NYU-POLY

Firefighting Boe-Bot

Mechatronics Integrated Final Report

Carlos Fernandez, David Lopez, Andre Roman

Contents

Introduction

During fires in relatively large building, dousing the structure through windows and roof openings allows for control of the fire but remaining fires within the building may last. This requires firefighters to enter said structure, which may have become structurally unstable, in search of surviving fires which may endanger the firefighter due to lack of visibility and breathable air.

Removal of the human factor in task where the user may experience non-necessary danger has led to introduction of remote controller vehicles. These devices are capable of doing as the human user desires through a series of interactions between machine and user. The possibility for these robots to be controlled remotely allows the user to have eyes and ears in the field without the need to physically endangering him/herself. Firefighting has become a curious field where robotics is being applied, to the point where competitions are held for handheld robots to extinguish randomly positioned fires [1], as well as companies approaching the concept as an untapped robotic market [2].

The proposed robot design allows for sensing and dousing of small fires through manual means. The robot will be able to sense a fire from any position within its working range allowing for manual control of the robot by the user, who in turn is able to 'interact' with the environment as necessary while having constant feedback without the need of a computer screen.

Mechanical Design

The mechanical design of the system calls for a light and small system which will allow for ease in mobility. The proposed design incorporates components which are relatively simple to assemble while also being readily available.

Chassis

The working platform of the design will be the Boe-Bot provided by Parallax Inc (Fig. 1). Due to its simplicity in assembly and being readily available it will provide a sustainable chassis for the proposed system.

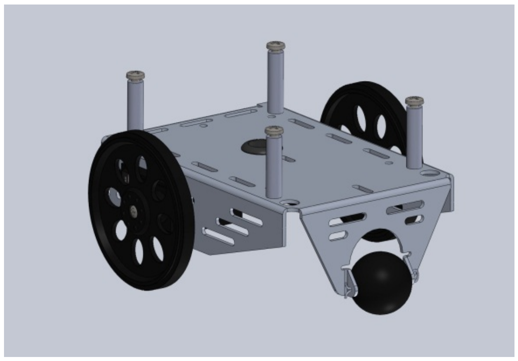


Figure 1 - Boe-Bot Platform

The chassis provided in the kit from Parallax Inc. is made from aluminum. It is specifically design to hold two continuous servos, which serve as the actuators for movement of the robot. The back wheel is a free moving rubber ball. The pegs seen in the figure, are mounted on the chassis, provide support for the Parallax Board of Education, which contains the Basic stamp 2 microcontroller.

Water Tank

The water tank is constructed from Plexiglas box of dimensions 5" x 3.25" x 2", sealed such as to avoid leaking of water into the electrical system. The tank is positioned over the electrical components due to the available supports provided by the Boe-Bot platform as well as the increased range of having the water being pumped from the highest position available, allowing for the robot to be farther from the fire.

Water Delivery

The water delivery is achieved using an Edmund Electronic miniature water pump placed inside the tank. The pumps working range being 1.5-3V allows for proper usage within the system. Water is sent through a tube whose free end is attached to a servomotor allowing for wider displacement of water by changing the angular position of the motor.

The following figure (Fig. 2) shows the assembly of the system water system incorporating the pump, servomotor and glass case as it's placed in top of the Boe-Bot.

