Biomimetic Rat

ME 5643 – Mechatronics NYU Poly

Bilal Gill, Harish Nair and Joseph Frezzo

Table of Contents

I. Introduction

The United States Census Bureau results put the population of New York City (NYC) in 2012 at approximately 8.3 million[1]. Although the estimates vary, experts have concluded that the rat population in NYC is approximately equal to the human population[2]. The ratio of rats to humans has shifted in favor of the rats since the events of superstorm Sandy. The storm which wreaked havoc along the coastlines, pushed a large population of rats inland to more populous areas providing a financial boon to commercial extermination services and a thorny annoyance to both the NYC Department of Health and the NYC Housing Authority[2,3].

The influx of rats and rat-related problems reached a tipping point in the approaching winter months following the storm as the number of 311 complaints increased by 21% in Manhattan's Lower East Side and nearly doubled in Lower Manhattan. Areas in Brooklyn such as Coney Island, Carroll Gardens, Red Hook and Greenpoint also experiences a significant surge in rat activity as complaints increased 70 over the same time the previous year[2]. The apparent surge in rat populations started during the storm and continues to be a problem for for two reasons a) the flooding of sewers inland forced rats from their normal habitat to the surface and b) the generation of so much damage debris created a new safe habitat for the rats to propagate.

Even more worrisome is the issue of condensed rat populations and eventual population growth. With greater rat populations comes greater risk of zoonotic disease transferal. It's been known for quite a long time that the bubonic plague of the middle ages was caused by the bacterium, *Yersinia pestis*, and the carrier of that bacteria were rats. Although public health conditions have improved drastically since the bubonic plague, rats are still an ample source of other potentially harmful viruses and other pathogens such as hantavirus, salmonella and typhus[3].

It is for these reasons that public health officials should educate the public on the behavior of rats for the purpose of preventing uncontrollable infestations. To engage the public from the ground up so to speak, we propose a robotic tool that can be used in grades kindergarten through twelfth grade (GK-12) that would provide a dynamic educational service without the need for actual handling of a physical rat or similarly evolved rodent. We propose here a Biomimetic Rat that shares with a rat many physical characteristics along with several sensor-based systems that mirror the physiologically-responsible behavior of a rat (Figure 1).

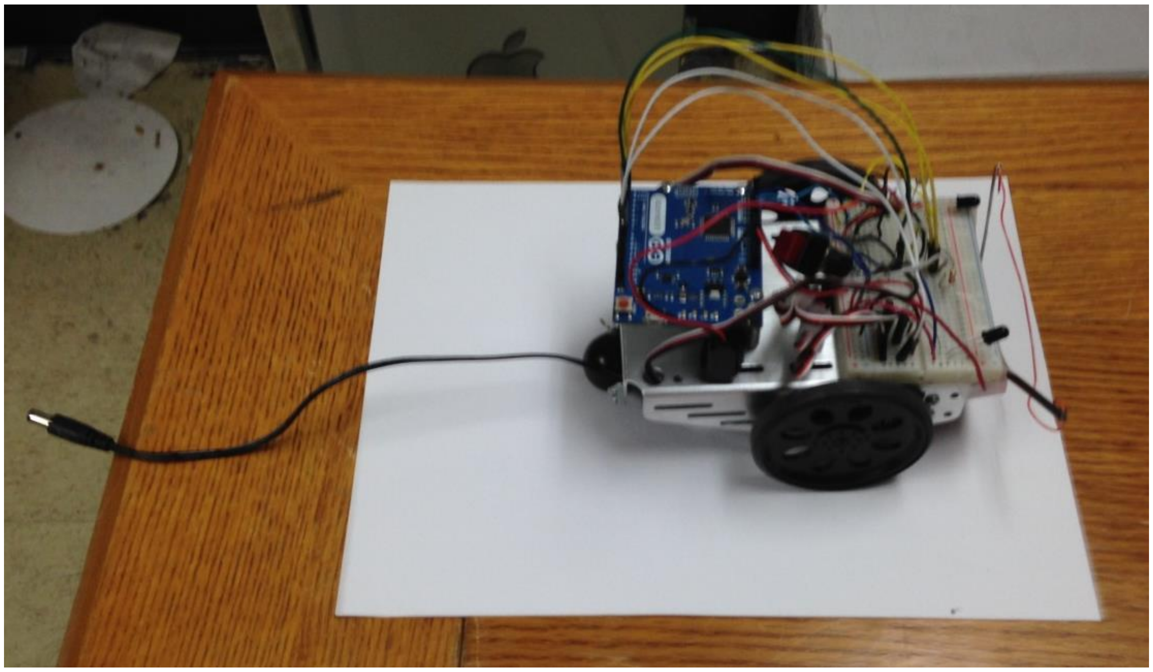

Figure 1. Biometic rat. The body of the biomimetic rat is built off a Board of Education Bot (BOE-BOT) frame and is controlled by an Arduino Leonardo microcontroller.

