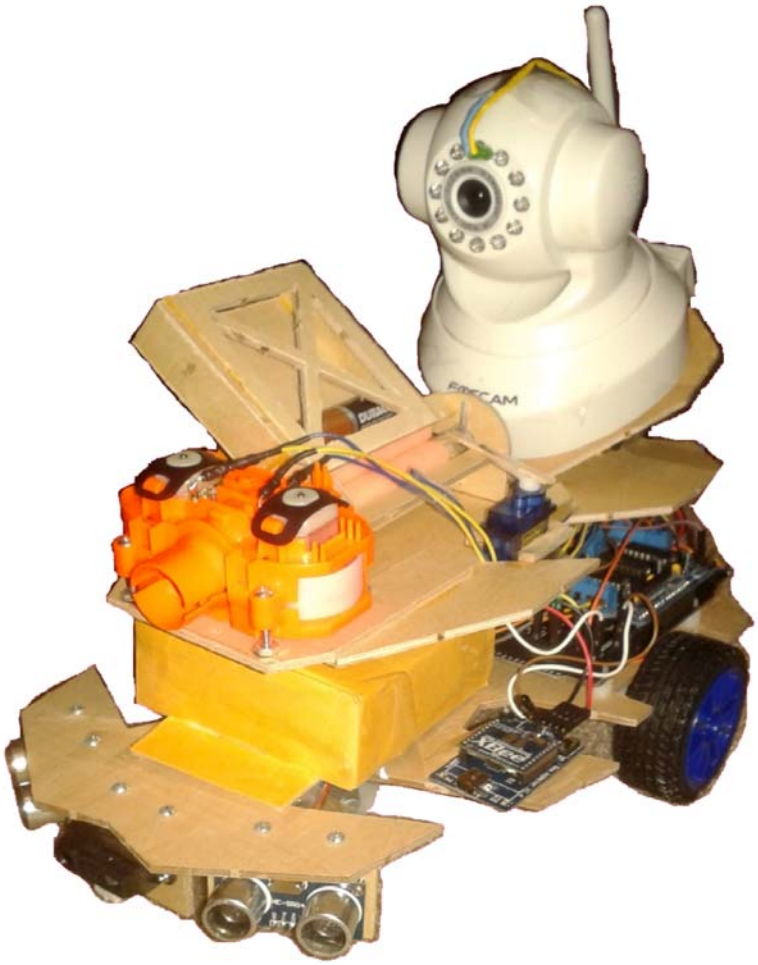
Night Watcher

Yanming Wang
January 28, 2014



University of Florida
Department of Electrical and Computer Engineering
EEL 5666C – IMDL –Written Report
Instructors: A. Antonio Arroyo, Eric M. Schwartz
TAs: Andy Gray, Josh Weaver, Nick Cox, Daniel Frank

Table of Contents

Ⅰ.Abstract

Night Watcher is an autonomous mobile robot designed to search, aim, and drive away small animals get into people's garden or camping site. Night Watcher will start to work when night descends. It will move around searching for circles using a camera with night vision. Once it has found a target, it will use its special range finder to measure the distance between them. If the distance is too short, Night Watcher will drive away the target by shooting at it. All these functions are accomplished by using multiple sensors and programming languages like OpenCV.

Ⅱ.Executive Summary

Night Watcher is designed to detect small animals in people's yards. It can move around the yard and avoid obstacles on its way with two ultrasonic range finders and an infrared proximity sensor. The IP camera is mounted on the turret and the turret will swing around to search for circles. The video is sent to the laptop and the OpenCV program in the laptop will detect circles using Hough circles method. When a circle is found, the laptop will send a command to the robot via XBee. Then the robot will stop and turn its turret to aim at the target. When the circle is in the middle of the screen, the laptop will measure the distance between them and tell if the target is out of the shooting range or not then send a command to tell robot to shoot. However, due to the low resolution of the camera and the powerful launching mechanism, the shooting range is much farther than the detecting range. Therefore, the gun can always reach the target which can be detected by the OpenCV program.

Ⅲ.Introduction

People who live near forests and camp in the wild sometimes need to share their yard with wild animals. Sometimes these animals are cute and interesting, but there are some time people want to live without disturbance. Especially at night, the unwelcomed little guests can be annoying. Moreover, getting too close to wild animals can be dangerous, as many diseases can transfer between human and animals. Disease like Severe Acute Respiratory Syndromes (SARS) comes from wild animal and killed many people in China. Therefore, this three-wheeled robot is designed to solve this problem. With Sonar and IR sensors, it can avoid bumping into any obstacle. It has a specially designed range finder using laser and camera to measure the distance between it and threats. Xbee is used to send commands from personal laptop. The camera is used to find targets based on their shapes with the help of OpenCV. Other functions like tracking and follow any moving objects using the camera, or use other sensors to detect warm blood animals that get into your yard may be applied after this course.

Ⅳ.Integrated System

The system uses an IP camera with night vision to catch images and communicate with laptop using the built in wireless connection module in the camera. The pictures are sent to the computer and then analyzed via an OpenCV program. The program finds circles which is defined as target and tells the coordinates of its core. Then a command is sent to the robot using Xbee, and the robot will turn its turret to aim at the core by putting it in the middle of the image. After that, special range finder works to measure the distance. If the distance is too short, the camera will take a picture of the invader, and then the weapon system will fire at the target try to drive it away. The robot can also move around the yard using information gathered by sensors to avoid collision. The turret swings around to search targets. When a target is spotted, the robot stops and begins to aim.

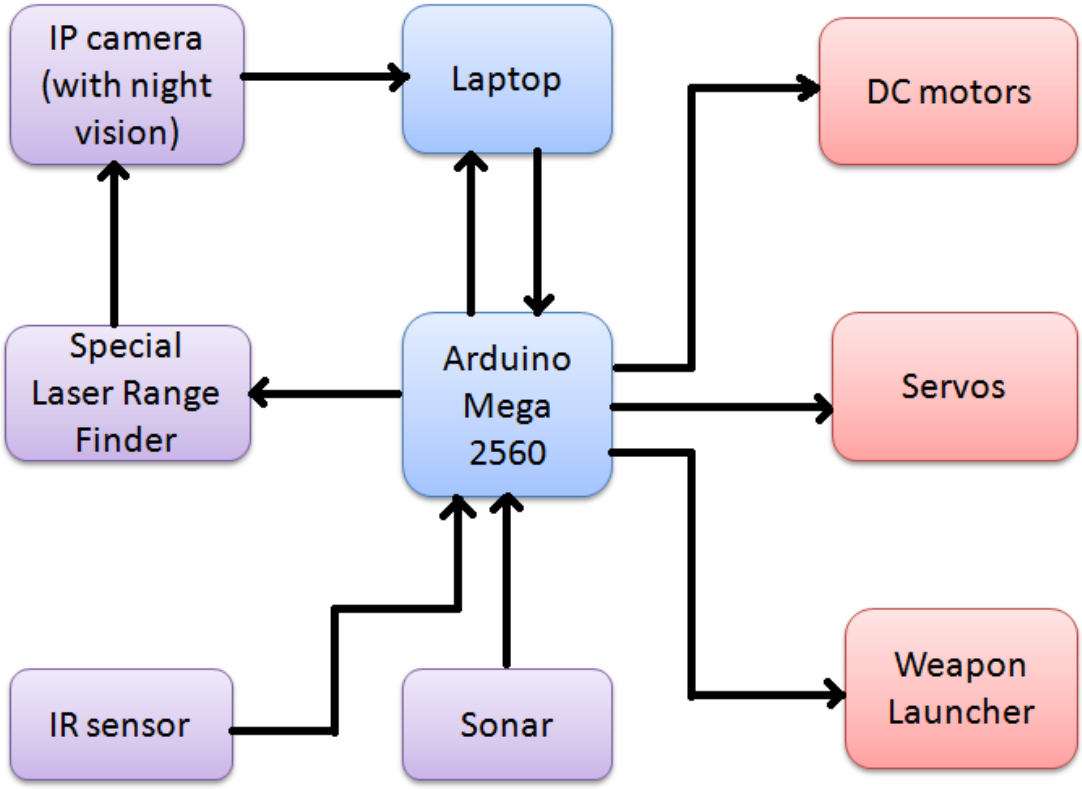


Fig 1. Functional block diagram

Ⅴ.Mobile Platform

The mobile platform will carry all the modules of the robot and drive around the yard. It has Three wheels and two DC motors. This platform is consisted with two levels. The bottom level is the power and processor level. The motors and wheels are mounted under this level. On this level, battery, Arduino board, servo for the turret and sensors for collision detection are fixed on it. The upper lever is the turret. IP camera, special range finder and weapon system are attached there. They will move

with the turret.

