. First, a repeated start must be sent, then  
  acks/nacks after every read  
  i2c_rep_start(slave+I2C_READ); //Sending repeated start command (when SCL is high, bring SDA low) to the slave  
  address and begin reading process  
  dataLo = i2c_readAck(); //Read the low data byte in Tobj1, and send ack  
  dataHi = i2c_readAck(); //Read the high data byte in Tobj1, and send ack  
  PEC = i2c_readNak(); //Read the error data byte, and send nack because transmission is over  
  i2c_stop(); //Send the stop condition (when SCL is high, bring SDA high)  
  
  //Declaring conversion variables  
  int Tobj1 = 0x0000; //Tobj1 is the temperature received by the MLX device  
  double d2Kslope = 0.02; //0.02 resolution per least sig. bit via the MLX90614 Data sheet  
  double K2Coffset = 273.15; //Converts Kelvin to Celsius by an offset  
  double C2Fslope = 1.8; //Multiplication factor when converting from Celsius to Fahrenheit  
  double C2Foffset = 32; //Offset factor when converting from Celsius to Fahrenheit  
  
  //Converting the data from LSB to Fahrenheit
```

Appendix B

//The first bit in the dataHi byte is an error flag, dataHi must be bit masked and then shifted left and dataLo is concatenated

```
Tobj1 = (int)(((dataHi & 0x007F) << 8) + dataLo);  
Tobj1 = (Tobj1 * d2Kslope);           //Converting from LSB to Kelvin  
Tobj1 = Tobj1 - K2Coffset;           //Converting from Kelvin to Celsius
```

```
Tobj1 = Tobj1 * C2Fslope + C2Foffset; //Converting from Celsius to Farenheit  
Serial.println(Tobj1);              //Outputing to serial monitor for debugging
```

```
int Tamb = 75;                      //Ambient temperature of my house through emperical data. May need to calibrate.
```

```
//If temperature is above threshold, send signals to both the LED and Xmega  
if (Tobj1 > Tamb){  
    digitalWrite(13, HIGH); //turns on led  
    digitalWrite(26, HIGH); //sends 5V signal to Xmega  
    delay(50);              //finite amount of time for signal to be sent  
} else {  
    digitalWrite(13, LOW);  
    digitalWrite(26, LOW);  
}  
delay(50);                  //don't want to ping device to death.  
}
```

Appendix B

Main loop

```
//Determining which mode to be in
if(lightValue > lightValueThreshold){ //photoresistor detects its dark out if true
    Guard();
} else {
    Play();
}
```

Guard Mode

```
/**Guard Mode Functions**/
void Guard(void){
    // Guard Mode Sequence
    setServoAngle(2,sweepCenter); //setting pan&tilt motor straight
    setServoAngle(3,tiltCenter);

    fprintf(&lcd_str,"Guard Mode\n");
    sndAmb = sndCal(); //determine ambient sound
    wait4sound(1.5, .5); //while loop that forces robot to wait until sound
reaches 10% above or below ambient sound

    //sound threshold reached, begin searching for cat
    sweepAngle = search();

    //Found cat, turn towards cat
    faceTarget(sweepAngle);

    //Facing cat, run at it and spray
    attack();
}

int sndCal(void){
    int i;
    //determines ambient sound
    fprintf(&lcd_str,"\nCalibrating\n");
    sndAmb = 0;
    //calibrating microphone for conditions
    for (i=0;i<8;i++){
        sndAmbA[i] = analogRead_ADCA(6);
        fprintf(&USB_str,"sndAmbA%d: %d\r",i,sndAmbA[i]);
        sndAmb += sndAmbA[i];
        _delay_ms(50);
    }
    sndAmb = sndAmb/8; //taking average of 8 readings and making it the threshold
    fprintf(&USB_str,"sndAmb: %d\r",sndAmb);
    return sndAmb;
}

void wait4sound(double sndHi, double sndLo){
    //while loop that forces robot to wait until sound reaches 10% above or below
ambient sound
    fprintf(&lcd_str,"\nListening\n");
```

Appendix B