II. Rat Research For Sensor Design

A. Vision

Compared to human vision, the field of visibility for a rat is very poor. Much like colorblind humans, rats are dichromats so they only perceive colors in two wavelengths. Because their vision is poor in perceiving color, a greater emphasis is placed on distinguishing between black and white objects (Figure 2)[4]. Unfortunately for the rats, even their light-sensitive eye cells are severely lacking as evidenced from acuity studies.

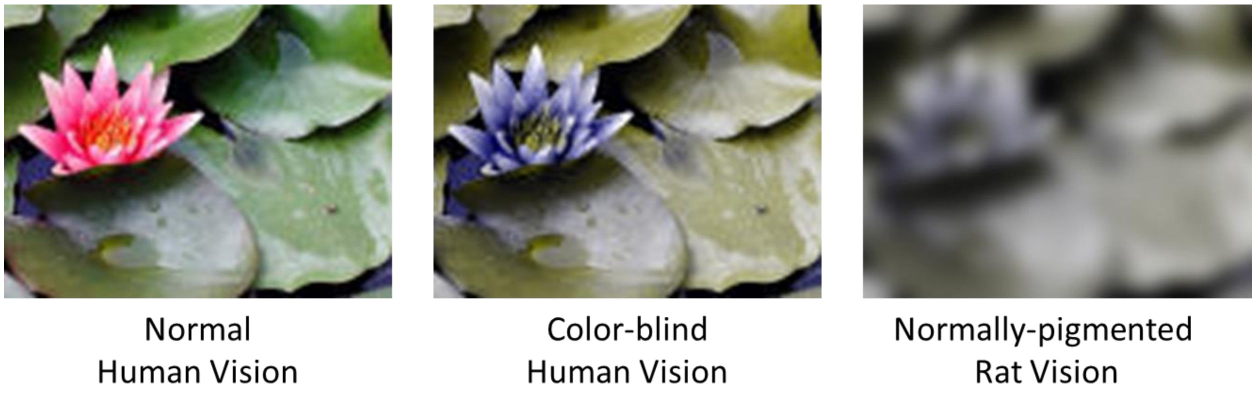

Figure 2. The perceived image of a water lily in normal human vision, dichromatic human vision and normal rat vision[4].

Acuity can be measured in two ways: a) physiologically in retinal density of ganglion cells or b) quantitatively in units of cycles per degree (cpd). Correlations between visual acuity and ganglion density has been determined across a spectrum of organisms. A rat's retinal ganglion density for instance is approximately 6800 cells/mm^2. A human ganglion density on the other hand is far greater at 38,000 cells/mm^2 (4) . The qualitative definition of cpd is simply "the measurement of the number of lines that can be seen as distinct within a degree of the visual field" (Figure 3)[4,5]. A rat eye possesses a cpd value of 1-1.5. For frame of reference, a human eye has a visual acuity between 30-50 cpd [4,5].

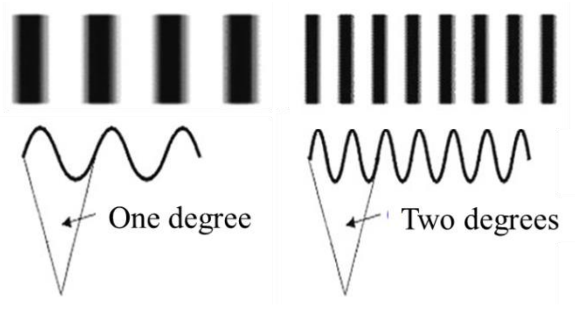

Figure 3. Cycles per degree representation for perceiving the number of distinct lines without confusion. A rat eye has a cpd value between 1-1.5 while a human eye has a cpd value in the range of 30-50[5].

In terms of rat behavior, it's fairly obvious that rats often seek areas of low light because although their vision isn't particular strong in this arena, it is still stronger than their color perception. It is also an survival adaptation in the face of poor vision. Rats will seek out areas adjacent to darker, sheltering materials. For this reason, we equipped our Biomimetic rat with photoresistor-dependent servos wheels. When the Biomimetic Rat is exposed to high incident light, the photoresistor will drive the servos circuit to mobilize to an area of lesser light intensity. This function is an excellent representation of what happens when a darkened room with rats inside becomes well-lit. The rats will scatter toward the walls in seeking an area of low light and hence more condusive to their survival.

B. Tactile Sensing

A theorized evolutionary tradeoff for poor vision is the rat's heightened sense of touch via their whiskers. The whiskers act almost as a complex sight sensory system as they convey immediate surroundings to the rat's brain. Much like the prickly sensation of goosebumps, a rats whiskers are controlled by piloerectile musles [6]. Research has shown that rat whiskers are arranged schematically according to the frequency at which the whiskers sweep back and forth (Figure 4)[6].