The bottom and upper levels are as shown below:

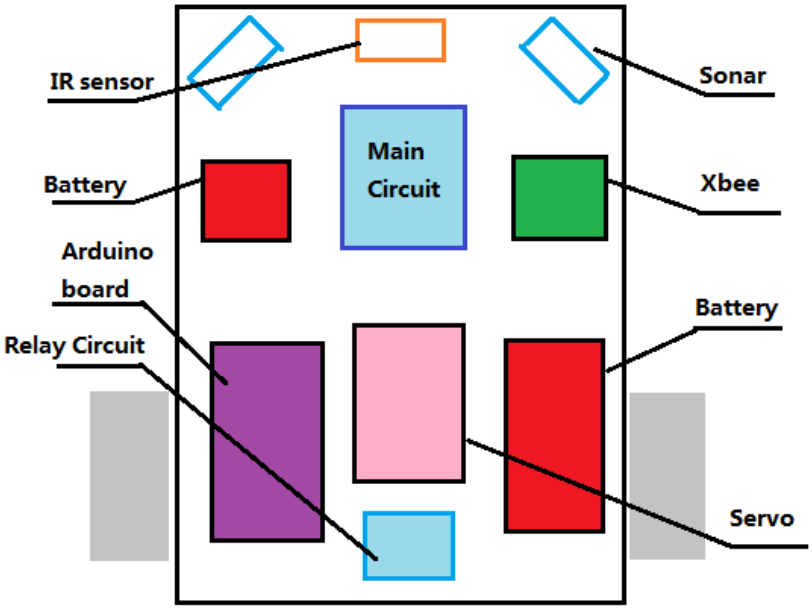


Fig 2. Bottom Level

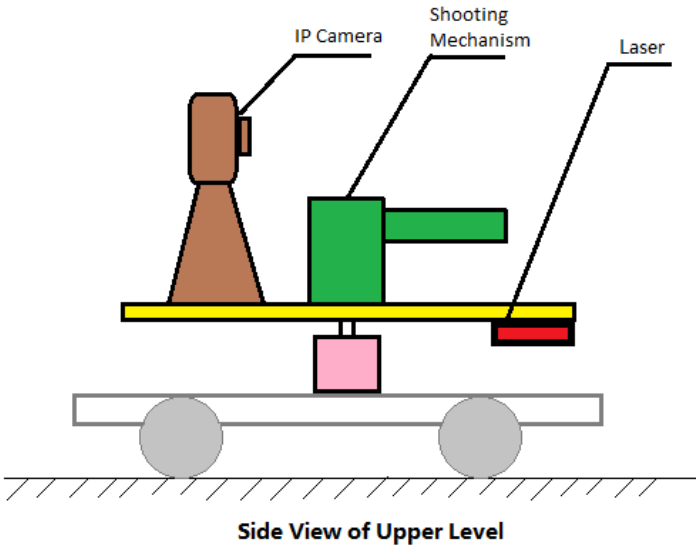


Fig 3. Side View of upper level

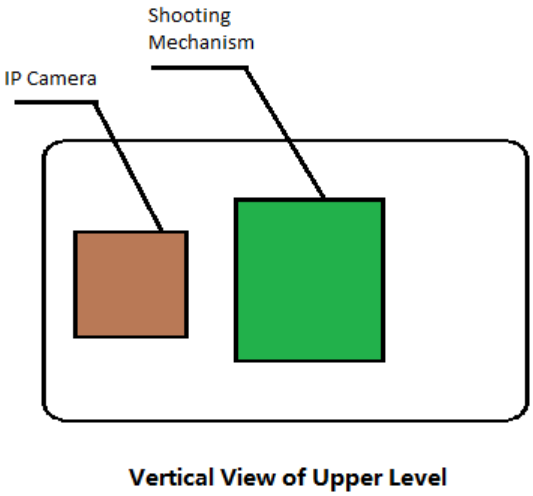


Fig 4. Vertical View of upper level

As the robot is designed to work outside the house, the motors are mounted on the rear end of the robot and beneath the chassis to lift up the robot. The center of gravity is put near the motors so the wheels can get more traction. The axis of the turret is also put at the rear end to get enough room for the shooting mechanism and the auto loading mechanism.

Due to the limited scale of wood board T-tech can cut out, the size of the chassis is limited and is crowded with all parts on it. And the wires have to be fixed on the board so they will not interfere with the movement of the turret.

Ⅵ.Actuation

Night Watcher uses three wheels to support the chassis, and use two-wheel drive via two DC motors. The motors are controlled directly by Arduino board. For the rotation of the turret, a servo is used. There are two servo motors. One is used to turn the turret, and the other works as a reload system for the shooting mechanism. Two small DC motors are used to launch sponge bullets in the shooting mechanism.

DC Motors

Fig 5. ASLONG JGA25-371 DC motor

Two ASLONG JGA25-371 DC motors are used to drive the robot. These motors have encoders on them. I wrote some code to use the encoder get the speed of the wheels but did not put those programs in the final program. The reduction gear ratio is selected based on the torque and moving speed required for the robot. The total weight of the robot was estimated to be 2kg and the moving speed is close to human moving speed. Based on the tutorial on Pololu [7] I chose the reduction gear ratio to be 78 and the stall torque is 9.7 kg·cm.

Servos

Fig 6. Pololu servo

Two servos are used in this robot. Both are ordinary servos. The servo with higher torque is used to turn the turret as there are many parts on the turret. The smaller servo works as the rammer for the loader.

There was an issue with the turret servo. At first, I bought a continuous rotation servo as it has the highest torque I can get, but found out I cannot control it to rotate to desired angle. So I switched to an ordinary one and it worked fine.

Shooting Mechanism

The shooting mechanism uses two rotating wheels to squeeze and push a cylindrical sponge as bullet. The mechanism is shown below. The wheels will be driven by two motors separately. Due to lacking of proper motors, this shooting mechanism is taken from a nerf gun.

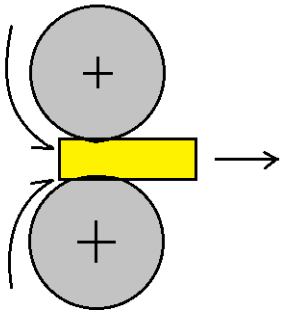


Fig 7. Shooting Mechanism

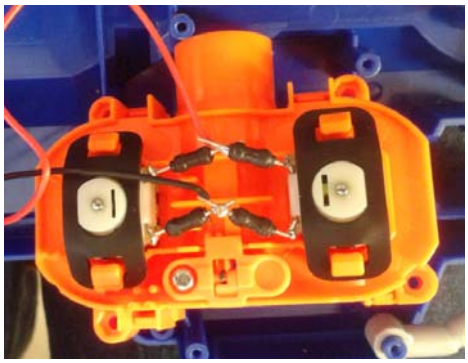


Fig 8. Launching system in nerf gun

It has a clip to hold nerf darts and an AA battery is used to force the darts go downward to a track. The servo attached with a rammer pushes the nerf dart into the shooting mechanism to launch it.