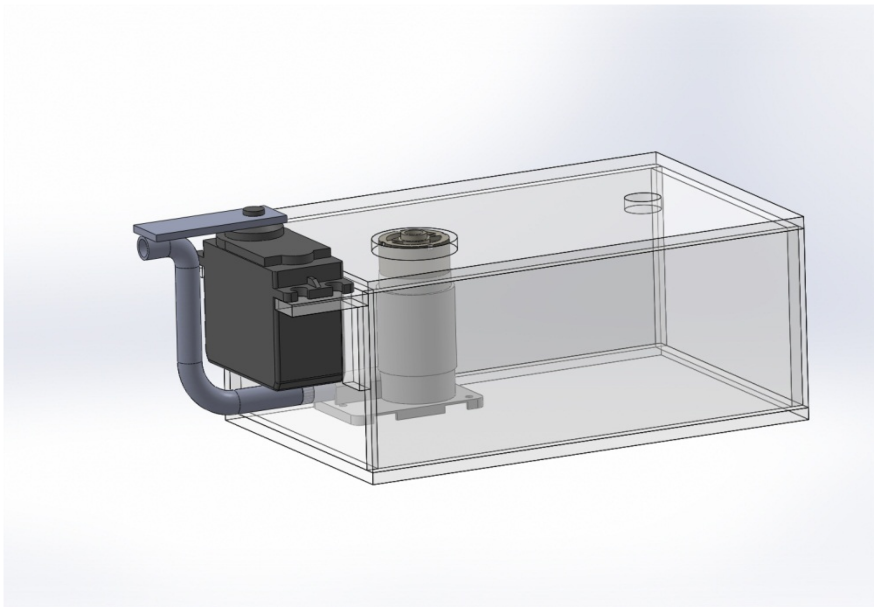
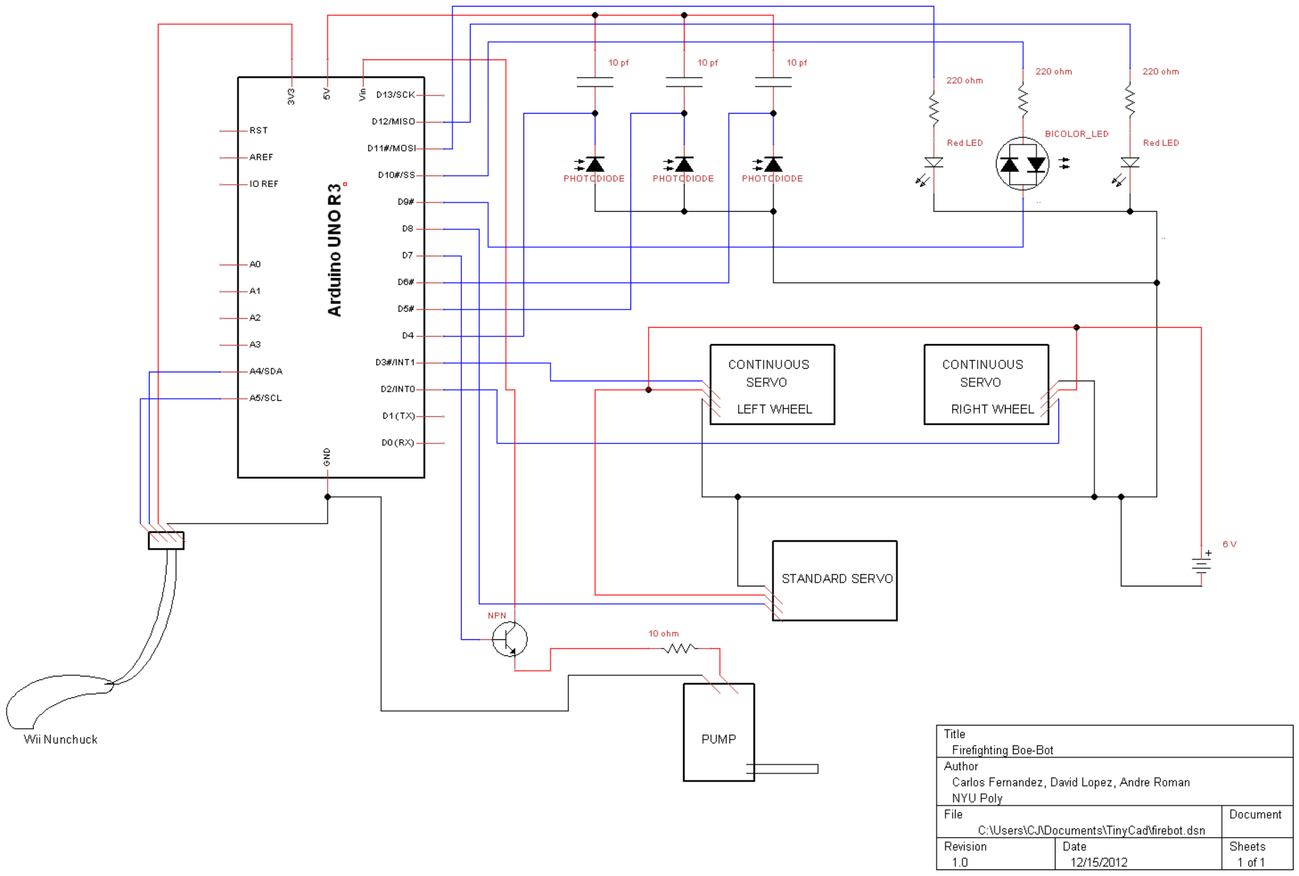


Figure 2 - Water Delivery System

Electrical Design

The electrical system proposed allows for proper interaction with the user through a tethered system. The user receives feedback from the system in the form of light signals through light emitting diodes (LEDs) to central hub from which the controller in connected. The overall system utilizes components available from the mechatronics class kit as well as extra components acquired to ease the interaction between the user and machine.

A general electrical schematic can be seen in Fig. 3.