```
    snd = sndAmb;
    while(snd<sndAmb*sndHi && snd > sndAmb*sndLo){ //1.25 and .75 work for clapping
        snd = analogRead_ADCA(6);
        fprintf(&USB_str,"snd: %d\r",snd);
        _delay_ms(50);
    }
}

int search(void){
    //makes pan servo sweep back and forth to look for cat
    //returns the angle that the cat was found at

    fprintf(&lcd_str,"\nSearching\n");
    int theta = 0;
    int sweepSign = 1;
    bool sweepFlag = true;

    setServoAngle(3,tiltCenter - 5);

    //if cat straight ahead, say so
    cat = catThere();
    if(cat == true){
        sweepFlag = false;
        //sweepAngle = theta-sweepCenter;
        fprintf(&USB_str,"turning angle: %d \r",sweepAngle);
        theta = sweepCenter;
    }

    while (sweepFlag){

        setServoAngle(2,theta); //so angle = theta = 0 intially.

        //sensor read in
        cat = catThere();
        //bmpR = analogRead_ADCA(4);
        fprintf(&USB_str,"%d\r",cat);
        //if(bmpR < bmpThresh){
        if(cat == true){
            sweepFlag = false;
            //sweepAngle = theta-sweepCenter;
            fprintf(&USB_str,"turning angle: %d \r",sweepAngle);
            break;
        }

        if(theta >= 180){
            sweepSign = -1;
        }
        else if(theta <= 0){
            sweepSign = 1;
        }
        theta += sweepSign*sweepDist;

        //allowing servo to catch up, but checking for cat
        for (int i = 0; i<7; i++){
            _delay_ms(50);
        }
    }
}
```

Appendix B

```
        //sensor read in
        cat = catThere();
        //bmpR = analogRead_ADCA(4);
        fprintf(&USB_str,"%d\r",cat);
        //if(bmpR < bmpThresh){
        if(cat == true){
            sweepFlag = false;
            //sweepAngle = theta-sweepCenter;
            fprintf(&USB_str,"turning angle: %d \r",sweepAngle);
            break;
        }
    }
    // _delay_ms(300); //give time for servo to catch up
}

setServoAngle(2,sweepCenter);
fprintf(&lcd_str,"\nCat Found\n");
if (theta<=sweepCenter){
    return theta+5*sweepSign; //offsets the angle so robot attacks at center of
cat instead of just at first edge
} else {
    return theta;
}
}

void faceTarget(int sweepAngle){
    //rotates robot
    //sweepTime = sweepAngle*sweepSlope;
    //fprintf(&USB_str,"sweepAngle: %d \r",sweepAngle);
    fprintf(&lcd_str,"\nFacing Target\n");

    int i;

    if (sweepAngle > sweepCenter){
        //Right side

        setMotorDuty(1,turnSpeedL,MOTOR_DIR_FORWARD_gc);
        setMotorDuty(2,turnSpeedR,MOTOR_DIR_BACKWARD_gc);

        for (i=sweepCenter; i < sweepAngle; i++){
            _delay_ms(30); //30ms of turning at the speeds 600,590 for every
degree
        }
        setMotorDuty(1,0,MOTOR_DIR_BACKWARD_gc);
        setMotorDuty(2,0,MOTOR_DIR_FORWARD_gc);
    }
    else if(sweepAngle < sweepCenter){
        //Left Side

        //i=sweepCenter;
        //sweepAngle = sweepAngle+sweepCenter;

        //fprintf(&USB_str,"i: %d sweepAngle: %d i>sweepAngle:
%d\r",i,sweepAngle,(i>sweepAngle));
        setMotorDuty(1,turnSpeedL,MOTOR_DIR_BACKWARD_gc);
        setMotorDuty(2,turnSpeedR,MOTOR_DIR_FORWARD_gc);
        for (i=sweepCenter; i > sweepAngle; i--){
```

Appendix B