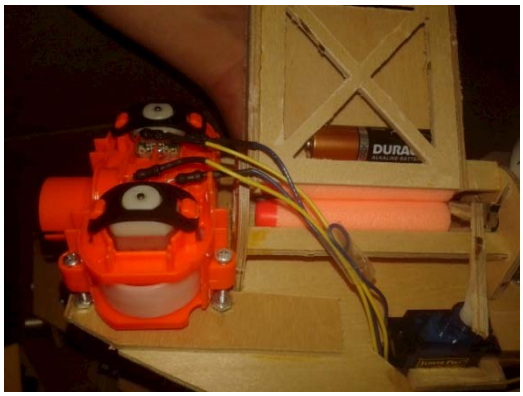


Fig 9. Gun system on the turret with auto loader and magazine

The most painful part of building up the actuators is the power supply. The servos works fine when the arduino board is powered through USB cable from my laptop. But when I use the battery to power the arduino board then power the servos by arduino, the servos just refused to work properly. It took me some time to figure out that it was because the arduino board cannot give enough power to the servos when the board is powered by battery. So I shared the power line of the camera which is 5V with my two servos. And that solved the problem.

Ⅶ.Sensors

Night Watcher has two ultrasonic range finders and an IR sensor for obstacle detection. And two switches are mounted behind the front bumper. They will help the robot to stop if it bumps into something. There is a special range finder. This range finder is consisted with one laser source, and an IP camera.

Special sensor

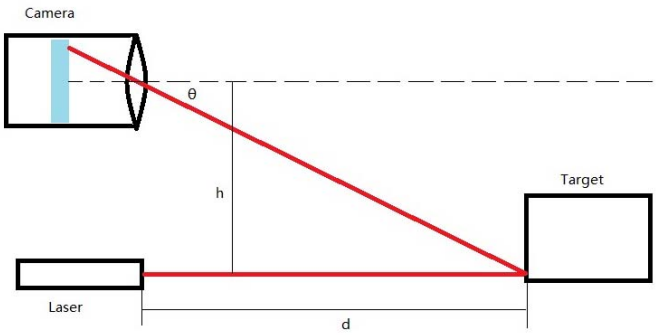


Fig 10. Design of the special sensor

The center line of the camera should be parallel to laser beam. When the sensor is told to measure distance, the laser pointer will be turned on. After that, a laser dot can be read by the IP camera. As is shown in the figure above, the y position of the laser dot will change with different angle θ. And the distance between the rangefinder and the target determines the angle θ. Therefore, by reading the position of the laser dot, we can calculate the distance. The equation is as shown below. This equation is referred from a Chinese website [2]. On the website, the author happens to make the same thing as I am doing.

$$d = \frac{h}{\tan\left(NumP * \mathrm{Rpp} + \mathrm{ERR}\right)}$$

In the equation, d is the distance between rangefinder and the laser dot, h is the vertical distance between laser pointer and camera. NumP is the y direction pixel number between laser dot and the center of the frame, Rpp is radian per pixel. ERR is the error when carrying out the calibration.
The system should be calibrated to get Rpp and ERR for each camera. By putting targets at certain distance we already known, we can get Rpp and ERR by solving the equations.

The OpenCV program used for detecting the laser dot is based on color detection. The program code is modified from a tutorial on website [1]. And to pick up the laser color I used software named ColorPic to get the value of colors on my screen.

The most advantage of this sensor compared to ultrasonic and IR sensor is that it can get the distance of a point instead of any obstacles appear in the searching range. And it also has disadvantage that it relies greatly on the camera resolution and it will remain accurate only in a certain range. Because when the object is too near, the camera won't catch the laser dot and when the object is too far away, the variation of the laser coordinate will be too small to drive an accurate answer. My rangefinder can get the distance in a range of 0.4 ~ 2 m.

I met with some problems while setting up the laser. The laser source uses

3V power supply, but the arduino board can only supply 3.3V. Moreover, the 3.3V pin is taken by the motor shield. So I have to use two batteries to power the laser. I also want to control when to turn on the laser. Therefore, a relay circuit is used. The circuit is as shown below, which comes from Arduino forum:

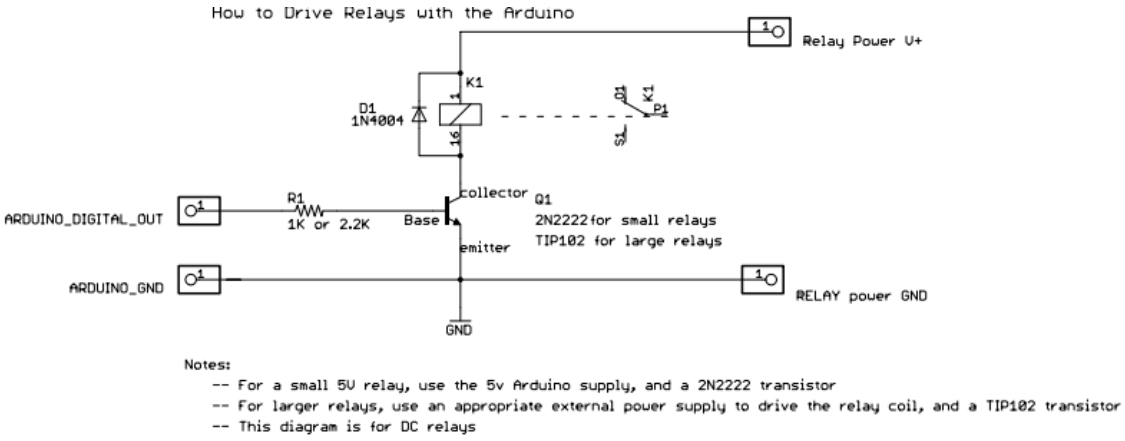


Fig 11. Relay circuit for Arduino

Due to poor soldering skill, I soldered and re-soldered the circuit several times to get it work. It is important to test every pin of the circuit with a multi meter before power it.

Ⅷ.Behaviors

The main logic map is shown below.

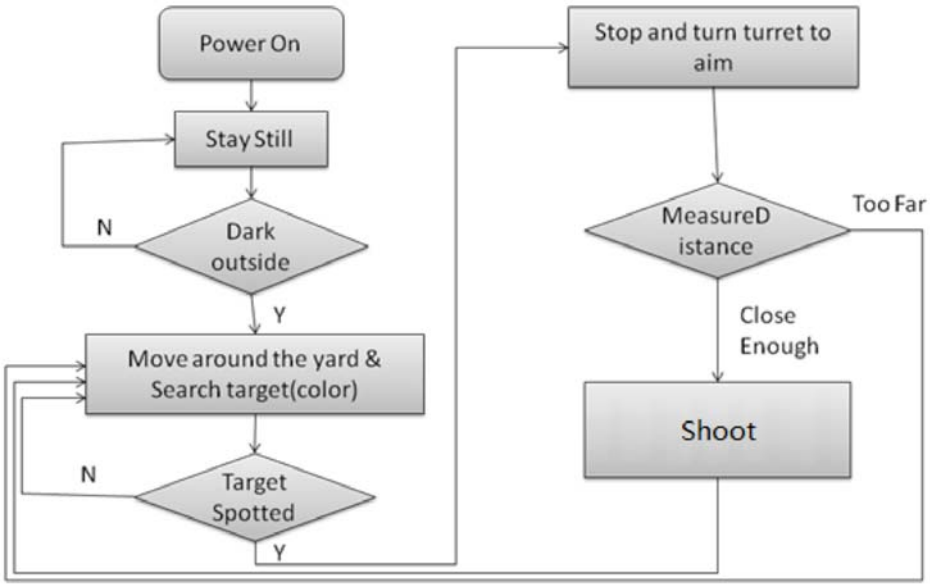


Fig 12. Action loop

When power on, the robot measures the light strength to tell day and night. If it is night, the camera will switch on night vision mode. As it moves, it will swing its turret to look for targets. If it finds a target, it will stop moving, and camera will send pictures back to laptop. The laptop analyses the picture and tell the robot to turn its turret to left or right. After turning the turret, the special range finder begins to function. The range finder tells the distance between the target and the robot. If the distance is too short, the robot will shoot the target. Or it will just keep doing patrol duty and leave that target alone. However the detecting range of the camera is shorter than the shooting range, the robot will always be able to shoot. Xbee is used to send command from the laptop to the robot and sensor values from the robot to the laptop.