Title Firefighting Boe-Bot	
Author Carlos Fernandez, David Lopez, Andre Roman NYU Poly	
File C:\Users\CJ\Documents\TinyCad\firebot.dsn	Document

Revision 1.0	Date 12/15/2012	Sheets 1 of 1

Figure 3 - Electrical Scheme

Microcontroller

The electrical system will be controlled by the Arduino UNO R3 microcontroller. The advantages of the Arduino microcontroller are its open source software, which allows for an extensive library for different interfaces. Due to certain equipment used, the Arduino is favored over the Basic Stamp 2 used during the class. The Arduino contains 14 digital I/O pins, which can provide 5 volts. Each pin can source/sink 40mA of current. Also, 6 of the pins have pulse width modulation (PWM). The Uno has 6 analog input pins, with 10 bit accuracy. These features provided by the Arduino platform were reasons why the Arduino was chosen over the Basic Stamp.

Wii Nunchuck

The human interaction component of the design will entail the use of the Wii Nunchuck. The Nunchuck uses a STMicroelectronics accelerometer to detect changes in position as well as the addition of a joystick and buttons for interaction.

The purpose of the Nunchuck in this design uses the accelerometer data to move the servomotor controlling the hose from the pump. The joystick allows for driving of the Boe-bot in any desired direction and a button is used as a trigger mechanism for the water pump.

Infrared LEDs

The electrical design of the system calls for the application of the duality of diodes as presented in class. A LED placed in reverse bias can act as a sensor to its expected output. For this project infrared LEDs will be used, which detect infrared light when a flame is present. The LEDs are placed in series with a capacitor such as to able to utilize the RCTime command to acquire a signal strength inversely relating to the distance from the source.

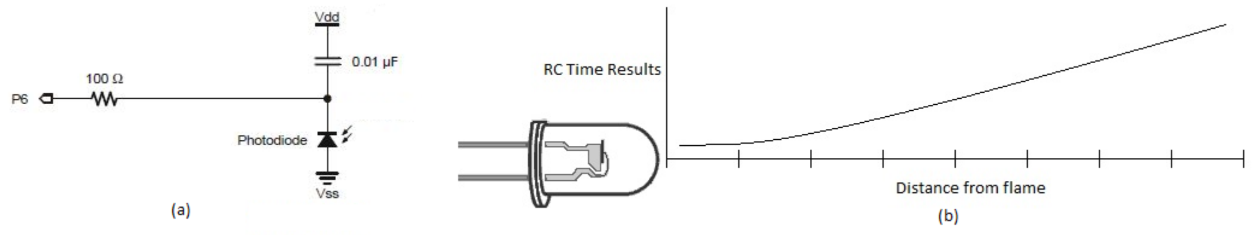


Figure 4 - (a) Photodiode in reverse bias, (b) Signal strength in relation to distance

Transistor

The PNP transistor utilized in this system allows for control of the water delivery system. Depending on the input supplied to the diode the pump will be activated or shut off. The following diagram graphically represents the above (Fig. 5).

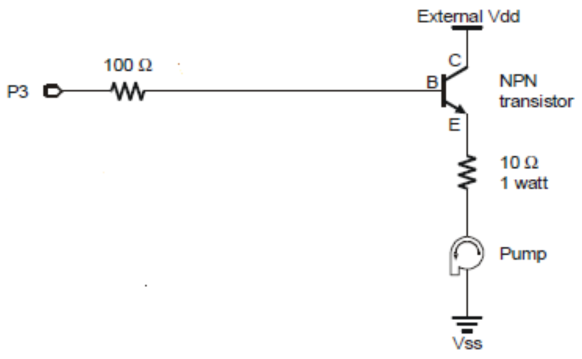


Figure 5 - Circuit diagram of pump and BJT

Servomotors

The Boe-Bot kit provides two continuous servomotors which are used to drive the vehicle as desired. An additional standard servomotor is used to position the hose as desired without the need to reposition the Boe-bot.

Indicators

Three LEDs will act as indicators to monitor the current status of the firefighting robot. The left and right LEDs are red, which will indicate that the fire is currently to the left or right of the boe-bot. The center LED is a bi-color LED (red/green), which will display red when the fire is sensed by the center infrared detector. Once the robot is in position, all three LEDs will be lit. When the fire is put out, green will be lit.

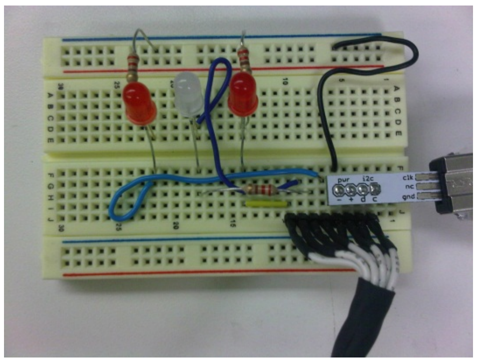


Figure 6: Indicators

Software Design

The software for the proposed project required a logic based system due to the implementation of several sensors to feedback into the user the relative position of the fire in relation to the front of the Boe-Bot. Note that the software accounts for proper allocation of signals into the following hub, where the controller is connected.