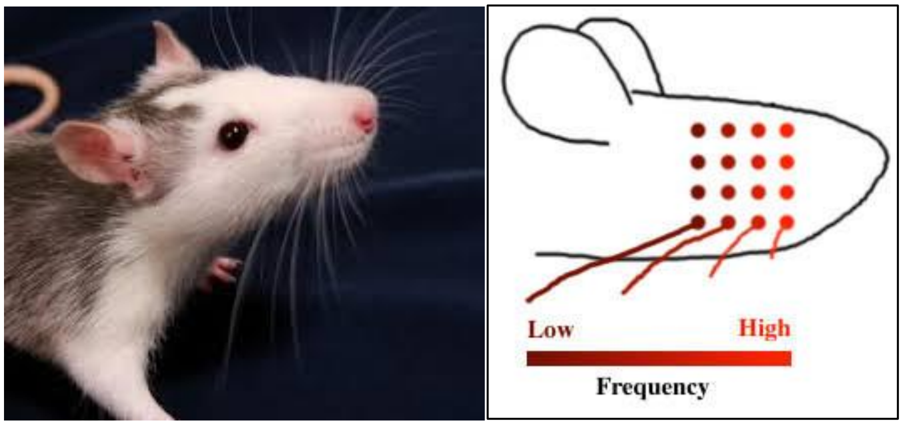

Figure 4. Rat whiskers on an actual rat and a representative image of the sweeping frequency by location on the rat head [6].

Areas on the most terminal end of the rat's head have whiskers that sweep back and forth at the highest frequencies while whiskers closest to the body have whiskers that sweep at lesser frequencies [6]. It's apparent in the figure above that whiskers have different lengths as well. The shorter whiskers are more sensitive simply from the fact that the piloerectile muscles can move them more quickly. This means that all whiskers are generally the same structurally but physiologically convery different senses due to their muscle activity. It is the physical bending of the whiskers that triggers the pseudo-sight sensation[6,7].

The whiskers, in their sweeping contact-dependent motion, are more reliable than vision so rats, as stated in the previous section, tend to seek areas of low light with materials around them to guide their movement. This is apparent, expecially in urban environments suchas NYC, where rats are predominantly seen in areas rich in debris and generally shaded. This became an issue after superstorm Sandy as significant damage created a mass of accumulated debris which provided a home for many rats.

In developing a similar system for the Biomimetic Rat, a contact dependent sensor needed to be added to the “head” of the robot. For this reason, two limit switch devices were engineered with each limit switch governing servos-dependent motion of the robot. This served as a simplified yet adequate biomimetic representation of rat whiskers.

C. Warmth-seeking Behavior

Rats, unlike humans, do not have a highly evolved temperature governor. Humans, when exposed to cold environments, will autonomously shiver as a means of generating heat to maintain body temperature. Rats do have this shivering function but they rely more on physically seeking heated areas to escape cold conditions [8]. This is apparent as rats, much like all rodents, are burrowing animals. The obvious reasons for this behavior is predator evasion but warmth is just as important to rats. In NYC this is even more apparent in the winter months, as rats are less likely to be seen on the streets and more likely to be seen in the relatively warmer undeground subway stations.

To mimic such behavior, the Biomimetic Rat is equipped with a heat-seeking sensor. The sensor itself is an ultraviolet diode programmed to be responsive to the specific wavelength of fire. Although, the Biomimetic Rat is seeking a specific wavelength of light and not heat itself, the UV diode is an excellength educational substitute that adequately demonstrates a rat’s tendency to seek out a warmer environment.

III. ROBOT DESIGN OVERVIEW

The basic purpose for the design of our project was to mimic some of the qualities exhibited by a rat for its survival in the urban environment. For our project we decided to concentrate on three main features.

- It's tendency to stay away from bright light and seek darker areas.
- It's behavior to seek heat during cold exposure.
- It's ability for tactile sensing using its whiskers.

To achieve these features, we used an array of sensors, mounted on top of the frame of a BOE-Bot, all controlled by an Arduino Leonardo Microcontroller.

A. Arduino Leonardo Microcontroller

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller[9].

There are many versions of the official Arduino hardware that have been commercially produced to date. For the purpose of our project, we are using an Arduino Leonardo Microcontroller, with the following specifications.