Ⅸ.Experimental Layout and Results

Most of the experiment is about my special laser rangefinder. The original idea was to use the CDs cell to get the laser dot. And the experimental layout is as shown follows.

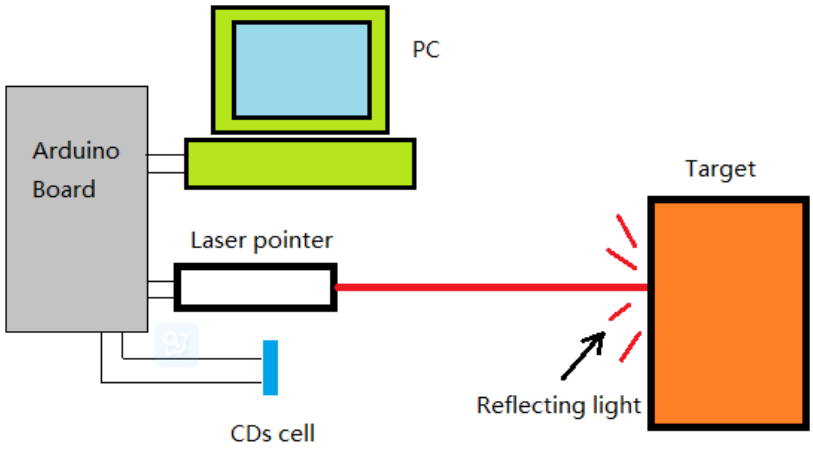


Fig 13. Experimental setting for original idea of special range finder

Arduino Board is used to collect the signal from CDs cell and show it on PC screen.

This experiment is to test whether I can use CDs cell to detect the reflecting light of the laser dot accurately. However, the result came out that the data collected from the CDs cell will not vary a lot even the experiment is done in complete darkness and the target is really close to the CDs cell.

After that, I modified the rangefinder design and did another experiment. This experiment is to test if the position of the laser dot will vary at different distance. In this experiment, the center line of the camera and the laser beam are parallel to each other. The layout is shown below.

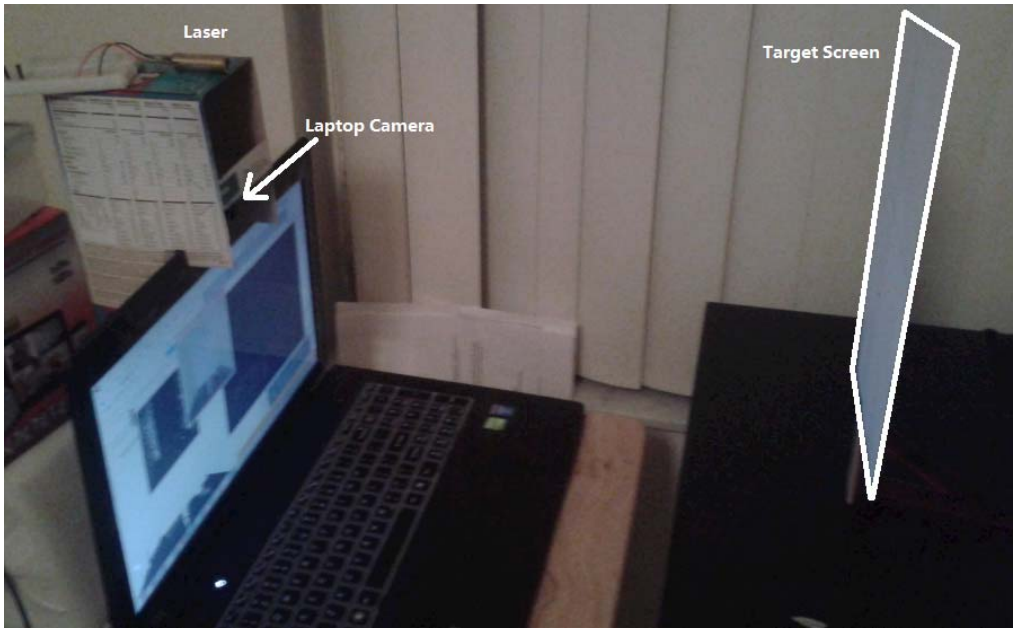


Fig 14. Experimental setting for final design of special range finder

I modified the distance between laptop and target screen and to see if the position of the laser dot varies. The following picture shows the camera captured the laser dot successfully. And the y position of the laser dot changes with the variation of the distance.

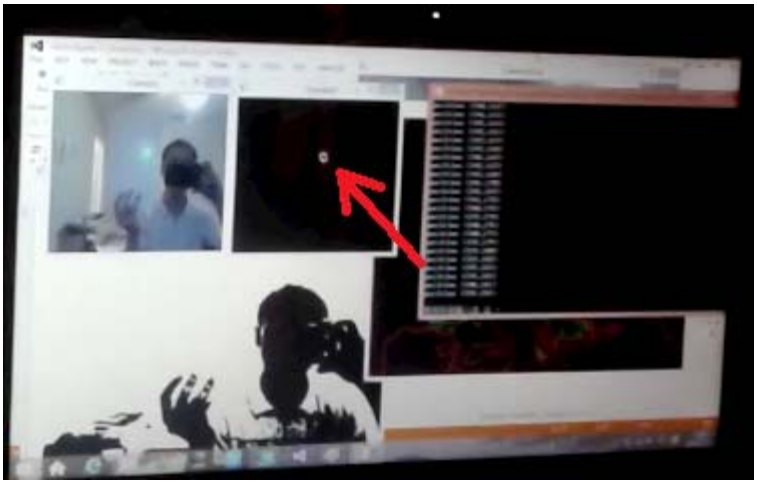


Fig 15. Detecting of laser dot

The third experiment is to get the constants in the equation and to test if this equation works. I measured the distance between laser pointer and the camera which is 13.3cm. The screen is located 51cm and 61cm from the camera. After calculation, Rpp is 0.001443908311 and ERR is 0.02985287293 for my laptop camera. Then, I tested my rangefinder and it works. The test video can be found under project updates on my web page.

Ⅹ.Future Work

As mentioned in the previous report. There are some parts can be improved in

the future. One is the OpenCV method used to detect target. As the contours of small animals are not simply circles, detecting moving objects may be a good method to use. Or using thermo camera instead of ordinary camera with night vision may be easier to focus on those small animals.

Another improvement can be the laser range finder. As the resolution of the camera greatly limited the detecting range, a better camera may be used to improve this sensor.

XI.Conclusion

In conclusion, Night Watcher successfully completed the tasks set up for it. This robot can autonomously find target and shoot at it. Such behavior fulfilled the goal that humans drive small animals away without direct contact with them. Thus, lower the risk of being hurt or catching diseases from those animals. And the nerf dart will not hurt small animals, it will just fright them. With the help of this robot, unwelcomed small animals are kept away from people safely.

If the detecting system used on this robot can be improved to adapt more complex natural environment, this robot can be a good helper for those who have to live with wild animals but do not want to have too much direct contact with them. And the robot can be made bigger and use anaesthetic rifle to deal with bigger and more dangerous animals. Moreover, when being remote controlled, these robots can be used as police on the street or soldiers in the battle field.