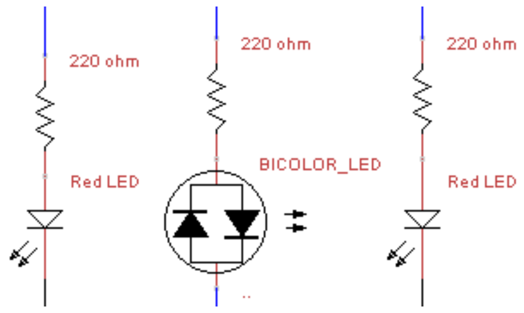
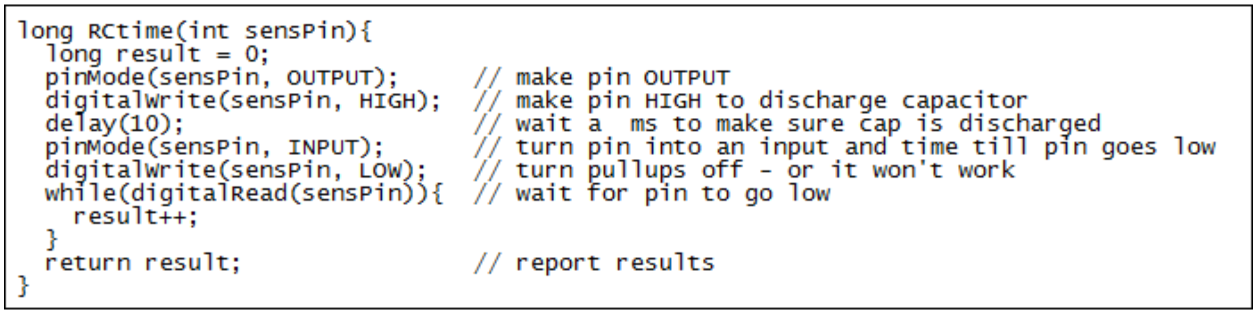


Figure 7 - Signal Plant for User Interaction

The following are excerpts of the code which are of interest. Initially it should be noted that there is lack of the RCTime function within the Arduino system as such the function is defined as follows (Fig. 7). The function set the pin as initially HIGH such as to allow for discharge of the capacitor into ground, after which the pin is set as input and defined LOW during which the results is continuously increasing until the pin physically LOW.



```
long RCtime(int sensPin){
  long result = 0;
  pinMode(sensPin, OUTPUT);      // make pin OUTPUT
  digitalWrite(sensPin, HIGH);   // make pin HIGH to discharge capacitor
  delay(10);                     // wait a  ms to make sure cap is discharged
  pinMode(sensPin, INPUT);       // turn pin into an input and time till pin goes low
  digitalWrite(sensPin, LOW);    // turn pullups off - or it won't work
  while(digitalRead(sensPin)){   // wait for pin to go low
    result++;
  }
  return result;                 // report results
}
```

With the ability to acquire data relating to the fire the following code is a breakdown of the logic used to tell the user where the fire is, which is done by the flashing of LEDs in a respective manner (recall Fig.6)

The following is a snippet of code that we used to make the boe-bot autonomous. There were problems implementing autonomous control because the values of RCtime varied greatly with the amount of light the fire was emitting. Precision was not obtainable with the current setup.

```
void lightLEDs() {
  if (leftSensor < 5000 || centerSensor < 5000 || rightSensor < 5000) {
    if (leftSensor == min(leftSensor, min(centerSensor, rightSensor))) {
        // Fire detected closer to left sensor, light left LED
    }
    else if (rightSensor == min(leftSensor, min(centerSensor, rightSensor))){
        // Fire detected closer to right sensor, light right LED
    }
    else {
      if (centerSensor > 500) {
        // Fire in front of Boe-Bot, Bi-color LED (red)
      }
      else {
        // Fire in front and within pump range, red LEDs lit, Bi-color LED (red)
      }
    }
  }
  else {
        // No fire detected in front, Bi-color LED flashing (green)
  }
}
```

Accounting that the user know knows where the fire is s/he may properly react by using the Wii Nunchuck accordingly. The data is read and set as the right constants to denote direction of travel.

```
void getDir() {
  const int ymax = 220;
  const int center = 125;
  const int ymin = 30;
  spd = map(nunchuk.analogY, ymin, ymax, 1450, 1550);
  const int xmax = 227;
  const int xcenter = 127;
  const int xmin = 27;
  turn = map(nunchuk.analogX, xmin, xmax, -15, 15);
}
```