```
        //fprintf(&USB_str,"i: %d sweepAngle: %d\r",i,sweepAngle);
        _delay_ms(30);
    }
    setMotorDuty(1,0,MOTOR_DIR_FORWARD_gc);
    setMotorDuty(2,0,MOTOR_DIR_BACKWARD_gc);
}

}

void attack(){
    bool catAhead = 0;
    //Driving Forward until sonar detected
    fprintf(&lcd_str,"\nAttack\n");
    if (catAhead == 0){
        setMotorDuty(1,attackSpeedL,MOTOR_DIR_FORWARD_gc);
        setMotorDuty(2,attackSpeedR,MOTOR_DIR_FORWARD_gc);

        while(catAhead == 0){
            snrC = analogRead_ADCA(1);
            snrCin = snrC * snrCslope - snrCoffset;

            if(snrCin < firingDist){
                setMotorDuty(1,0,MOTOR_DIR_BACKWARD_gc);
                setMotorDuty(2,0,MOTOR_DIR_BACKWARD_gc);
                catAhead = 1;
            }
        }
        //drove toward cat, now fire
        fire();
    }
}

//*****Firing Sequence*****
void fire(void){
    setServoAngle(1,firingAngle);
    _delay_ms(750);
    setServoAngle(1,firingNeutral);
    _delay_ms(750);
}
```


Appendix B

Play Mode

```
// *****Play Mode Functions*****
void Play(void){
    fprintf(&USB_str,"Play Mode\n");
    PORTF.OUTCLR = PIN7_bm; // turns laser on

    //rotates looking for a clear area to play
    findClearArea();

    setServoAngle(1,firingNuetral+10); // so it doesnt accidentally fire once its
clear

    //Variables used to determine the random angles
    int theta2max = 180;
    int theta3min = 60;
    int theta3max = tiltCenter-theta3min; //adding 20 degrees at the end so range is
20-90
    int theta2,theta3;
    int trollFlag = 0;

    uint16_t kittyCounter = 0;//timeout used for wiggleLaser
    while(1){
        if (trollFlag == 0){
            theta2 = rand() % theta2max;
            theta3 = rand() % theta3max + theta3min;
            fprintf(&lcd_str,"theta: %d %d\n",theta2, theta3);
            setServoAngle(2,theta2);
            setServoAngle(3,theta3);
            _delay_ms(300); //to not trigger a reactivation
        } else {
            trollFlag = 0;
            kittyCounter = 0;
        }
        while(cat == 0){
            //waiting for cat to attack laser
            cat = catThere();
            if (cat){
                break;
            }

            //grab the cat's attention by wiggling the laser after 20 seconds of
inactivity

            kittyCounter++;
            if(kittyCounter>500){
                wiggleLaser(theta2, theta3);

                // delay an extra 60 milliseconds when wiggling because a lot
of servo action and check for cat
                cat = catThere();

                //check for cat
                if (cat){
                    break;
                }
            }
        }
    }
}
```

Appendix B

```
        _delay_ms(20);

        cat = catThere();
        //check for cat
        if (cat){
            break;
        }
        _delay_ms(20);

        cat = catThere();
        //check for cat
        if (cat){
            break;
        }
        _delay_ms(20);
    }

    //move laser after 30s of inactivity
    if (kittyCounter>600){
        //troll(theta2, theta3);
        //trollFlag = 1;
        break;
    }
    _delay_ms(40); //minimal delay for servos
}
kittyCounter = 0;
cat = 0;
}
}

void findClearArea(){
    bool areaClearFlagLocal = 0;
    bool turning = 0;
    while (areaClearFlagLocal == 0){
        snrL = analogRead_ADCA(0);
        snrC = analogRead_ADCA(1);
        snrR = analogRead_ADCA(2);
        bmpL = analogRead_ADCA(3);
        bmpR = analogRead_ADCA(4);