XII.Documentation

[1] Tracking colored objects in OpenCV Accessed on 14 March 2014. http://www.aishack.in/2010/07/tracking-colored-objects-in-opencv/

[2] Use camera to detect distance. Accessed on 14 March 2014. http://blog.csdn.net/xylary/article/details/1843809

[3] Using OpenCV 2.44 in Vsiual Studio 2012. Accessed on 25 February 2014. http://answers.opencv.org/question/6495/visual-studio-2012-and-rtlfreeheap-error/#6603

[4] AdaFruit Motor shield. Accessed on 14 February 2014. https://www.youtube.com/watch?v=vN_gcyWKCxY

[5] Serial communication between PC and Arduino. Accessed on 18 March 2014. http://aleksandarkrstikj.com/

[6] Relay circuit for Arduino. Accessed on 25 March 2014. http://playground.arduino.cc/uploads/Learning/relays.pdf

[7] Calculating force and torque needed for the motors. Accessed on 30 January 2014. http://www.pololu.com/blog/10/force-and-torque

[8] Setting up XBees. Accessed on 20 March 2014. https://www.youtube.com/watch?annotation_id=annotation_262505&feature=iv&src_vid=odekkumB3WQ&v=mPx3TjzvE9U

XIII.Appendices

Design of the main circuit

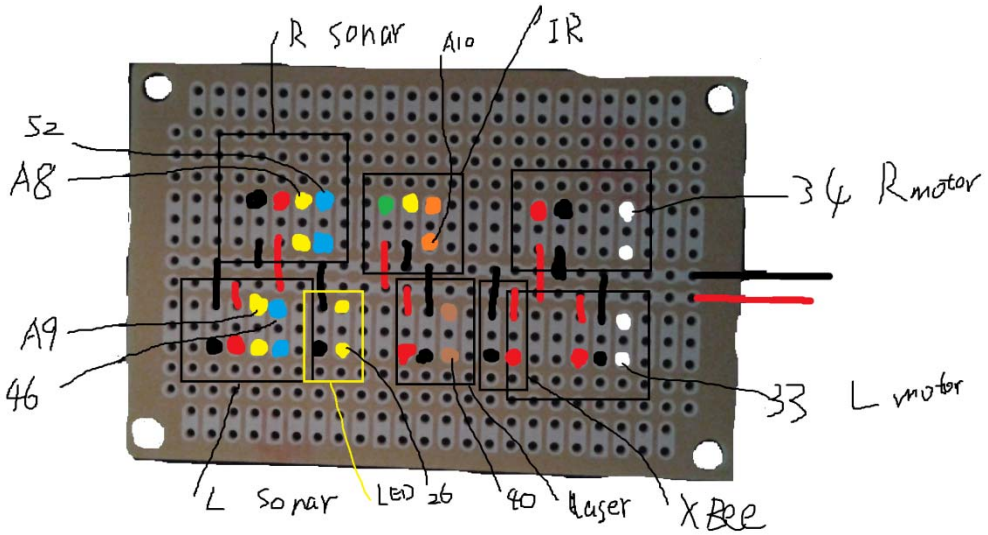


Fig 16. Design of main circuit

Weight estimate

UltraSonic	23 grams	
IR sensor	10 grams	
switch	20 grams	
switch	20 grams	
Xbee	5.6 grams	
Stepper Motor	285 grams	
DC motor	20 grams	
DC motor	20 grams	
DC motor	20 grams	
DC motor	20 grams	
Stepper Motor Ctrl40 grams		
IP camera	453 grams	
Something else	200 grams	
Battery * 8	240 grams	
Wood25*20*3mm	400 grams	
Wood Turret	200 grams	
Total	2174.6	Use 3kg as total
wheel	40 mm diameter	
friction coefficient	0.25-0.4	
		torque total 2.4kg.cm