The microcontroller then applies the defined system constant for that instant into the respective tires, applying motion to the Boe-Bot.

```
void drive(int spd, int turn) {
  int rightSpeed = spd;
  int leftSpeed = map(spd, 1450, 1550, 1550, 1450);
  rightWheel.writeMicroseconds(rightSpeed + turn);
  leftWheel.writeMicroseconds(leftSpeed + turn);
}
```

The following represents the main loop of the system in logical succession; sensors are checked, which feeds data to the user through LEDs, followed by the checking Nunchuck state, depending the state of the z-button turn pump on finally moving the Boe-Bot if required by Nunchuck input by defining turning and speed then applying to servos.

```
void loop() {
  updateSensors();
  lightLEDs();
  nunchuk.update();
  waterPump.writeMicroseconds(map(nunchuk.accelX, 300, 700, 1275, 1725));
  digitalWrite(pumpPin, nunchuk.zButton);
  getDir();
  drive(spd, turn);
}
```

Complete code is available at the end of the report.

<u>Final Prototype</u>

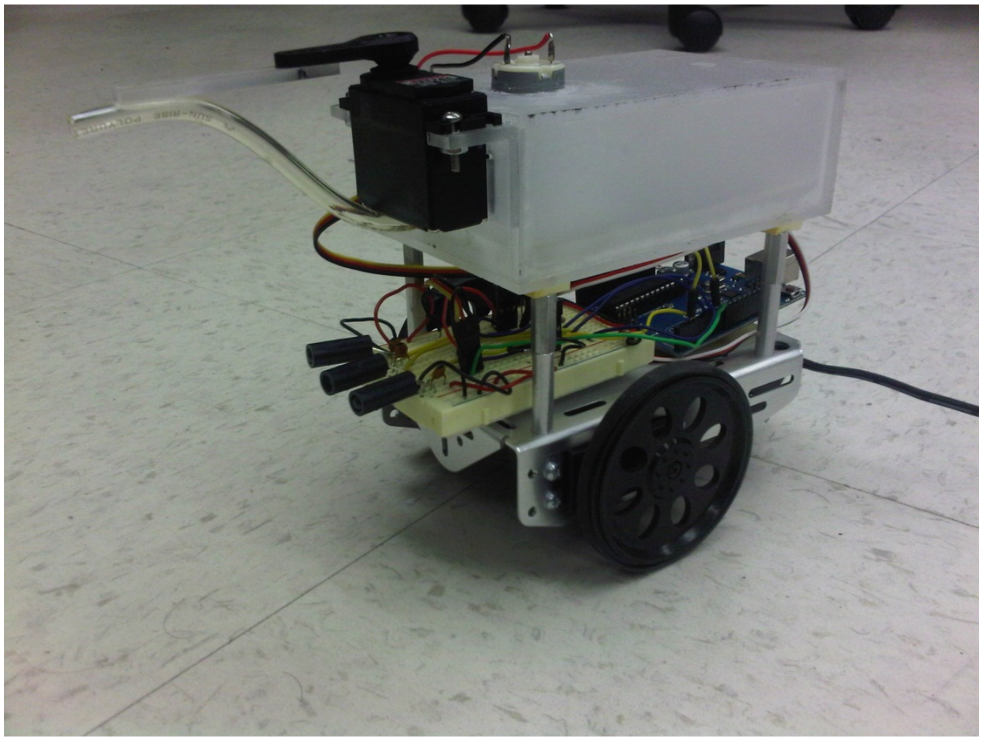


Figure 8: Prototype

Cost Analysis

Table 1 - Parts Cost

Item	Description	Vendor	Part No.	Quantity	Price per	Prototype	Production
1	Boe-Bot Robot Kit	Parallax Inc.	28132	1	$81.99	$81.99	$65.59
	1/16" cotter pin		700-00023				
	1" tail wheel		700-00009				
	(4) rubber band tires		721-00002				
	(2) plastic wheels		721-00001				
	(8) 3/8" 4-40 pan head screws		700-00002				
	(2) 4-40 flathead screws		700-00016				
	(8) 7/8" 4-40 pan head screws		700-00028				
	(2) 7/8" 4-40 pan head screws		710-00007				
	(10) 4-40 zinc-plated nuts		700-00003				
	(4) 1" round 4-40 standoff		700-00060				
	(2) spacer, 1/2" round		713-00007				
	13/32" rubber grommet		700-00025				
	(2) nylon washers (screw size #4)		700-00015				
	(2) 3-pin headers		451-00303				
	(2) Parallax continous rotation servos		900-00008				
	(2) infrared LEDs		350-00003				
	(2) LED standoffs		350-90000				
	(2) LED shields for 350-90000		350-90001				
	Jumper wires (2 bags of 10)		800-00016				
	Aluminum chassis		700-00022				
	battery holder with cable and barrel plug		700-00038				
2	Arduino Uno R3	SmartProjects		1	$25.00	$25.00	$25.00
3	infrared LEDs	Parallax Inc.	350-00003	2	$0.99	$1.98	$1.52
4	Wii Nunchuk Controller	Nintendo		1	$20.00	$20.00	$20.00
5	WiiChuck Adapter	Sparkfun Electronics	DEV-09281	1	$1.95	$1.95	$1.56
6	10 pf Capacitors	Mouser Electronics	140-500N2-100J-RC	3	$0.10	$0.30	$0.26
7	Breadboard	Sparkfun Electronics	PRT-09567	2	$5.59	$11.18	$9.52
8	Plexiglass Case	Proffesional Plastics		1	$20.00	$20.00	$20.00
9	Standard Servo	Parallax Inc.	900-00005	1	$12.99	$12.99	$11.69
10	Pump	Lightobject	EXP-7L9	1	$8.95	$8.95	$6.70
11	Resistor Kit - 1/4 W	Sparkfun Electronics	COM-10969	1	$7.95	$7.95	$6.36
12	Break Away Headers	Sparkfun Electronics	PRT-00116	1	$1.50	$1.50	$1.20
13	Red LED	Parallax Inc.	350-00006	2	$0.50	$1.00	$0.80
14	Bi-color LED	Parallax Inc.	350-00005	1	$0.99	$0.99	$0.79
	TOTAL					**$195.78**	**$170.99**

