Columbus with Bluetooth - a simple wireless map maker

Zheng Zhang
Eunjoo Park

Motivation and concept:

A good safety system can detect the environment and sense the potential danger in effective means. One domestic application could be detection of invaders: if a spy camera detects visitors entering the house, the robot will come to visitors to ask them to show certification (certain signals) and record their identity.

We designed a system that can acquire the approximate location of an obstacles by an ultrasonic sensor scanning in two dimensions. A car with a laser would move to the object based on the locations sensed by ultrasonic scanning. At the same time, the object would be mapped onto a plot on the computer and the corresponding region would be marked with a color that was reconstructed using the data parsed in by a rgb sensor.

Functional Definition:

An assembly of an ultrasonic sensor and a laser on a mini pan which has a servo motor inside scans 0 to 180 degrees.As the ultrasonic sensor acquires signal of obstacles, the car adjusted its angle and moved toward the object. For changing the angle, one of the two gears moved by little amount in order to adjust to a suitable angle. The laser was automatically on if the object lined up with the car. Then the car started to move and would stop until the ultrasonic sensor sensed an appropriate distance to stop the c The certain range should be small enough so that RGB sensor on the front of the car can read the color of the target. Based on the information from the ultrasonic sensor and RGB sensor, we could map it on the monitor with angle, distance, and RGB value.

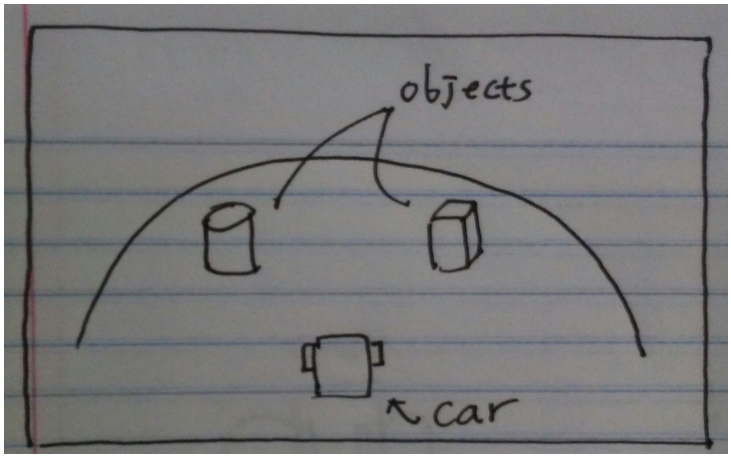
Sensors:

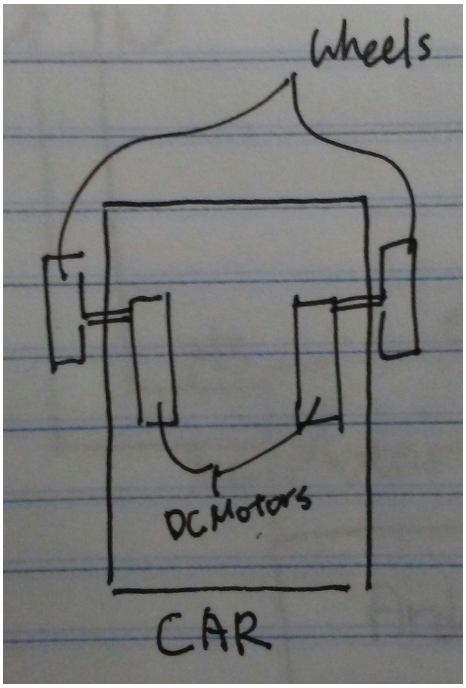
For the environment scan, we used a SR04 ultrasonic sensor. Sat on the mini pan, the sensor is able to spin 180 degrees, reaching every corner of the background. Also the sensor can cover the whole scope with the maximum range of 40 meters.