Arduino Leonardo Specifications	
Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz

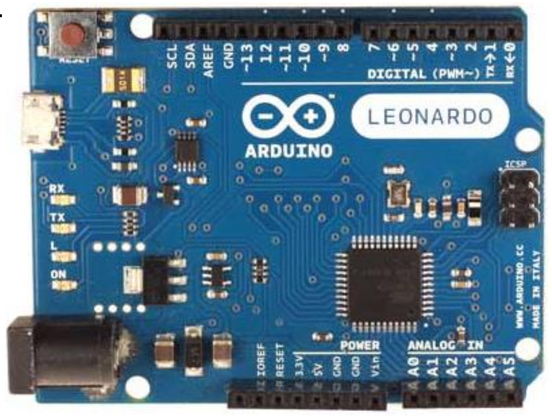
Figure5 - Arduino Leonardo

B. BOE-Bot Frame

The frame of our bot is the frame from the Parallax BOE-Bot Robot Kit. The frame consists of:

- Durable brushed-aluminum chassis with mounting holes for servos and accessories.
- Parallax continuous rotation servos for the drive wheels.
- Plastic machined wheels with rubber band tires.
- Tail wheel ball.

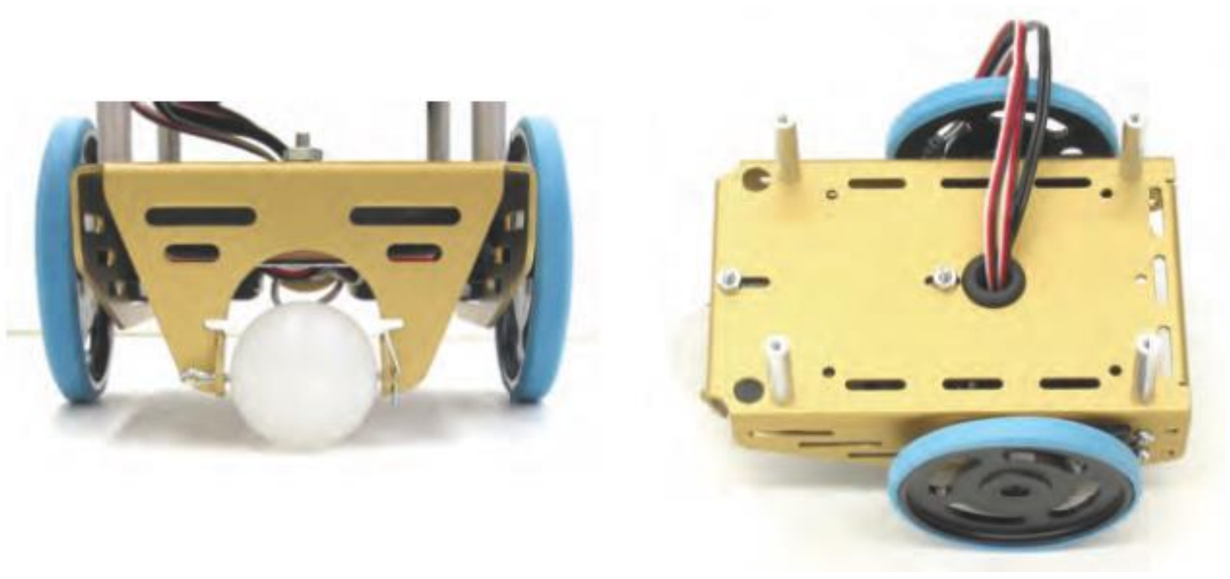

Figure 6. BOE-Bot Frame[10]

C. Sensors & Attachments

There are many different sensors and attachments used in our bot. Here is a brief description for each of them.

- **Servo Motors**

  A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. For our bot, we are using two Parallax Continuous Rotation Servos. The key features for these servos are:

  - Bidirectional continuous rotation.
  - 0 to 50 RPM, with linear response to PWM for easy ramping.
  - Weighs only 1.5 oz. (42.5 g).

Figure7. Parallax Continuous Rotation Servo[10]

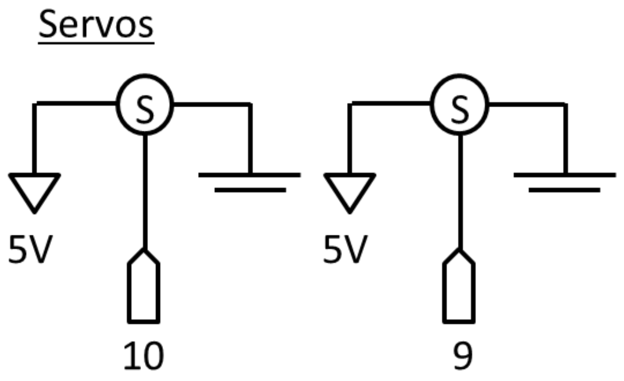

Figure8. Circuit Diagram for Servos

- **Limit Switch**

  A limit switch is an electromechanical device that consists of an actuator mechanically linked to a set of contacts. When an object comes into contact with the actuator, the device operates the contacts to make or break an electrical connection. For our bot, we are using two limit switches at the front of the bot, which acts as whiskers for a rat, and help to avoid collisions with walls.