        //converting ADC ticks in to inches using a linear approximation
        snrLin = snrL * snrLslope - snrLoffset;
        snrCin = snrC * snrCslope - snrCoffset;
        snrRin = snrR * snrRslope - snrRoffset;

        fprintf(&USB_str, "SnrL: %d SnrC: %d SnrR: %d\r", snrLin, snrCin, snrRin);

        if(bmpL<bmpThresh && bmpR < bmpThresh){
            //stop motors
            setMotorDuty(1, speedL, MOTOR_DIR_BACKWARD_gc);
            setMotorDuty(2, speedR, MOTOR_DIR_BACKWARD_gc);
            fprintf(&lcd_str, "Area Clear\n");
            areaClearFlagLocal = 1;
            break;
        } else if(bmpL<bmpThresh && bmpR >= bmpThresh){
            setMotorDuty(1, speedL, MOTOR_DIR_FORWARD_gc);
```

Appendix B

```
        setMotorDuty(2, speedR, MOTOR_DIR_BACKWARD_gc);
    } else if (bmpL >= bmpThresh && bmpR < bmpThresh) {
        setMotorDuty(1, speedL, MOTOR_DIR_BACKWARD_gc);
        setMotorDuty(2, speedR, MOTOR_DIR_FORWARD_gc);
    }

    if (snrLin > playDist && snrRin > playDist && snrCin > playDist) {
        //if all clear, tell motors to stop and begin playing sequence
        _delay_ms(1000);
        setMotorDuty(1, 0, MOTOR_DIR_FORWARD_gc);
        setMotorDuty(2, 0, MOTOR_DIR_FORWARD_gc);
        fprintf(&lcd_str, "Area Clear\n");
        areaClearFlagLocal = 1;
        break;
    } else if (snrLin > playDist && snrRin > playDist && turning == 0) {
        //if object is straight ahead, and left and right is clear, rotate
        towards whatever is more clear

        if (snrLin > snrRin) {
            //left is more clear, rotate left
            setMotorDuty(1, speedL, MOTOR_DIR_BACKWARD_gc);
            setMotorDuty(2, speedR, MOTOR_DIR_FORWARD_gc);
        } else {
            //right is more clear, rotate right
            setMotorDuty(1, speedL, MOTOR_DIR_FORWARD_gc);
            setMotorDuty(2, speedR, MOTOR_DIR_BACKWARD_gc);
        }
        areaClearFlagLocal = 0;
        turning = 1;
    } else if (snrLin > playDist && turning == 0) {
        //if object to the right and left is clear, rotate left

        setMotorDuty(1, speedL, MOTOR_DIR_BACKWARD_gc);
        setMotorDuty(2, speedR, MOTOR_DIR_FORWARD_gc);
        areaClearFlagLocal = 0;
        turning = 1;
    } else if (turning == 0) {
        // if object to the left or left and straight, rotate right

        setMotorDuty(1, speedL, MOTOR_DIR_FORWARD_gc);
        setMotorDuty(2, speedR, MOTOR_DIR_BACKWARD_gc);
        areaClearFlagLocal = 0;
        turning = 1;
    }
}

}

void wiggleLaser(int theta8, int theta7) {
    //theta8 = theta2, theta7 = theta3
    int mag8, sign8, mag7, sign7;
    fprintf(&lcd_str, "\nwiggle wiggle\nwiggle wiggle\n");

    //determining the wiggle magnitude and direction randomly
```

Appendix B

```
mag8 = rand() % 10; //needs more angle when panning then tilting
mag7 = rand() % 5;
sign8 = rand() % 1;
sign7 = rand() % 1;

if(sign8==1){
    setServoAngle(2,theta8+mag8);
} else {
    setServoAngle(2,theta8-mag8);
}
if(sign7 == 1){
    setServoAngle(3,theta7+mag7);
} else {
    setServoAngle(3,theta7-mag7);
}
}
```

Reading in from Arduino

```
bool catThere(void){
    catVal = analogRead_ADCA(7);
    fprintf(&USB_str, "%d\r", catVal);
    if (catVal > catThresh){
        return true;
    } else {
        return false;
    }
}
```