Arduino code

```
/*
  Code used to drive Ultrasonic Range Finder HC-SR04. This type of range finder
  has four pins. VCC for power in, GND for ground, Trig for output signal and Echo
  for input signal. When given a high voltage signal over 10 us the range finder
  will work automatically. It will send back the time sound takes to get back. so
  the distance is (time*sound speed(340m/s))/2.

  This code will measure distance, and if distance between 20 and 15 cm, green led
  on, if between 15-10cm, green and yellow leds on, if less than 10cm, all leds
  on and the buzzer on. Else all leds off.

  created 10 Feb. 2014

  by Yanming(Patton) Wang

  */

// These constants won't change.    They're used to give names
// to the pins used:
const int ECHOPIN = A8;    // Analog input pin for ECHO of sonar
const int TRIGPIN = A9;    // Analog output pin for TRIG of sonar
const int speaker = 46;    // Digital output pin for buzzer
const int ledG = 48;          // Digital output pin for Green led
const int ledY = 50;          // Digital output pin for Yellow led
const int ledR = 52;        // Digital output pin for Red led


void setup() {
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
  pinMode(ECHOPIN,INPUT) ;
  pinMode(TRIGPIN,OUTPUT);
  pinMode(ledG,OUTPUT);
  pinMode(ledY,OUTPUT);
  pinMode(ledR,OUTPUT);
  pinMode(speaker,OUTPUT);
}

void loop() {
digitalWrite(TRIGPIN,LOW);    //set trig low
delay(2);
digitalWrite(TRIGPIN,HIGH); //set trig high start measuring distance
delay(10);
digitalWrite(TRIGPIN,LOW);
```

```
float distance=pulseIn(ECHOPIN,HIGH);//get time the ssound traveled
distance=distance/58;                  // change time to distance
Serial.println(distance); // print distance
delay(200);                                  // decide which led on and whether buzzer on
if(distance<=20 && distance>15){
   digitalWrite(ledG,HIGH);
   digitalWrite(ledY,LOW);
   digitalWrite(ledR,LOW);
   delay(100);
}
else if(distance<=15 && distance>10){
   digitalWrite(ledG,HIGH);
   digitalWrite(ledY,HIGH);
   digitalWrite(ledR,LOW);
   delay(100);
}
else if( distance<=10){
     digitalWrite(ledG,HIGH);
   digitalWrite(ledY,HIGH);
   digitalWrite(ledR,HIGH);
   int i=0;
   while(i<=20){
   digitalWrite(speaker,HIGH);
   delay(2);
   digitalWrite(speaker,LOW);
   i++;
   }
   delay(100);
}
else{
     digitalWrite(ledG,LOW);
   digitalWrite(ledY,LOW);
   digitalWrite(ledR,LOW);
   delay(100);
}
}
```

```
/*
use one input pin of the encoder to count the number of slits passed by.
the Ratio of the Gear Box is 77, There are 334 pulses for each full turn
of the motor. This code can clearly count the number of pulses even the
motor runs in full speed.
```

```
Num=number of pulses
D=wheel diameter 25mm
t=time duration of the test
Speed=(Num/334/78 * PI*D)/t
2014/2/23
by Yanming Wang

*/


#include <AFMotor.h>;

AF_DCMotor motor1(3);
AF_DCMotor motor2(4);

const int encoderPin1 = 34;
const int encoderPin2 = 33;
unsigned long time = 0;
const int power = 22;
long encoderPos1 = 0;
long encoderPos2 = 0;

boolean encoderLast1 = LOW;    // remembers the previous pin state
boolean encoderLast2 = LOW;    // remembers the previous pin state

void setup()
{
   pinMode(power, OUTPUT);
   pinMode(encoderPin1, INPUT);
   pinMode(encoderPin2, INPUT);

   digitalWrite(encoderPin1, HIGH);
   digitalWrite(encoderPin2, HIGH);

   Serial.begin (9600);
}

void loop()
{
       motor1.run(FORWARD);
       motor1.setSpeed(244);
       motor2.run(FORWARD);
       motor2.setSpeed(244);
```

```
   time = millis();

   testSpeed();

   delay(1000);
}

void testSpeed()
{
   encoderPos1 = 0;
   encoderPos2 = 0;

   digitalWrite(power,HIGH);

   while (millis()-time<300)
   {
   boolean encoder1 = digitalRead(encoderPin1);
   if ((encoderLast1 == HIGH) && (encoder1 == LOW))
   {
        encoderPos1=encoderPos1 + 1;
       }
   encoderLast1 = encoder1;
   }
   time = millis();
  delay(5);


  while (millis()-time<300)
   {
   boolean encoder2 = digitalRead(encoderPin2);
   if ((encoderLast2 == HIGH) && (encoder2 == LOW))
   {
        encoderPos2=encoderPos2 + 1;
       }
   encoderLast2 = encoder2;
   }
   long speed1;
   long speed2;
   speed1 = encoderPos1/309;
   speed2 = encoderPos2/309;
   Serial.print ("Motor1 Speed = ");
   Serial.print (speed1);
   Serial.print ("mm/s");
   Serial.print ("Motor2 Speed = ");
```

```
    Serial.print (speed2);
    Serial.print ("mm/s");
    Serial.println ();
    time = millis();
    delay(5);
    return;
}
```

```
/* Final code for demo day including obstacle avoidance*/
#include <Servo.h>
#include <AFMotor.h>

  int go = 1;
  int jj=0;
//for turret servo
Servo servo;
int servoPosition = 95;
int lastPosition = 95;
//for auto loader servo
Servo loader;
int loaderPosition = 30;
//for driving motors
AF_DCMotor motor1(3);
AF_DCMotor motor2(4);
//for gun motor
AF_DCMotor gun1(1);
AF_DCMotor gun2(2);
//for sonar and IR sensors
const int ECHOPIN1 = A8;    // Analog input pin for ECHO of sonar1
const int TRIGPIN1 = 52;    // Analog output pin for TRIG of sonar1
const int ECHOPIN2 = A9;    // Analog input pin for ECHO of sonar2
const int TRIGPIN2 = 46;    // Analog output pin for TRIG of sonar2
const int analogInPin = A10; // Analog input IR sensor
//distances gathered from ordinary sensor
float distance1 = 0;
float distance2 = 0;
float distance3 = 0;
int ran;//randam number
//for Serial communication
int i=0;
char incomingByte = 0;      // for incoming serial data
int num=0;
int t = 0;
int led = 26; //led to switch on IP camera day mode
```

```
int laser = 40; //laser launcher
int counter = 0;
int target = 0;

void setup()
{
   Serial.begin(9600); //    opens serial port, sets data rate to 9600 bps
   Serial1.begin(9600);// opens Xbee serial port
   servo.attach(24); // attaches the servo on pin 10 to the servo object
   servo.write(servoPosition); // set the servo at the mid position
   loader.attach(9);//    attaches the loader on pin 9
   loader.write(loaderPosition);//set the loader at beginning position
   pinMode(led,OUTPUT);//set led
   pinMode(laser,OUTPUT);//set laser
     pinMode(ECHOPIN1,INPUT) ;
 pinMode(TRIGPIN1,OUTPUT);
   pinMode(ECHOPIN2,INPUT) ;
 pinMode(TRIGPIN2,OUTPUT);
   motor1.setSpeed(200);
   motor1.run(RELEASE);
   motor2.setSpeed(200);
   motor2.run(RELEASE);
   gun1.setSpeed(100);
   gun1.run(RELEASE);
   gun2.setSpeed(100);
   gun2.run(RELEASE);
}

void attack(){
   gun1.setSpeed(150);
   gun1.run(FORWARD);
   gun2.setSpeed(150);
   gun2.run(FORWARD);
   delay(3000);
   loader.write(180);
   delay(150);
   loader.write(30);
   delay(500);
   gun1.setSpeed(100);
   gun1.run(RELEASE);
   gun2.setSpeed(100);
   gun2.run(RELEASE);
}
```

```
void laserON(){
   digitalWrite(led, HIGH);
   digitalWrite(laser,HIGH);
   delay(50);
}

void laserOFF(){
   digitalWrite(led,LOW);
   digitalWrite(laser,LOW);
   delay(50);
}

void motoroff(int t){
   motor1.setSpeed(200);
   motor2.setSpeed(200);
   motor1.run(RELEASE);
   motor2.run(RELEASE);
   delay(t);
   return;
}

void motorforward(int t){
   motor1.run(FORWARD);
   motor2.run(FORWARD);
   motor1.setSpeed(170);
   motor2.setSpeed(170);
   //Serial.println("Forward");
   delay(t);
   return;
}

void backoff(int t){
     motor1.run(BACKWARD);
   motor2.run(BACKWARD);
   motor1.setSpeed(200);
   motor2.setSpeed(200);
   delay(t);
   return;
}

void turnRight(int t){
   motoroff(10);
   backoff(200);
   motoroff(10);
```

```
   motor1.run(BACKWARD);
   motor2.run(FORWARD);
   motor1.setSpeed(150);
   motor2.setSpeed(150);
   delay(t);
  // motoroff(10);
}

void turnLeft(int t){
       motoroff(10);
   backoff(200);
   motoroff(10);
   motor2.run(BACKWARD);
   motor1.run(FORWARD);
   motor1.setSpeed(150);
   motor2.setSpeed(150);
   delay(t);
   //motoroff(10);
}

void bubble_sort(float d[], int SIZE)
{
   int exchange = SIZE - 1;
        while(exchange)
      {
            int bound = exchange;
      exchange = 0;
           for(int i = 0; i < bound; i++)
            {
              if (d[i] > d[i + 1])
                 {
                  float t = d[i];
                  d[i] = d[i + 1];
                  d[i + 1] = t;
                  exchange = i;
                   }
            }
       }
  }

void TurretLeft(int t){
     servoPosition+=5;
       if (servoPosition > 180)
          {
```

```
            servoPosition = 180;
            turnLeft(200);
          }
          lastPosition=servoPosition;
          servo.write(servoPosition);
   delay(t);
}

void TurretSlightlyL(int t){
     servoPosition+=1;
       if (servoPosition > 180)
          {
            servoPosition = 180;
            turnLeft(50);
          }
          lastPosition=servoPosition;
          servo.write(servoPosition);
   delay(t);
}

void TurretRight(int t){
   servoPosition-=5;

          if (servoPosition < 5)
          {
            servoPosition = 5;
            turnRight(200);
          }
          lastPosition=servoPosition;
          servo.write(servoPosition);
   delay(t);
}

void TurretSlightlyR(int t){
   servoPosition-=1;

          if (servoPosition < 5)
          {
            servoPosition = 5;
            turnRight(50);
          }
          lastPosition=servoPosition;
          servo.write(servoPosition);
   delay(t);
```

```
}

void TurretCenter(int t){
   if( lastPosition < 95)
   {
   for(servoPosition = lastPosition; servoPosition < 97; servoPosition += 1)    // goes
from 0 degrees to 180 degrees
   {                                                  // in steps of 1 degree
      servo.write(servoPosition);                     // tell servo to go to position in
variable 'pos'
      delay(15);                                   // waits 15ms for the servo to reach the
position
   }
   servo.write(95);
   delay(15);
   }
   else
   {
      for(servoPosition = lastPosition; servoPosition > 93; servoPosition -= 1)    // goes
from 0 degrees to 180 degrees
   {                                                  // in steps of 1 degree
      servo.write(servoPosition);                     // tell servo to go to position in
variable 'pos'