Figure 9. Limit Switch[10]

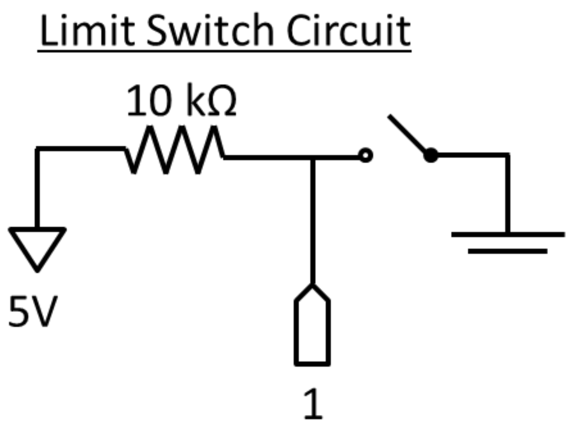

Figure 10. Circuit Diagram for Limit Switch

- **Ultraviolet Diode**

  For the purpose of mimicking the warmth seeking feature of the rat, we are using an Ultraviolet (UV) diode. There are two UV diodes at the front of the bot, which are programmed to be responsive to a specific wavelength of fire. With this the bot, would try to seek fire, which would act as mimicking warmth seeking feature of rats.

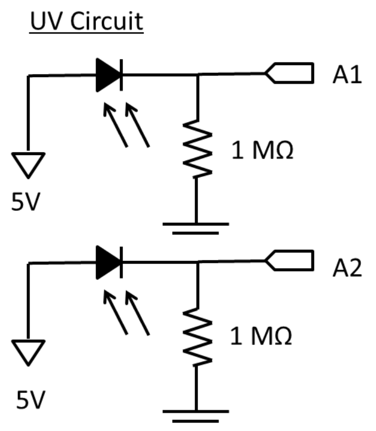

Figure 11. Circuit diagram for UV diodes

- **Photoresistor**

  A photoresistor is a resistor whose resistance decreases with increasing incident light intensity. For our bot to exhibit light avoiding feature, we have attached a photoresistor at the front of the bot, which is programmed to move the bot away light beyond a specified light intensity.

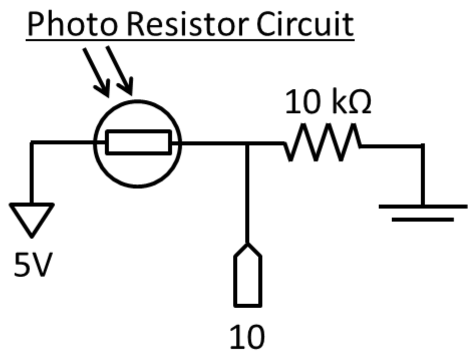

Figure 12. Circuit diagram for Photoresistor

- **Kill Switch**

  A Kill Switch or emergency stop is a safety mechanism used to shut off a device in an emergency situation in which it cannot be shut down in the usual manner. This switch is also provided in the bot to instantly stop the system at any given moment.

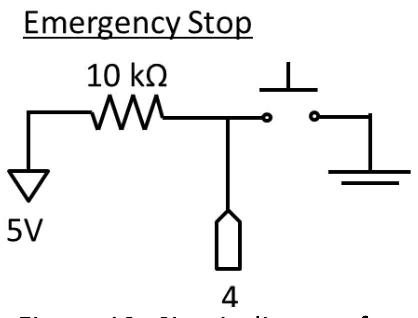

Figure 13. Circuit diagram for Emergency Stop

IV. Conclusion

The proposed Biomimetic Robot could be used as an educational tool for dynamic GK-12 lesson planning in the areas of science, technology, computer programming, and also public health. A few issues still remain however. For example, the robot is “attracted to fire” but the robot is not attracted to the warmth, but rather the specific wavelength of light that fore emits. This brings up the question of whether or not the robot will be attracted to a large fire and will effectively commit suicide by driving into the fire. This can be fixed by setting a limit on the light intensity and also attaching an additional temperature sensor that could limit the distance the robot moves toward the heat source. Another example of an attachment that can be fine-tuned is the limit switch whiskers. Perhaps more whiskers of varying intensity could be attached to the robot to convey the importance of tactile sensing being a alternative means of sight for the Biomimetic Rat.

