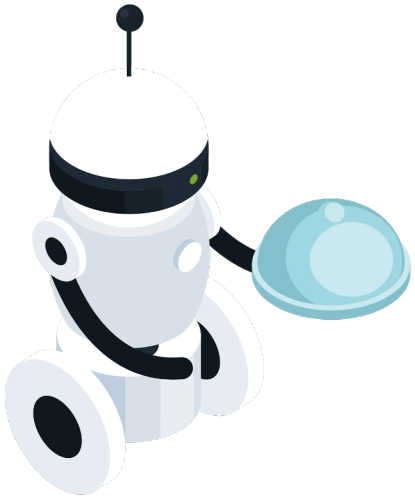
NYU TANDON SCHOOL OF ENGINEERING



[1]

ME-GY 6933
Advanced Mechatronics
Final Project

Vision Controlled Automatic Delivery Bot

Supervisor:
Professor Vikram Kapila
Team:
Kshitij Jindal, Yang Liu, Shiheng Wang

Background and Goal

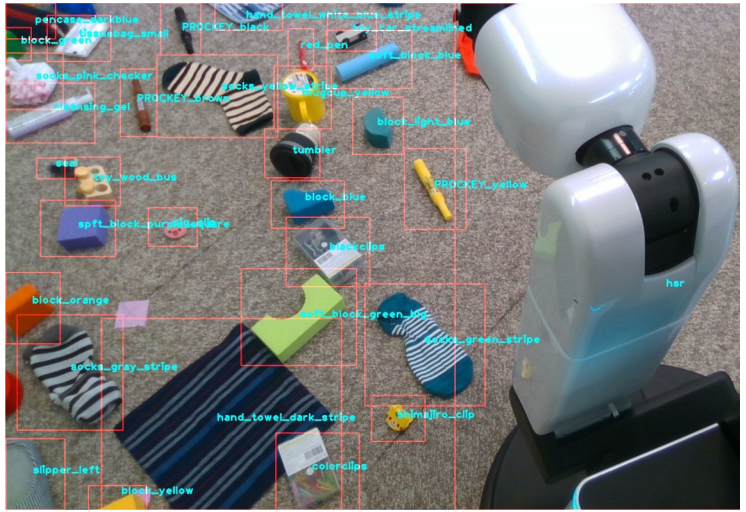
Currently in the market, there's already restaurant using autonomous robot waiters. However, these robots were moving based on line tracking, which was not smart enough.

http://news.ifeng.com/a/20141128/42589560_0.shtml

SHOULD I PICK THIS?

SHOULD I AVOID IT?

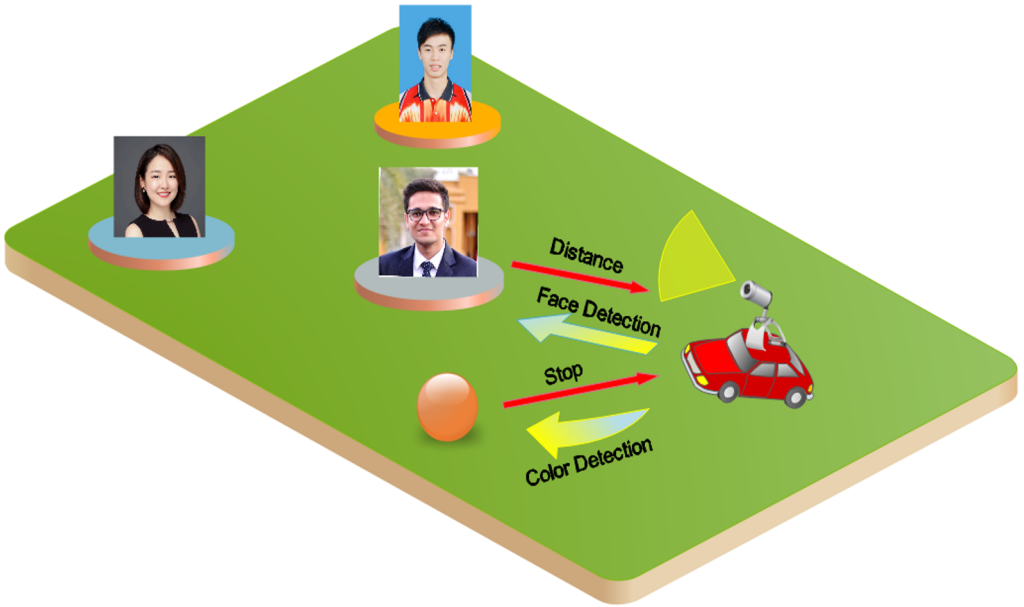
HOW FAR I STILL NEED TO GO?



Objectives

The goal of this project is to prototype of an autonomous delivery bot, which represents an autonomous waiter in real life. The goal of the project includes:

- To build a small size bot which can move to target location and stop when encounters obstacles in midway.
- Set block with human front face picture as objective location, and colored box as obstacle, the bot would start to move when it detect the objective location, but stop while detect obstacle.