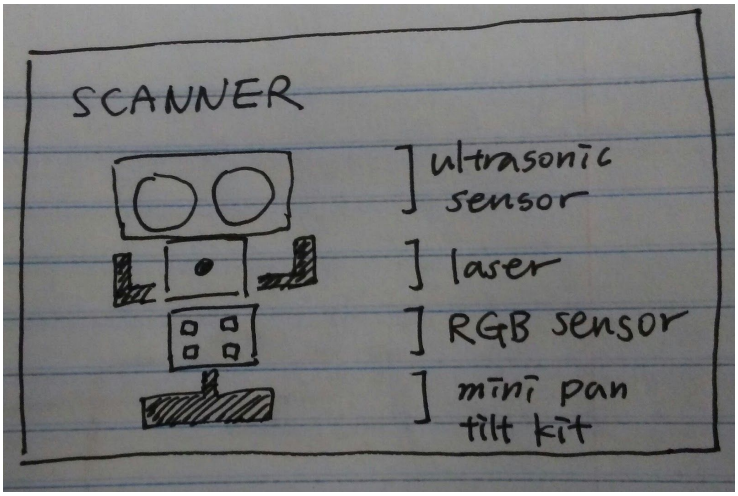
To distinguish the color, we used RGB sensor TCS34725. It acquired the frequencies of red, green, blue and clear color respectively, with which we could reconstruct the object's color in processing.org.

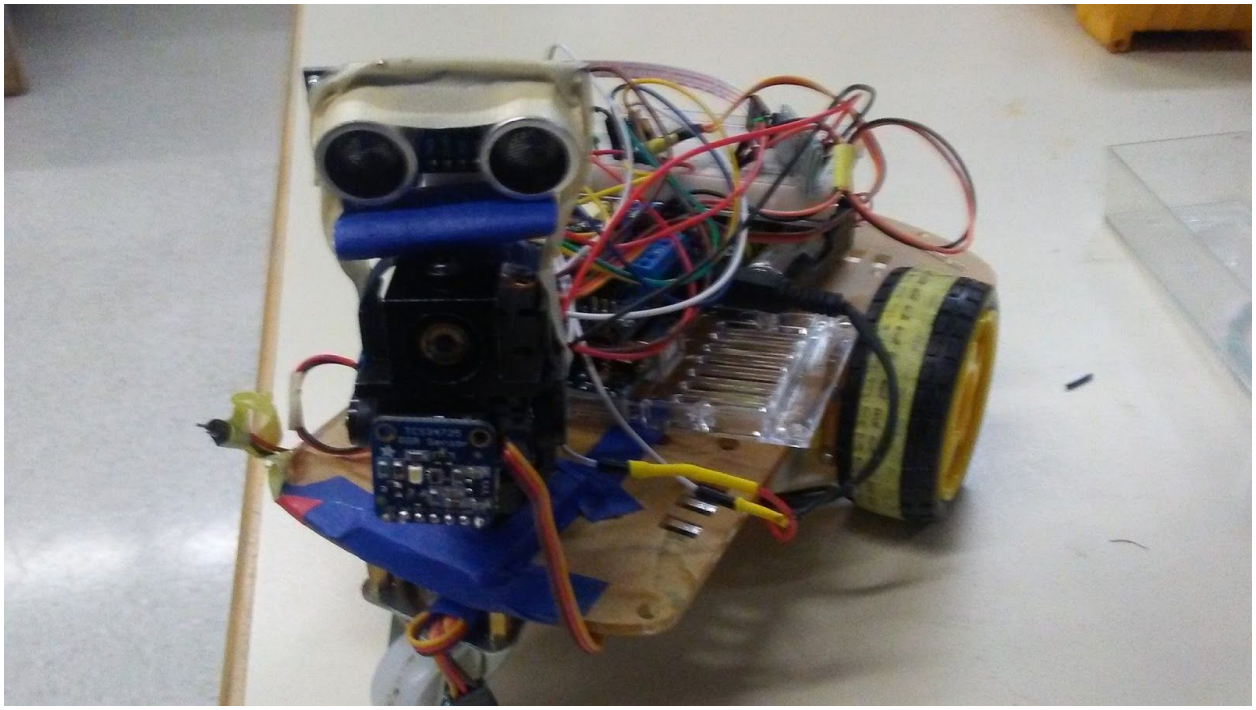
Mechanical Considerations:

The car consisted of 2 DC motors and 2 wheels. On front side of the car, the ultrasonic sensor, laser, and RGB sensor were placed on the mini pan. The min pan assembly was located in the middle of one side and mounted on the car. We used motor shield to control the car moving speed and direction. A bluetooth, batteries, and motor shield with arduino were placed on the car accordingly. The objects needed to be tall and wide enough to reach the level of ultrasonic sensor. The laser and the ultrasonic sensor were mounted in a proper way to avoid interference in gears' movement.





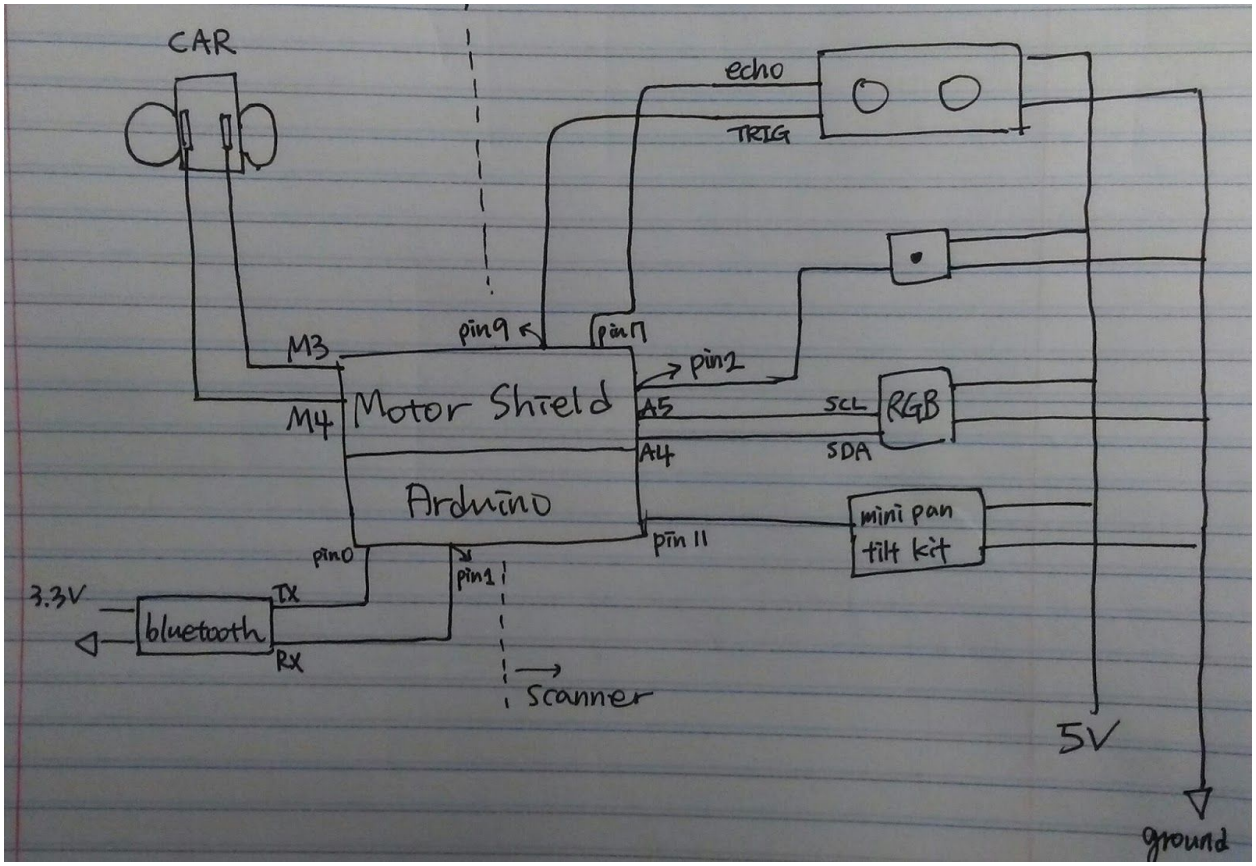




Electrical Considerations:

In a scanning part, mini pan tilt kit, laser, RGB sensor, and ultrasonic sensor were grounded and supported by the batteries, and controlled by arduino. For the ultrasonic sensor, ECHO was connected to pin 7, and TRIG was connected to pin 9(PWM). Servo was connected to pin 11(PWM), and laser was connected to pin 2 and resistor is added between 5V and laser to reduce the brightness. For RGB sensor, SCL and SDA were connected to A5 and A4.

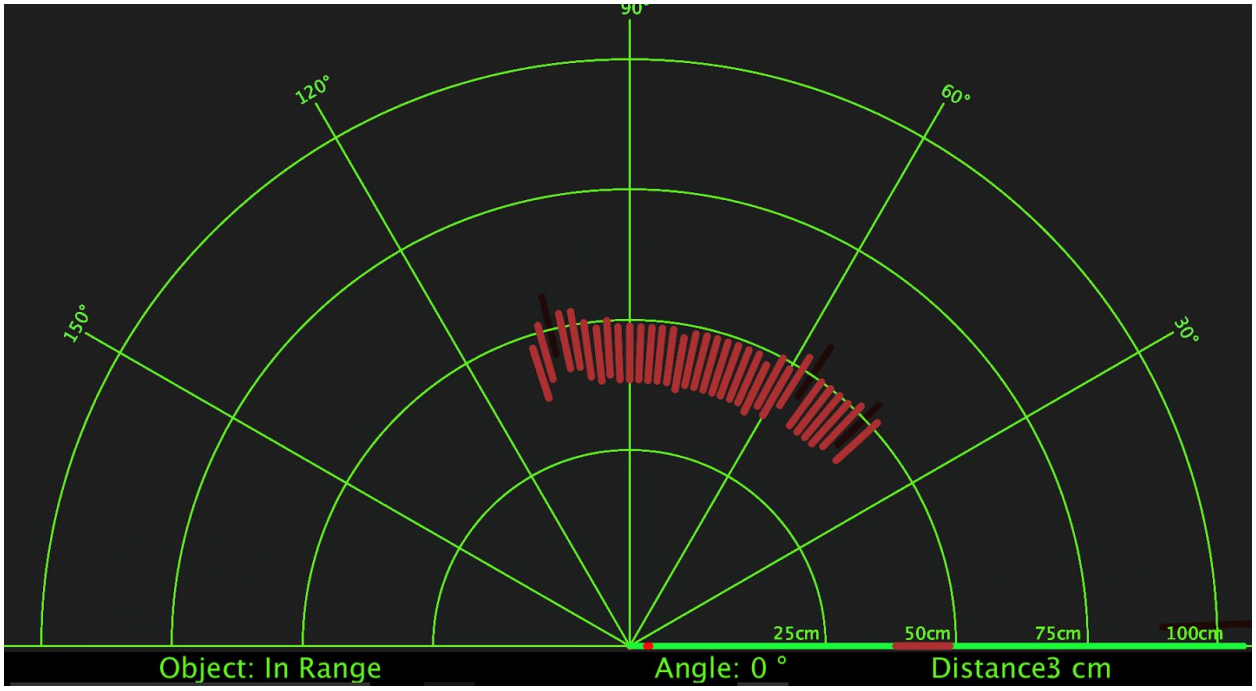
Two DC motors on the car were connected to pin M3 and M4 on the motor shield respectively. TX on the bluetooth was connected to pin 0 which is RX pin, and RX on the bluetooth was connected to TX(pin 1) on the Arduino. The bluetooth needed to be connected to 3.3V and ground.