The price of the Firefighting Boe-Bot is finalized at $ 195.78 for the prototype with a mass production price estimated at $ 170.99 acquired by looking into mass production of 100 units at once. A price drop of 12% is expected for mass production, the number might increase if contact is done with major manufacturers of the major pieces such as Nintendo and SmartProjects for the Wii Nunchuck and Arduino, respectively.

Safety Operation

The Firefighting Boe-Bot allows for remote extinguishing of fires, placing the user in a harm free environment. Therefore safety parameters are only applied to the robot and care should be taken when using it due to its proximity to fire and water.

The signals sent back to the user through the LEDs should be taken seriously when proximity to the fire is detected, as the infrared sensors lead the Boe-Bot and therefore damage can be done to them in which case operation should be stopped. The possibility of water damage during loading is also a real possibility due to the exposed electronics under the tank and any possible leaks should be addressed. Finally it should also be taken into account that at low water levels the Boe-Bot may not overcome the sensors under the hose especially while in motion therefore the tank should be fully filled before operation.

Possible Design Improvements

The design allows for further improvement if desired for expansion into a more complex system. Improvements can be done in the water delivery system as well as the sensing system of fires. General improvements could include water proofing of electronics while also fire proofing certain components.

A particular limitation proposed system is the lack of ability to detect fires not in direct eye level with the sensors, particularly at a higher level. To overcome such case additional infrared sensors can be included at different heights allowing for higher range. An addition of a temperature sensor could also be incorporated to better determine the distance from a fire or heat source as the current design uses plastic, high heat environments could damage electronics.

The limited water supply due to the dimensions of the tank could be overcame if a tethered design is preferred in which case a direct line to a greater water source is achieved. An addition of a designed nozzle can also allow for better water delivery, optimizing the surface area of water applied.

Conclusion

The Firefighting Boe-Bot was successful in putting out fires through the aid of a user remotely controlling the robot. The robot is able to be maneuvered using a Wii Nunchuck through tether while also sending signals back to the user to central hub from which the Nunchuck can be attached or detached as needed. System expansion to improve the design is possible including the original desire for an autonomous mode when the system is detached from the tether. Ultimately the project was successful and project guidelines were met.