V. Commented Code

<u>The main file</u>

```
#include "fire_detect.h" // These includes add all the custom written libraries
#include "PhotoResistor.h"
#include "ServoSpeed.h"
#include "Moves.h"
#include "Whiskers.h"
#include "EmergencyStop.h"
#include "State.h"

#include <Servo.h> // this is a standard arduino library
```

```
extern Servo leftServo; // This initializes the leftServo object in this part of the code
extern Servo rightServo; // the same as above but with the right servo

void setup()
{
  Serial.begin(9600);  //Begin serial communcation>
  leftServo.attach(PINLEFT); // Attaches the left servo object to a pin
  rightServo.attach(PINRIGHT); // Does the same but for the right servo
  ServoSpeed(RIGHT,100); // Sets the right servo to max speed since the beginning
  ServoSpeed(LEFT,100);  // Sets the left servo to max
  pinMode(1,INPUT);      // Sets pin one to read digital values
  pinMode(2,INPUT);      // Sets pin two to read digital values
  pinMode(5,OUTPUT);     // Sets pin five to set digital values
  pinMode(6,OUTPUT);     // Sets pin six to set digital values
  pinMode(7,OUTPUT);     // Sets pin seven to set digital values

}

void loop()
{

  while( EmergencyStop()) { // While the emergency stop button is not pressed
    WhiskerTurn(); // Check if your whiskers are pressed
    StateCheck();  // Check if it is light, dark or youre seeing a fire
    StateResponse(); // Do the appropriate action due to the state
    delay(10); //short delay to not overload the sensors
  }
```

```
   Stop(); // If the emergency stop is pressed, stop
}
```

EmergencyStop.h

```
#ifndef EmergencyStop_h // Defines the header file for the compiler
#define EmergencyStop_h

#include "Arduino.h"

boolean EmergencyStop(); // Function prototype for the Emergency Stop

#endif
```

EmergencyStop.cpp

```
#include "EmergencyStop.h"

boolean EmergencyStop() {
        return digitalRead(4); // Just read the pin that the Emergency stop latch switch is pressed to.
}
```

Fire_detect.h

```
#ifndef fire_detect_h
#define fire_detect_h

#include "Arduino.h"
#include "Moves.h" // WE have to add the library which has all the



enum PIN {LEFTONE, RIGHTONE}; // This is to decide whether to read the left or right UV Diode
```

```
float fire_detect(enum PIN pin); // This is the function which reads the analog from the UV Diode


void fire_follow(); // This is the function which helps follow the fire


#endif
```

Fire_detect.cpp

```
#include "fire_detect.h"


float fire_detect(enum PIN pin) { // The main function which reads the UV Diode
  if ( pin == LEFTONE) { // Read the left UV Diode
    float intensity;     // Make a variable for the read intensity
    intensity = analogRead(1);  // The read intensity is the value read from the first analog pin
    return intensity; // Return the read intensity
  }
  else if ( pin == RIGHTONE) { // do the same for the right pin
    float intensity;
    intensity = analogRead(2);
    return intensity;
  }
}


void fire_follow() {  // This is the function which follows the fire
  if ( (fire_detect(LEFTONE) < 1023) || (fire_detect(RIGHTONE) < 1023) && ((fire_detect(LEFTONE) > 50)
|| (fire_detect(RIGHTONE) > 50))) { // If the value of the left uv diode or right uv diode is within some
bound, go into this function
```

```
   if ( ((fire_detect(LEFTONE) - fire_detect(RIGHTONE)) > 50) && ((fire_detect(RIGHTONE) > 150) ||
(fire_detect(LEFTONE) > 150))) { // If the left UV diode is some value greater than the right uv diode and
both of them are reading greater than a threshhold, go into here

     while ((fire_detect(LEFTONE) - fire_detect(RIGHTONE)) > 50) { // while the left uv diode reading is
greater than the right uv diode

       LeftTurn((fire_detect(LEFTONE) - fire_detect(RIGHTONE))); // turn left at the speed of the difference

     }

     Forward(100); // go forward

   }

   else if ( (fire_detect(RIGHTONE) - fire_detect(LEFTONE)) > 50) { // the same thing as above but for the
right uv diode

     while (((fire_detect(RIGHTONE) - fire_detect(LEFTONE)) > 50) && ((fire_detect(RIGHTONE) > 150) ||
(fire_detect(LEFTONE) > 150))) {

       RightTurn((fire_detect(RIGHTONE) - fire_detect(LEFTONE)));

     }

     Forward(100); // go forward

   }

  }

  else{

   Stop(); // if you dont see a fire, stop until it reads the state again

  }

  if ((fire_detect(LEFTONE) < 150) || (fire_detect(RIGHTONE) < 150)) {

   Stop(); // If both of the uv diodes are below a threshold, stop

  }

}
```

Moves.h

```
#ifndef Moves_h
```

```
#define Moves_h


#include "Arduino.h"

#include <Servo.h> // Include the arduino standard 180 degree servo library

#include "ServoSpeed.h" // Include the library which sets the servo speeds


void LeftTurn(float Speed); // prototype ofr the left turn function

void Forward(float Speed);  // prototype of the forward function

void RightTurn(float Speed); // prototype of the right turn function

void Stop(); // prototype of the stop function


#endif
```