      delay(15);                                   // waits 15ms for the servo to reach the
position
   }
   servo.write(95);
   delay(15);
   }
   servoPosition = 95;
   lastPosition=servoPosition;
   delay(t);
}

void measure()
{
   int i;
  float a[2];

  for(i=0;i<=2;i++)    //Sonar Right
  {
digitalWrite(TRIGPIN1,LOW);    //set trig low
delay(2);
digitalWrite(TRIGPIN1,HIGH); //set trig high start measuring distance
```

```
delay(10);
digitalWrite(TRIGPIN1,LOW);
float distance1=pulseIn(ECHOPIN1,HIGH);//get time the ssound traveled
a[i]=distance1;
Serial.print((distance1)/58);
Serial.print(" ");
  }
Serial.println();
  bubble_sort(a,3);
  distance1=a[1];
distance1=(distance1)/58;                  // change time to distance
Serial.print("Distance1=");
Serial.print(distance1); // print distance
Serial.println();
delay(10);


for(i=0;i<=2;i++)    //Sonar Left
  {
digitalWrite(TRIGPIN2,LOW);    //set trig low
delay(2);
digitalWrite(TRIGPIN2,HIGH); //set trig high start measuring distance
delay(10);
digitalWrite(TRIGPIN2,LOW);
float distance2=pulseIn(ECHOPIN2,HIGH);//get time the ssound traveled
a[i]=distance2;
Serial.print((distance2)/58);
Serial.print(" ");
  }
Serial.println();
  bubble_sort(a,3);
  distance2=a[1];
distance2=(distance2)/58;                  // change time to distance
Serial.print("Distance2=");
Serial.print(distance2); // print distance
Serial.println();
delay(10);

for(i=0;i<=2;i++)      //Sensor Middle
{
distance3 = analogRead(analogInPin);
Serial.print("IR=");
Serial.print(distance3); // print distance
Serial.print(" ");
```

```
a[i]=distance3;
delay(5);
}
bubble_sort(a,3);
  distance3=a[1];
  Serial.println(distance3);
  return;
}

void patrol()
   {
      Serial.println(incomingByte);
      go = 1;
   for(servoPosition = 95; servoPosition < 145; servoPosition += 1)
   {
      servo.write(servoPosition);
      lastPosition=servoPosition;
      if(Serial1.available()>0)
      {
        num=Serial1.available();
                if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
                  }
                    incomingByte = char(Serial1.read());
            if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
              {
                  go = 0;
                  Serial.println(incomingByte);
                  break;
              }
      }
      delay(250);
   }
           delay(500);
           if(go == 1)
             {
                for(servoPosition = 145; servoPosition > 94; servoPosition -= 1)
                 {
                    servo.write(servoPosition);
                    lastPosition = servoPosition;
                  delay(50);
                    }
```

```
            for(servoPosition = 95; servoPosition > 45; servoPosition -= 1)
           {
      servo.write(servoPosition);
      lastPosition = servoPosition;
        num=Serial1.available();
              if(num>1){
                    for(int i=0; i<num-1; i++)
                    int waste = Serial1.read();
                        }
                incomingByte = char(Serial1.read());
                   if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
              go = 0;
              Serial.println(incomingByte);
              break;
                    }
      delay(250);
   }
           }
         delay(500);
         if(go ==1)
          {
            TurretCenter(50);
          }


      if(go ==1)
      {
        for( i=0;i<3;i++)
        {
      measure();
      if(distance3 < 650){
        if(distance1 <= 4 || distance2 <= 4){
          backoff(600);
          num=Serial1.available();
              if(num>1){
                    for(int i=0; i<num-1; i++)
                    int waste = Serial1.read();
                        }
                incomingByte = char(Serial1.read());
                   if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
```

```
                    {
                break;
                    }
        }
        else if (distance1 >= 15 && distance2 >=15){
           motorforward(600);
         // motoroff(50);

              num=Serial1.available();
               if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
                             }
                  incomingByte = char(Serial1.read());
                     if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
                break;
                    }
           //motoroff(50);
        }
        else{
           if(distance1 >= distance2){
              turnRight(500);

                num=Serial1.available();
               if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
                             }
                  incomingByte = char(Serial1.read());
                     if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
                break;
                    }
           }
           else{
              turnLeft(500);

                num=Serial1.available();
               if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
```

```
                    }
                incomingByte = char(Serial1.read());
                    if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
              break;
                    }
          }
        }
      }
      else{
        motoroff(50);
        backoff(500);
        motoroff(50);

          num=Serial1.available();
            if(num>1){
                  for(int i=0; i<num-1; i++)
                  int waste = Serial1.read();
                    }
                incomingByte = char(Serial1.read());
                    if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
              break;
                    }
        measure();
        if( distance1 <= 15 || distance2 <= 15)
        {
          if( distance1 <= 15 && distance2 <= 15 )
          {
            backoff(300);
            motoroff(50);

              num=Serial1.available();
            if(num>1){
                  for(int i=0; i<num-1; i++)
                  int waste = Serial1.read();
                    }
                incomingByte = char(Serial1.read());
                    if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                    {
              break;
```

```
                    }
            }
            else if( distance1 > distance2)
            {
               turnRight(200);

                  num=Serial1.available();
                 if(num>1){
                          for(int i=0; i<num-1; i++)
                          int waste = Serial1.read();
                               }
                    incomingByte = char(Serial1.read());
                       if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                          {
                 break;
                          }
            }
            else
            {
               turnLeft(200);

                  num=Serial1.available();
                 if(num>1){
                          for(int i=0; i<num-1; i++)
                          int waste = Serial1.read();
                               }
                    incomingByte = char(Serial1.read());
                       if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                          {
                 break;
                          }
            }
         }
         else
         {
            int ran = random(100);
            if(ran>=50)
            {
               turnRight(200);
               motoroff(50);

                  num=Serial1.available();
```

```
                if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
                             }
                   incomingByte = char(Serial1.read());
                      if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                        {
                break;
                        }
            }
            else
            {
               turnLeft(200);
               motoroff(50);

                  num=Serial1.available();
                if(num>1){
                        for(int i=0; i<num-1; i++)
                        int waste = Serial1.read();
                             }
                   incomingByte = char(Serial1.read());
                      if(incomingByte == 'l'|| incomingByte == 'r'||incomingByte == 'k'||
incomingByte == 't')
                        {
                break;
                        }
            }
         }
      }

      }

      }

            return;
   }

void action()
{

  int num = Serial1.available();

  incomingByte = 'p';
```

```
  for(int i=0; i<num-1; i++)
  {
                              int waste = Serial1.read();
                                   }
                         incomingByte = char(Serial1.read());
if(target == 0)
{
if(incomingByte == '0')
{
   counter++;
}
else
{
   counter=0;
}
}
else if (target ==1)
{
  if(incomingByte == '0')
  {
     incomingByte= 'p';
  }
  else
  {
    target = 0;
  }
}
if( counter ==3)
{
   incomingByte = 'p';
   counter=0;
   target = 1;
}
Serial.println(incomingByte);
     switch(incomingByte)
     {
        case 'l':
       TurretLeft(50);
        break;

        case 'r':
        TurretRight(50);
        break;
```

```
            case 'c':
            TurretCenter(50);
            break;

            case 'k':
            TurretSlightlyL(10);
            break;

            case 't':
            TurretSlightlyR(10);
            break;

            case 'm':
            laserON();
            break;

            case 'a':
            attack();
            laserOFF();
            TurretCenter(50);
            target =1;
            num=Serial1.available();
                  if(num>1){
                        for(int i=0; i<num; i++)
                        int waste = Serial1.read();
                    }
                 go=1;
            break;

            case 'p':
            patrol();
            break;

            case 's':
            motoroff(10);
            break;
        }
//      if(num>1){
//          for(int i=0; i<num-1; i++)
//           int waste = Serial1.read();
//    }
    num = 0;
}
```

```
void loop(){
     if(Serial1.available()){
       action();
   }
   else
      motoroff(500);
  }
```