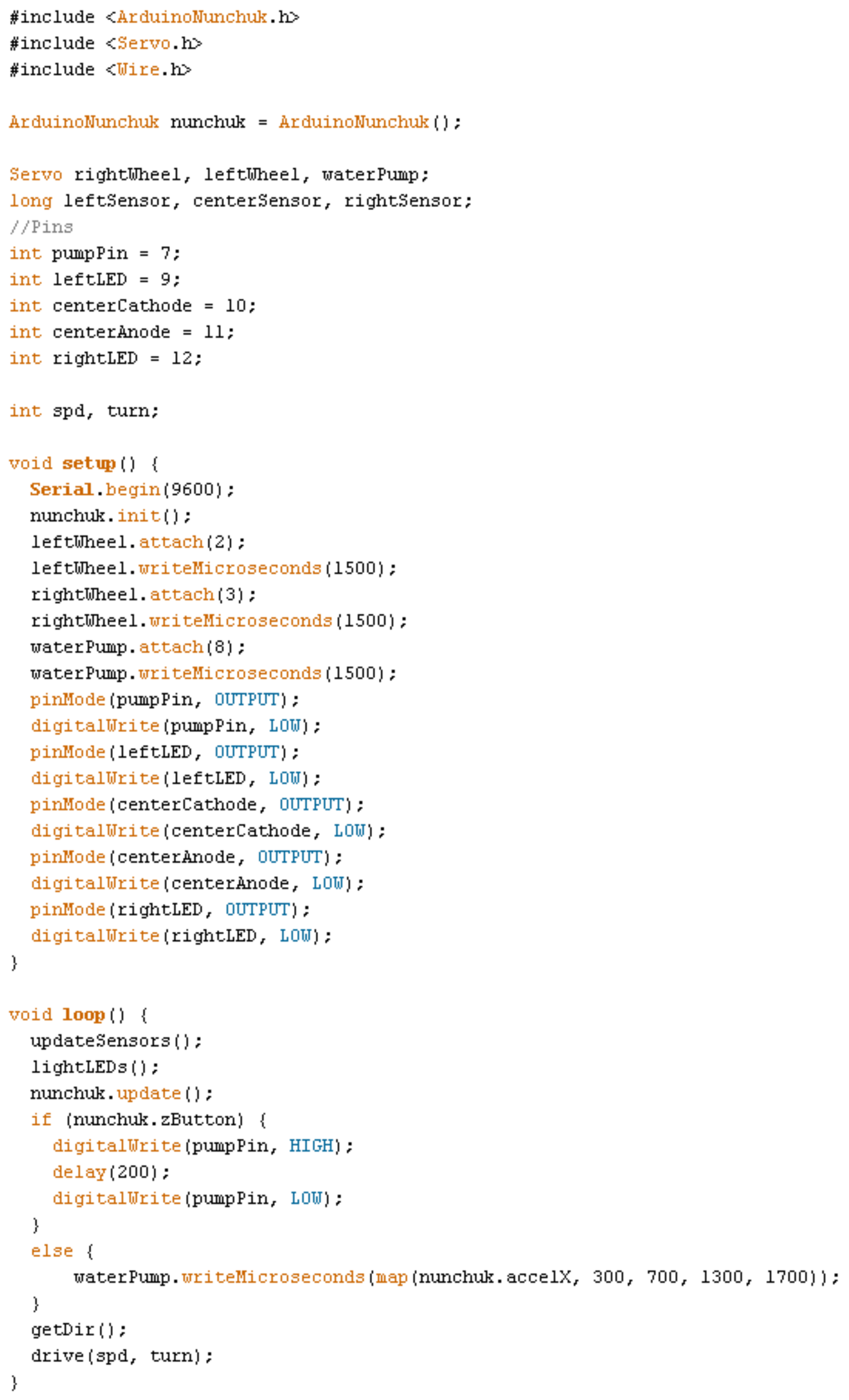
References

[1] - <http://www.trincoll.edu/events/robot/>

[2] - <http://www.wired.com/dangerroom/2012/10/fire-fighting-robots/>

Code

Initialization of variables, setup, and main loop



```
#include <ArduinoNunchuk.h>
#include <Servo.h>
#include <Wire.h>

ArduinoNunchuk nunchuk = ArduinoNunchuk();

Servo rightWheel, leftWheel, waterPump;
long leftSensor, centerSensor, rightSensor;
//Pins
int pumpPin = 7;
int leftLED = 9;
int centerCathode = 10;
int centerAnode = 11;
int rightLED = 12;

int spd, turn;

void setup() {
  Serial.begin(9600);
  nunchuk.init();
  leftWheel.attach(2);
  leftWheel.writeMicroseconds(1500);
  rightWheel.attach(3);
  rightWheel.writeMicroseconds(1500);
  waterPump.attach(8);
  waterPump.writeMicroseconds(1500);
  pinMode(pumpPin, OUTPUT);
  digitalWrite(pumpPin, LOW);
  pinMode(leftLED, OUTPUT);
  digitalWrite(leftLED, LOW);
  pinMode(centerCathode, OUTPUT);
  digitalWrite(centerCathode, LOW);
  pinMode(centerAnode, OUTPUT);
  digitalWrite(centerAnode, LOW);
  pinMode(rightLED, OUTPUT);
  digitalWrite(rightLED, LOW);
}

void loop() {
  updateSensors();
  lightLEDs();
  nunchuk.update();
  if (nunchuk.zButton) {
    digitalWrite(pumpPin, HIGH);
    delay(200);
    digitalWrite(pumpPin, LOW);
  }
  else {
      waterPump.writeMicroseconds(map(nunchuk.accelX, 300, 700, 1300, 1700));
  }
  getDir();
  drive(spd, turn);
}
```