Moves.cpp

```
#include "Moves.h"


void LeftTurn(float Speed){ // The point turn left function

  ServoSpeed(RIGHT, Speed); // Set the speed of the right wheel to be positive of what is set

  ServoSpeed(LEFT, -Speed); // Set the speed of the left wheel to be negative of what is set

}

void Forward(float Speed){ // The move forward function

  ServoSpeed(RIGHT, Speed); // Set the right servo to the desired speed

  ServoSpeed(LEFT, Speed);  // Set the left servo the desired speed

}

void RightTurn(float Speed){ // The point right turn function

  ServoSpeed(RIGHT, -Speed); // Set the speed of the right servo to the negative of the desired speed
  ServoSpeed(LEFT, Speed);   // Set the speed of the left servo to the desired speed
}
void Stop() { // The stop function
  ServoSpeed(RIGHT, 0); // The right Servo Speed is set to zero
  ServoSpeed(LEFT,0);  // The left servo speed is set to zero
}
```

PhotoResistor.h

```
#ifndef PhotoResistor_h
#define PhotoResistor_h

#include "Arduino.h"
#include "Moves.h" // Include the moves library
#include "fire_detect.h" // Include the fire detect
#include "State.h"

enum STATE {LIGHT, DARK, FIRE}; // An enum which describes each one of the States

// This is the pin that reads the photoresistor
#define PIN 0

// This is the function prototypes

void StateCheck();

int PhotoResistor();
```

```
void StateResponse();


#endif
```

PhotoResistor.cpp

```
#include "PhotoResistor.h"


int lightCount = 0; // Hoe many times has the state been light

enum STATE lastState = LIGHT; // The first state is light

enum STATE State;


int PhotoResistor() {

  return analogRead(PIN); //This function was created to simplify reading the Photoresistor

}




void StateCheck() { //Check the State

        if (PhotoResistor() > 350) { // If the photo resistor is above a threshold

                lastState = State; // The last state is now the state

                State = LIGHT; // The current state is now light

                State_LED(1); // turn on the LED which dictates light

        }

        else if (PhotoResistor() < 350) { // If the photoresistor is below the threshhold

                lastState = State; // The last state is set

                State = DARK; // The current state is dark
                State_LED(2); // Turn on the Dark State LED
        }
        if ( (fire_detect(LEFTONE) > 150) && (fire_detect(RIGHTONE)      > 150)) { // If both UV Diodes
are above a threshold
                State = FIRE; // The state is set to fire
                State_LED(3); // The fire state LED is turned on
        }
}


void StateResponse() { // What to do for each state
        if ((State != FIRE)) { // First thing is to check if that state is not fire
                if ((State == LIGHT) && (lightCount < 10)) {// If the state is light and has been light less
than 10 times
                                LeftTurn(100); //turn left
                                delay(250);    // wait
                                Forward(100); // go forward
                                delay(250);   // wait
                                if (lastState == LIGHT) { // if the last state was light
                                        lightCount++; // add to the light count
                                }
                }
                if ((State == LIGHT) && (lightCount >= 10)) { // if the state is light and the past 10 states
have been light
                        Forward(100); // just go forward looking for dark
                }
                if ((State == DARK)) { // if the state is dark
                        lightCount = 0; // reset the light count
```

```
                    Forward(100); // keep going forward
            }
  }
  if (( State == FIRE)) {// if the state is fire
        fire_follow(); // engage fire following routine
  }
}
```

<u>ServoSpeed.h</u>

```
#ifndef ServoSpeed_h
#define ServoSpeed_h

#include "Arduino.h"
#include <Servo.h>

#define PINRIGHT 9
#define PINLEFT 10

enum SERVO {LEFT, RIGHT}; // This tells you the orientation of the servo

void ServoSpeed(enum SERVO whichone,float Speed); // This is the function which scales the speed of
the servo for you from 0 - 100

#endif
```

<u>ServoSpeed.cpp</u>

```
#include "ServoSpeed.h"
```

```
Servo leftServo;

Servo rightServo;



void ServoSpeed(enum SERVO whichone,float Speed) {

 if ( whichone == RIGHT)  { // This is the case for a servo on the right side of the robot

  if (Speed > 100) { // A check for the case if the speed is over 100%

   Speed = 100;

  }

  if (Speed < -100) { // A check for the case if the speed is under 100%

   Speed = -100;

  }

  if (Speed == 0) { // What to set the speed to if the speed is equal to zero

   rightServo.write(95);

  }

  else if ( Speed > 0) { // What to set the speed to if the speed is over 0

   rightServo.write(95-(.96*Speed));

  }

  else if ( Speed < 0 ) { // What to set the speed to if the speed is under zero

   rightServo.write(95-(.96*Speed));

  }

 }

  if ( whichone == LEFT)  { // This is almost the same thing except for a left oriented servo

  if (Speed > 100) {

   Speed = 100;
```

```
    }
    if (Speed < -100) {
      Speed = -100;
    }
    if (Speed == 0) {
      leftServo.write(95);
    }
    else if ( Speed < 0) {
      leftServo.write(95+(.96*Speed));
    }
    else if ( Speed > 0 ) {
      leftServo.write(95+(.96*Speed));
    }
  }
}
```