OpenCV code

```
#include <stdio.h>
#include <math.h>
#include <deque>
#include "opencv/cv.h"
#include "opencv/highgui.h"
#include "opencv/cxcore.h"
#include <Windows.h>//for Serial communication
#include <math.h> //use for calculating distance based on laser



using namespace std;

CvSeq* getCirclesInImage(IplImage*, CvMemStorage*, IplImage*);
//float eucdist(CvPoint, CvPoint);    //what's this?
void drawCircleAndLabel(IplImage*, float*/*, const char**/);
bool circlesBeHomies(float*, float*);

const int MIN_IDENT = 50;
const int MAX_RAD_DIFF = 10;
const int HISTORY_SIZE = 5;
const int X_THRESH = 15;
const int Y_THRESH = 15;
const int R_THRESH = 20;
const int MATCHES_THRESH = 2;      //detect the same circle more than 3 times
const int HUE_BINS = 32;
  int kk=0;
  int collect=0;
  char direction;
  char lastdirection=1;
    float avg=0;

int main(int argc, char *argv[]) {
```

```
        int ii=0;
        int jj=0;

      // Setup serial port connection and needed variables.
       HANDLE hSerial = CreateFile("COM4", GENERIC_READ |
GENERIC_WRITE, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

      if (hSerial !=INVALID_HANDLE_VALUE)
       {
            printf("Port opened! \n");

            DCB dcbSerialParams;
            GetCommState(hSerial,&dcbSerialParams);

            dcbSerialParams.BaudRate = CBR_9600;
            dcbSerialParams.ByteSize = 8;
            dcbSerialParams.Parity = NOPARITY;
            dcbSerialParams.StopBits = ONESTOPBIT;

            SetCommState(hSerial, &dcbSerialParams);
      }
      else
      {
           if (GetLastError() == ERROR_FILE_NOT_FOUND)
           {
                printf("Serial port doesn't exist! \n");
           }

           printf("Error while setting up serial port! \n");
      }

      char outputChars[] = "c";
      DWORD btsIO;
      //end serial setup


    //CvCapture *capture = 0; //The camera
    IplImage* frame = 0; //The images you bring out of the camera

    //Open the camera
    //capture = cvCaptureFromCAM( 0 );
    CvCapture* capture =
cvCaptureFromFile("http://192.168.23.2/videostream.asf?user=admin&pwd=3612771
```

```
43");
   if (!capture ) {
      printf("Could not connect to camera\n" );
      return 1;
   }

   frame = cvQueryFrame( capture );
   //Create two output windows
   cvNamedWindow( "raw_video", CV_WINDOW_AUTOSIZE );
   cvNamedWindow( "processed_video", CV_WINDOW_AUTOSIZE );

   //Used as storage element for Hough circles
   CvMemStorage* storage = cvCreateMemStorage(0);

   // Grayscale image
   IplImage* grayscaleImg = cvCreateImage(cvSize(640, 480), 8/*depth*/,
1/*channels*/);

   deque<CvSeq*> samples;
   int key = 0;
   while(key != 27 /*escape key to quit*/ ) {
      //Query for the next frame
      frame = cvQueryFrame( capture );
      if( !frame ) break;

      deque<CvSeq*> stableCircles;
      //show the raw image in one of the windows
      cvShowImage( "raw_video", frame );

     if(kk==0)
     {
      CvSeq* circles = getCirclesInImage(frame, storage, grayscaleImg);
     int PreviousR=0;
     int TargetX=320;
      //Iterate through the list of circles found by cvHoughCircles()
      for(int i = 0; i < circles->total; i++ )
     {
         int matches = 0;
         float* p = (float*)cvGetSeqElem( circles, i );
         float x = p[0];
         float y = p[1];
         float r = p[2];
         if (x-r < 0 || y-r < 0 || x+r >= frame->width || y+r >= frame->height) {
            continue;
```

```
    }
    for (int j = 0; j < samples.size(); j++) {
       CvSeq* oldSample = samples[j];
       for (int k = 0; k < oldSample->total; k++) {
          float* p2 = (float*)cvGetSeqElem( oldSample, k );
          if (circlesBeHomies(p, p2)) {
             matches++;
             break;
          }
       }
    }

    if (matches > MATCHES_THRESH) //confirm target after 2 times
   {
         if ( r>PreviousR){      //get the x coordinate for the largest circle
                TargetX=x;
                jj=0;
          }
       drawCircleAndLabel(frame, p);
    }

 }
 samples.push_back(circles);
 if (samples.size() > HISTORY_SIZE) {
    samples.pop_front();
 }
 cvShowImage( "processed_video", frame);
//printf("position (%d)\n", TargetX);


if(TargetX>360)
{
      direction = 'r';
}
else if(TargetX<280)
{
      direction = 'l';
}
else if(TargetX>330 && TargetX<360)
{
      direction = 't';
}
else if(TargetX<310 && TargetX>280)
{
```

```
        direction = 'k';
    }
    else if(TargetX>=310 && TargetX<=330 && TargetX != 320)
    {
        direction = 'm';
        kk=1;
        ii=10;
    }
    //send through serial port
    outputChars[0] = direction;    //define variable for serial output
        ////////////////////////////
        if(direction == lastdirection)
        {
        ii=ii+1;
        }
        else
        {
            lastdirection=direction;
        }
        if(ii==10)
        {
        WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);//write
to serial port
        ii=0;
        direction= '0';
        }
    }

     if(kk==1)
      {

        // if(direction =='m')
        // {
        //outputChars[0] = 'm';    //define variable for serial output
        // }
          if(avg>0)
          {
                    outputChars[0] = 'a';
                 printf("/n");
                printf("%f",avg);
                avg=0;
                kk=0;
                collect=0;
          }
```

```
        WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);//write
to serial port
        //printf(outputChars);
        //printf("%f",avg);
        direction = '0';
        if(outputChars[0] != 'a')
        {
          float d[49];
        //convert to HSV image and get required color
        cvSetImageROI(frame, cvRect(290, 240, 30, 240));    //get interested area
        IplImage* imgHSV = cvCreateImage(cvSize(30, 240), frame->depth, 3);
        cvCvtColor(frame, imgHSV, CV_BGR2HSV);
        IplImage* imgThreshed = cvCreateImage(cvGetSize(frame), 8, 1);
        cvInRangeS(imgHSV, cvScalar(20, 48, 93), cvScalar(95, 107, 255),
imgThreshed);
        cvReleaseImage(&imgHSV);
        cvShowImage("processed_video", imgThreshed);
        //get the center of the laser dot use moment
        CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
          cvMoments(imgThreshed, moments, 1);

          // The actual moment values
          double moment10 = cvGetSpatialMoment(moments, 1, 0);
          double moment01 = cvGetSpatialMoment(moments, 0, 1);
          double area = cvGetCentralMoment(moments, 0, 0);
          // Holding the last and current ball positions
          static int posX = 0;
          static int posY = 0;

          int lastX = posX;
          int lastY = posY;

          posX = moment10/area;
          posY = moment01/area;

          // Print it out for debugging purposes
        // printf("position (%d,%d)\n", posX, posY);

        //calculate the distance between laser pointer and the target

        double Rop=0.001835;    // angle per pixel
        double offset=-0.076692; // offset
        double distance;                //distance to the target
```

```
            double h=115;                       // distance between center of the laser and
center of the cam
            double PixelNum = posY;

             distance=h/tan(PixelNum * Rop + offset);
            if(distance>0)
            {
                  d[collect]=distance;
                  collect++;
                    // We want to draw a dot only if its a valid position
              if(lastX>0 && lastY>0 && posX>0 && posY>0)
                   {
                    // Draw a yellow dot on that position
                    cvCircle(frame, cvPoint(posX, posY), 0, cvScalar(0,255,255), 5);
                   }
            }
            if(collect==30)
            {
                  int i;
                  float sum=0;

                  for(i=10;i<30;i++)
                  {
                        sum=sum+d[i];
                        d[i]=0;
                  }
                  avg=sum/20;
                  printf("distance: (%f)mm", avg);
                  printf("\n");
                  //printf("done");
                  cvResetImageROI(frame);
                  //kk=0;
                  //collect=0;
            }
            }
       }

      //Get the last key that's been pressed for input
      key = cvWaitKey( 1 );
   }
}

CvSeq* getCirclesInImage(IplImage* frame, CvMemStorage* storage, IplImage*
grayscaleImg) {
```

```
    // houghification
    // Convert to a single-channel, grayspace image
    cvCvtColor(frame, grayscaleImg, CV_BGR2GRAY);

    // Gaussian filter for less noise
    cvSmooth(grayscaleImg, grayscaleImg, CV_GAUSSIAN, 7, 9 );

    //Detect the circles in the image
    CvSeq* circles =
cvHoughCircles(grayscaleImg,storage,CV_HOUGH_GRADIENT,2,grayscaleImg->h
eight/4,200,100,50,200 );
    return circles;
}


void drawCircleAndLabel(IplImage* frame, float* p) {
    //Draw the circle on the original image
    CvFont font;
    cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX, 1, 1, 0.0, 1, 8);
    cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])), cvRound(p[2]),
CV_RGB(255,0,0), 3, 8, 0 );
}

bool circlesBeHomies(float* c1, float* c2) {
    return (abs(c1[0]-c2[0]) < X_THRESH) && (abs(c1[1]-c2[1]) < Y_THRESH) &&
            (abs(c1[2]-c2[2]) < R_THRESH);
}
```