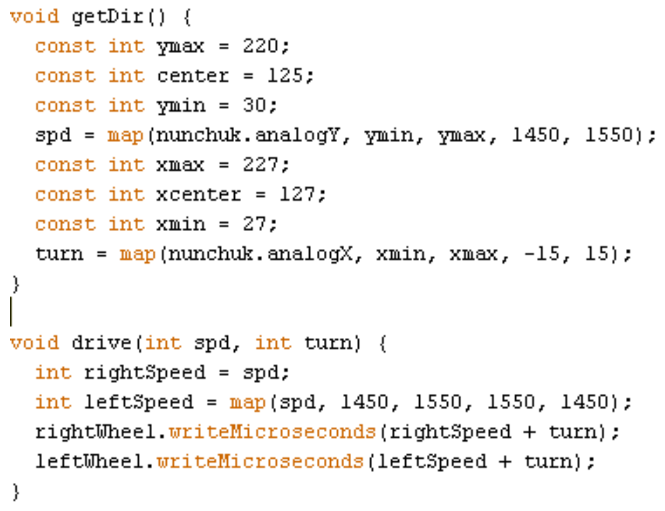
Update sensors, RCtime:

```
void updateSensors() {                                                  // obtain values for Infrared sensors
  leftSensor = RCtime(4);                                               // get RCtime value for left IR sensor (LED)
  centerSensor = RCtime(5);                                             // get RCtime value for center IR sensor (LED)
  rightSensor = RCtime(6);                                              // get RCtime value for right IR sensor (LED)
  Serial.print(leftSensor);                                             // Print left IR RCtime value to serial monitor
  Serial.print("\t\t");
  Serial.print(centerSensor);                                           // Print center IR RCtime value to serial monitor
  Serial.print("\t\t");
  Serial.println(rightSensor);                                          // Print right IR RCtime value to serial monitor
}

long RCtime(int sensPin){
  long result = 0;
  pinMode(sensPin, OUTPUT);       // make pin OUTPUT
  digitalWrite(sensPin, HIGH);    // make pin HIGH to discharge capacitor - study the schematic
  delay(10);                       // wait a  ms to make sure cap is discharged

  pinMode(sensPin, INPUT);        // turn pin into an input and time till pin goes low
  digitalWrite(sensPin, LOW);     // turn pullups off - or it won't work
  while(digitalRead(sensPin)){    // wait for pin to go low
    result++;
    if (result > 5000) {
      break;
    }
  }
  return result;                   // report results
}
```

Light LED indicators

```
void lightLEDs() {
  if (leftSensor < 5000 || centerSensor < 5000 || rightSensor < 5000) {
    if (leftSensor == min(leftSensor, min(centerSensor, rightSensor))) {
      digitalWrite(leftLED, HIGH);
      digitalWrite(centerCathode, LOW);
      digitalWrite(centerAnode, LOW);
      digitalWrite(rightLED, LOW);
    }
    else if (rightSensor == min(leftSensor, min(centerSensor, rightSensor))){
      digitalWrite(leftLED, LOW);
      digitalWrite(centerCathode, LOW);
      digitalWrite(centerAnode, LOW);
      digitalWrite(rightLED, HIGH);
    }
    else {
      if (centerSensor > 350) {
        digitalWrite(leftLED, LOW);
        digitalWrite(centerCathode, HIGH);
        digitalWrite(centerAnode, LOW);
        digitalWrite(rightLED, LOW);
      }
      else {
        digitalWrite(leftLED, HIGH);
        digitalWrite(centerCathode, HIGH);
        digitalWrite(centerAnode, LOW);
        digitalWrite(rightLED, HIGH);
      }
    }
  }
  else {
    digitalWrite(leftLED, LOW);
    digitalWrite(centerCathode, LOW);
    digitalWrite(centerAnode, HIGH);
    digitalWrite(rightLED, LOW);
  }
}
```

getDir() and drive



```
void getDir() {
  const int ymax = 220;
  const int center = 125;
  const int ymin = 30;
  spd = map(nunchuk.analogY, ymin, ymax, 1450, 1550);
  const int xmax = 227;
  const int xcenter = 127;
  const int xmin = 27;
  turn = map(nunchuk.analogX, xmin, xmax, -15, 15);
}

void drive(int spd, int turn) {
  int rightSpeed = spd;
  int leftSpeed = map(spd, 1450, 1550, 1550, 1450);
  rightWheel.writeMicroseconds(rightSpeed + turn);
  leftWheel.writeMicroseconds(leftSpeed + turn);
}
```