Interface:

We used one arduino Uno to control the whole operation. The data from ultrasonic scan was parsed into the computer via the arduino, and the arduino sent the response back to monitor the movement of the car. Once the laser was on, i.e. one object is detected, one of the wheels begins to move slowly to find the angle.

For mapping part, we used bluetooth to send the data to the processing.org. The dynamic radar scanning background simulates the ultrasonic scanning, (both angle and the distance).Information from ultrasonic sensor and RGB sensor parsed into Arduino. By bluetooth, serial from the Arduino was sent to processing.org. As the ultrasonic sensor finished scanning, processing.org find nearest point and mark the point with color of the target onto the map using distance and angle from Arduino serial.



(the water prints in the background is the object which is not marked)

Software:

Except mapping part, the software control is written in C, using Arduino IDE. In Arduino IDE, we will import servo library for motor control,(mini-tilt pan). The data from the ultrasonic scan will be processed in Arduino IDE.The rest of the programming will include attaching interrupts, reading from analogues and turning on the laser.

Processing.org is an IDE that is based on Java. We created a dynamic simulating radar scanning background and mapped the object in the range (one meter) onto our map. Sending massive data from arduino bluetooth to processing.org and reading from the data were the trickiest parts. We used a method that is similar to reading from csv files; using '.'.',' to separate the strings and read from buffer strings. Also we applied an queue data structure to find the nearest object on the map.

Also to find our scaling options and the color sensing, we used python for some calculation.

Testing:

There are some parts we need to test: ultrasonic sensing, laser controlling, RGB sensor reading, finding object, sending information via bluetooth and processing the data on the

processing.org. The ultrasonic scan can be tested using a single object in an empty space ; the laser can be tested using simple logic operating under a low speed ; RGB can be tested using any objects which has primary color in close enough distance ; finding objects can be tested with objects that wide and tall; bluetooth can be tested with mobile phone if the information is sent.

Safety:

This system operates at low voltage, low speed, and limited scope without any hazards that would require special consideration. The possible harm is that the laser, as 5 mv emission while turned on, can be harmful to human eyes, which can be avoided by lowering the laser setup level to be away from eye level.

Error Analysis and Suggested Solutions:

The DC motors, which were used to control the car's movement, did not work properly. As DC motors have different configuration, it is impossible to make them moving at the same speed by merely using the motor shield. One of the possible solutions is to encode the gears, The motors came with optical encoded rings which could be used to monitor the speed by adding a photo transistors on one side and led on the other. The speed of gears can be easily monitored by counting the peaks of the phototransistors' readings. We did not apply this method to our project because there were not proper phototransistors available and speed monitoring is not the focus of our project. Another solution is used a four wheels' car instead. Even though it requires more work in coding, a little speed discrepancy won't result in a big change of movement in the four wheels car.

In our project, we tried to solve this problem finding two motors in the lab which has the most similar speed. We measured the time how long those take to make one rotation, setting same speed. Then we adjusted the speed by using different speed and different delay time for each wheel. If one is slower than another one, we set a little bit longer delay.

Also, the RGB sensor required considerable delay time to give a composite of nice frequencies. Even though we tried to use error reducing methods, the final results still have big discrepancy from the object's original color. The discrepancy is reduced to the smallest by sending the sensors close to the car.

Future Expansion: (This was the core of our project, but it was canceled due to the motor issues)

As a better vehicle is applied in this project, we don't need to worry about the routine of the car. We can arrange the car to explore a broader region and we can monitor the car's direction more accurately. In this way, the vehicle can color the whole map by iterating every object. As more objects may be added onto the map and with smaller labels, it would be helpful to construct a weighted graph with objects as it vertices and find the most effective route by calculating its minimum spanning tree, according to which we can send the vehicle to sense the color of each node.