State.h

```
#ifndef State_h
#define State_h


#include "Arduino.h"
#include "PhotoResistor.h"


#define LIGHTPIN 5 // This is the pin which the Light LED is attached to
#define DARKPIN  6 // This is the pin which the DARK LED is attached to
#define FIREPIN  7 // This is the pin which the FIRE LED is attached to
```

```
void State_LED(int State);


#endif
```

State.cpp

```
#include "State.h"


void State_LED(int State) { // This function is to light up the state LED
        if (State == 1) { // IF the State is Light
                digitalWrite(DARKPIN,LOW); // Turn off the dark pin
                digitalWrite(FIREPIN,LOW); // Turn off the fire pin
                digitalWrite(LIGHTPIN,HIGH); // turn on the light pin
        }
                if (State == 2) { // The other two if statements follow the same order but for the dark
and fire state
                digitalWrite(FIREPIN,LOW);
                digitalWrite(LIGHTPIN,LOW);
                digitalWrite(DARKPIN,HIGH);
        }
                if (State == 3) {
                digitalWrite(DARKPIN,LOW);
                digitalWrite(LIGHTPIN,LOW);
                digitalWrite(FIREPIN,HIGH);
        }
}
```

Whiskers.h

```
#ifndef Whiskers_h

#define Whiskers_h


#include "Arduino.h"

#include "Moves.h"


enum WHISKER {LEFTW, RIGHTW}; // an enum with the choices of the left whisker or the right whisker


boolean Whiskers(enum WHISKER whisker);


void WhiskerTurn();


#endif
```

Whiskers.cpp

```
#include "Whiskers.h"


boolean Whiskers(enum WHISKER whisker) { // This function telles you which one of the whiskers is
pressed, if they are pressed

        if (whisker == LEFTW ) { // This asks for the status of the left whisker

                return digitalRead(1); // This returns the value of the left whisker

        }

        if (whisker == RIGHTW) { // The same as above but for the right whisker

                return digitalRead(2);

        }

}
```

```
void WhiskerTurn() { // This function tells the robot what to do once a whisker is pressed
        if (Whiskers(LEFTW) == 0) { // if the left whisker is pressed
                Stop(); // stop whatever you were doing previously
                RightTurn(100); // turn right
                delay(250); // do that for a quarter of a second
                Forward(100); // go forward
        }
        else if (Whiskers(RIGHTW) == 0) { // this is the same as above but for the right whisker
                Stop();
                LeftTurn(100);
                delay(250);
                Forward(100);
        }
}
```

VI References

1. United States Census Bureau Results. Last Updated 17 July 2013

2. Gregory, K. “Rise in Complaints About Rats Prompts Call for New Eradication Program” *New York Times*. 20 Feb. 2013: A4. Print

3. Peeples, L. “Hurricane Sandy Could Displace Rats, Spread Infectious Disease” *Huffington Post.* HPMG News. Web.. 9 Oct. 2012.

4. Hanson, A. "What Do Rats See?“ Rat behavior and biology. 7 Dec. 2006. [18 Nov. 2013]. http://www.ratbehavior.org/history.htm

5. Kalloniatis, M.; Luu, C. “Visual Acuity” Webvision: The Organization of the Retina and Visual System. 5 June 2007 [18 Nov. 2013]. http://webvision.med.utah.edu/book/part-viii-gabac-receptors/visual-acuity/

6. Hanson, A. “The World Through a Rat’s Whiskers” Rat behavior and biology. 7 Dec. 2006. [18 Nov. 2013]. http://webvision.med.utah.edu/book/part-viii-gabac-receptors/visual-acuity/

7. Sayre, C. “Rats’ Whiskers Have Feelings, Too” *Time.* Time, Inc. Web. 27 Feb. 2008.

8. Watanabe, T.; Nakamore, T.; Murakami, N.; Morimoto, A. Suppresion of non-shivering thermogenesis in the rat by heat-seeking behaviour during cold exposure. *J. Physio.* 1986; 380: 541-549.

9. ArduinoCC. Arduino SA. 2013. Web. 25. Nov. 2013. http://arduino.cc/en/Main/arduinoBoardLeonardo

10. Parallax. Parallax Incorporated, 15 2012. Web. 17 Dec 2012. http://parallax.com