References:

www.howToMechatronics.com , Dejan Nedelkovski
www.arduino.cc,

Appendix : processing file

```
/*  This java file is used to process the data sent via bluetooth
 *  It will plot the objects onto a 'radar' map
 *  framework reference:
 *  Dejan Nedelkovski
 */
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;
import java.util.*;
Serial myPort;
String angle="";
String distance="";
String data="";
String minangle="";
String mindistance="";
int minA, minD;
String noObject;
float pixsDistance;
int iAngle, iDistance;
int oAngle, oDistance;
int index1=0;
int index2=0;
int index3=0;
PFont orcFont;
int finish = 0;
ArrayList A;
ArrayList D;
String red = "";
String blue = "";
String green = "";
String clear = "";
float redI, blueI, greenI, clearI;
int paint;
int minI;
int minI2;
boolean mapped = false;
void setup() {
```

```
 A = new ArrayList();
 D = new ArrayList();
 size (1800, 1000);
 smooth();
 myPort = new Serial(this,Serial.list()[1],9600);
 // reads the data from the serial port up to the character '.'. So actually it reads this:
angle,distance.
 myPort.bufferUntil('.');
}

void draw() {
  fill(98,245,31);
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

 fill(98,245,31); // green color
  drawRadar();
  drawLine();
  drawObject();
  drawText();
  if(paint == 1){
  paintColor();
  A = new ArrayList();
  D = new ArrayList();
  finish =0;
  paint = 0;
  delay(500);
  }
}
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable
"data".
  if(finish == 1) {
          data = myPort.readStringUntil(';');
          data = data.substring(0,data.length()-1);
          index1 = data.indexOf("?");
          index2 = data.indexOf("/");
          index3 = data.indexOf(":");
```

```
        clear = data.substring(0,index1);
        red = data.substring(index1+1,index2);
        green = data.substring(index2+1,index3);
        blue = data.substring(index3+1,data.length());
        redI = float(red);
        blueI = float(blue);
        greenI = float(green);
        clearI = float(clear);
        paint = 1;
  }
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);
  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1"
  angle= data.substring(0, index1); // read the data from position "0" to position of the variable
index1 or thats the value of the angle the Arduino Board sent into the Serial Port
  distance= data.substring(index1+1, data.length()); // read the data from position "index1" to the
end of the data pr thats the value of the distance

  if (angle.charAt(0) == '*') {
        finish = 1;
  }
  // converts the String variables into Integer
  oAngle = iAngle;
  oDistance = iDistance;
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI); // distance  = 100
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI); // distance = 75
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI); // distance = 50
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI); // distance = 25
```

```
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(10);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025)*100/40/4/20*25/2; // covers the
distance from the sensor from cm to pixels
  // limiting the range to 100 cms
  if(iDistance<100){
          D.add(iDistance);
          A.add(iAngle);
          // draws the object according to the angle and the distance

line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(pixsDistance*1.2)*c
os(radians(iAngle)),-(pixsDistance*1.2)*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location

 line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)));
// draws the line according to the angle
  popMatrix();
```

```
}
void drawText() { // draws the texts on the screen

  pushMatrix();
  if(iDistance>100) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("25cm",width-width*0.3854,height-height*0.0833);
  text("50cm",width-width*0.281,height-height*0.0833);
  text("75cm",width-width*0.177,height-height*0.0833);
  text("100cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Object: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("            " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }
  textSize(25);
  fill(98,245,60);

translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/2*sin(ra
dians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();

translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2*sin(rad
ians(60)));
  rotate(-radians(-30));
```

```
  text("60°",0,0);
  resetMatrix();

translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2*sin(rad
ians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();

translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/2*sin(ra
dians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();

translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width/2*sin(r
adians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
void paintColor() {
            pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(10);
  // reconstruct the color using the frequencies
 stroke(redI/clearI*255,greenI/clearI*255,blueI/clearI*255);
// stroke(100,10,10);
 println(clearI);
 println (redI);
 println(blueI);
 println(greenI);
 // find the minDistance
 minD = 9000;
 minI = -1;
for(int i =0; i<D.size();i++){
  int Dis = (int)D.get(i);
  if (Dis < minD && Dis >10){
            minD = Dis;
```

```
            minI = i;
  }
}
// outline the nearest obj
 for (int i =0; i < min(D.size(),A.size());i++) {
            int An = (int)A.get(i);
            int Dis = (int)D.get(i);
            if(Dis<minD*1.3 && Dis >minD*0.7){
            float pix  =Dis*((height-height*0.1666)*0.025)*100/40/4/20*25/2;

line(pix*cos(radians(An)),-pix*sin(radians(An)),(pix*1.2)*cos(radians(An)),-(pix*1.2)*sin(radia
ns(An)));
            }
  }
 println(A.get(minI));
  noLoop();
  popMatrix();
}
```

Arduino files:

```
#include <AFMotor.h>
#include <Servo.h>
#include <Wire.h>
#include <Math.h>

// 9 is trig and 7 is echo
const int trigPin = 9;
const int echoPin = 7;

Servo hitec;   // instantiate a servo
int deg;   // where is servo (in degrees)
int minDistance;// count how many objects are found

int minDeg;
int curDistance;

// 80 degree is the front
AF_DCMotor motorL(3); //left 4
AF_DCMotor motorR(4); //right 1

int angle; //right 0; left 180
String inputStr = "";
// COLOR
#define SensorAddressWrite 0x29 //
#define SensorAddressRead 0x29 //
#define EnableAddress 0xa0 // register address + command bits
#define ATimeAddress 0xa1 // register address + command bits
#define WTimeAddress 0xa3 // register address + command bits
#define ConfigAddress 0xad // register address + command bits
#define ControlAddress 0xaf // register address + command bits
#define IDAddress 0xb2 // register address + command bits
#define ColorAddress 0xb4 // register address + command bits

byte i2cWriteBuffer[10];
byte i2cReadBuffer[10];

void setup(){
```

```
  // set up the servo
  hitec.attach(11,650,2281);

  // set up the laser
  pinMode (2, OUTPUT);

  // set up  the ultrasonic sensors
  pinMode(echoPin,INPUT);
  pinMode(trigPin,OUTPUT);


  motorL.setSpeed(170);
  motorR.setSpeed(175);

  // color stuff

  Wire.begin();
  Serial.begin(9600);  // start serial for output
  init_TCS34725();
  get_TCS34725ID();     // get the device ID, this is just a test to see if we're connected
}


void loop(){

  // find the object
  locate() ;
  // go to the object

  int curDeg = minDeg;
  if (minDeg >90){
    curDeg += 2;
    hitec.write(curDeg);
    motorR.setSpeed(250); //left
    delay(40);
    while (curDeg != 80) {
    motorR.run(BACKWARD);
    delay(30);
    curDeg -= 2;
```

```
    hitec.write(curDeg);
    motorR.run(RELEASE);
    delay(100);
   }
  }

 else if (minDeg < 70){ // the object is right,  right stop, left movesfor(int j = 0; j<
abs(90-minDeg)/2; j++ ){
  while (curDeg != 80) {
   motorL.setSpeed(250);
   motorL.run(BACKWARD);
   delay(25);
   curDeg +=  2;
   hitec.write(curDeg);
   motorL.run(RELEASE);
   delay(100);
 }
}
   delay(1000);

 motorL.setSpeed(195); //left
 motorR.setSpeed(200); //right
 delay(50);
 motorR.run(BACKWARD);
 delay(20);
 motorL.run(BACKWARD);
 curDistance = sense();

while (curDistance > 5 ) {
  delay(100);
  curDistance = sense();
  motorR.run(RELEASE);
  motorL.run(RELEASE);
  delay(30);
  motorR.run(BACKWARD);
  motorL.run(BACKWARD);
}

motorL.setSpeed(0);
```

```
motorR.setSpeed(0);
motorL.run(RELEASE);
motorR.run(RELEASE);
delay(1000);
digitalWrite(2,LOW);
get_Colors();
delay(1000);

}


void locate() {
  minDistance = 9000;
  digitalWrite(9,LOW);
  delayMicroseconds(10);
for ( deg =0; deg<= 180; deg = deg+2)
{
   hitec.write(deg);
  delay(50);
  curDistance = sense();
 Serial.print(deg);
  Serial.print(",");
  Serial.print(curDistance);
  Serial.print(".");

  if (minDistance >= curDistance){
  minDistance = curDistance;
  minDeg = deg;
  delay(170);
  }


}
 Serial.print("*");
  Serial.print(",");
  Serial.print("*");
  Serial.print(".");
  delay(3000);
```

```
hitec.write(minDeg);
digitalWrite(2,HIGH);
delay(100);

}


float sense(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin,HIGH);
  delayMicroseconds(20);
  digitalWrite(trigPin,LOW);
  float distance = pulseIn(echoPin,HIGH)*0.034/2;
  return distance;
}


/*****************************************************************************
***************/
// Everything about colors
/*
Send register address and the byte value you want to write the magnetometer and
loads the destination register with the value you send
*/
void Writei2cRegisters(byte numberbytes, byte command)
{
   byte i = 0;

   Wire.beginTransmission(SensorAddressWrite);   // Send address with Write bit set
   Wire.write(command);                          // Send command, normally the register address
   for (i=0;i<numberbytes;i++)                     // Send data
   Wire.write(i2cWriteBuffer[i]);
   Wire.endTransmission();

   delayMicroseconds(100);      // allow some time for bus to settle
}
/*
Send register address to this function and it returns byte value
```

```
for the magnetometer register's contents
*/
byte Readi2cRegisters(int numberbytes, byte command)
{
  byte i = 0;

   Wire.beginTransmission(SensorAddressWrite);   // Write address of read to sensor
   Wire.write(command);
   Wire.endTransmission();

   delayMicroseconds(100);      // allow some time for bus to settle

   Wire.requestFrom(SensorAddressRead,numberbytes);   // read data
   for(i=0;i<numberbytes;i++)
    i2cReadBuffer[i] = Wire.read();
   Wire.endTransmission();

   delayMicroseconds(100);      // allow some time for bus to settle
}
void init_TCS34725(void)
{
 i2cWriteBuffer[0] = 0x10;
 Writei2cRegisters(1,ATimeAddress);    // RGBC timing is 256 - contents x 2.4mS =
 i2cWriteBuffer[0] = 0x00;
 Writei2cRegisters(1,ConfigAddress);   // Can be used to change the wait time
 i2cWriteBuffer[0] = 0x00;
 Writei2cRegisters(1,ControlAddress);  // RGBC gain control
 i2cWriteBuffer[0] = 0x03;
 Writei2cRegisters(1,EnableAddress);    // enable ADs and oscillator for sensor
}

void get_TCS34725ID(void)
{
 Readi2cRegisters(1,IDAddress);
 if (i2cReadBuffer[0] = 0x44)
   Serial.println("TCS34725 is present");
 else
   Serial.println("TCS34725 not responding");
}
```

```
/*
Reads the register values for clear, red, green, and blue.
*/
char get_Colors(void)
{
  unsigned int clear_color = 0;
  unsigned int red_color = 0;
  unsigned int green_color = 0;
  unsigned int blue_color = 0;

  Readi2cRegisters(8,ColorAddress);
  clear_color = (unsigned int)(i2cReadBuffer[1]<<8) + (unsigned int)i2cReadBuffer[0];
  red_color = (unsigned int)(i2cReadBuffer[3]<<8) + (unsigned int)i2cReadBuffer[2];
  green_color = (unsigned int)(i2cReadBuffer[5]<<8) + (unsigned int)i2cReadBuffer[4];
  blue_color = (unsigned int)(i2cReadBuffer[7]<<8) + (unsigned int)i2cReadBuffer[6];

  // send register values to the serial monitor

  Serial.print(clear_color,DEC);
  Serial.print("?");
  Serial.print(red_color, DEC);
  Serial.print("/");
  Serial.print(green_color, DEC);
  Serial.print(":");
  Serial.println(blue_color, DEC);
  Serial.print(";");


}
```