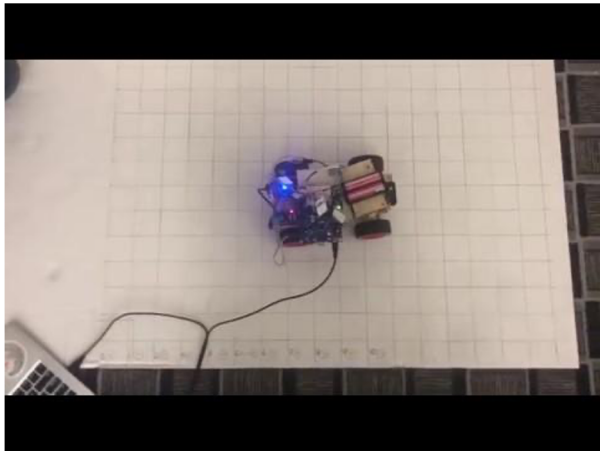
Previous work

Arduino Project

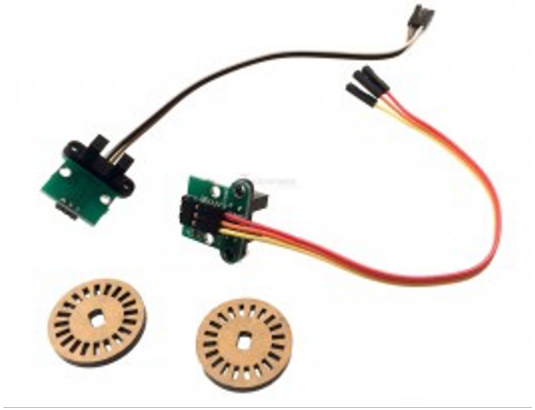
- Design and build an automated delivery bot
- Grab the bottle, drive to designated point and drop it

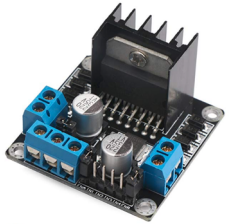
Propeller Project

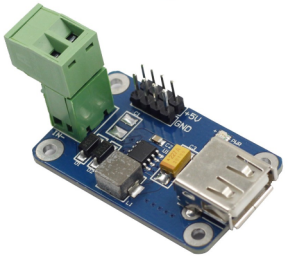
- With multiple cogs, develop the robot to be "smart" enough knowing real-time position without interrupting other tasks processing
- Give more reliable position feedback.

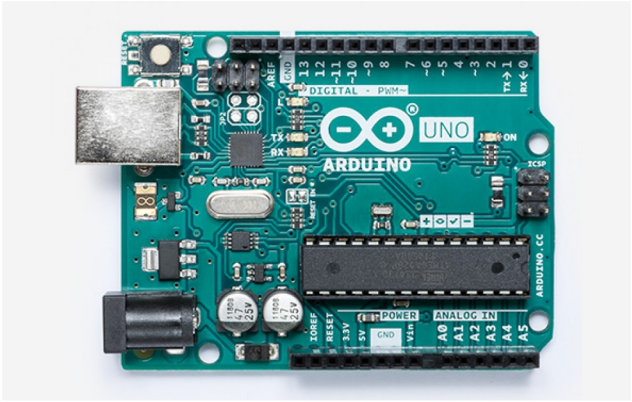


Devices and Elements



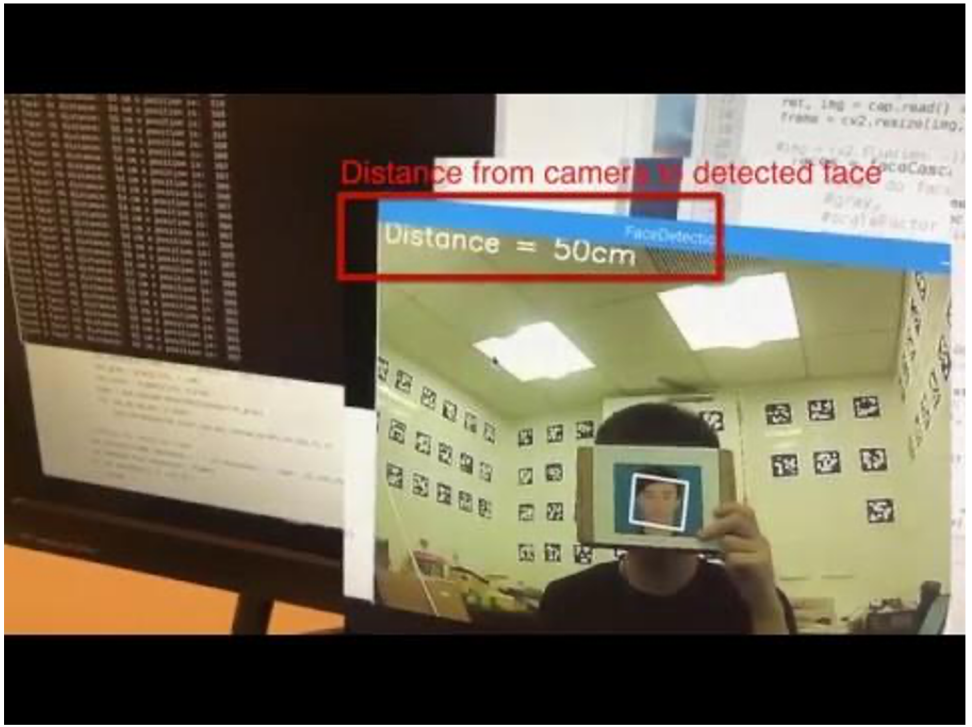






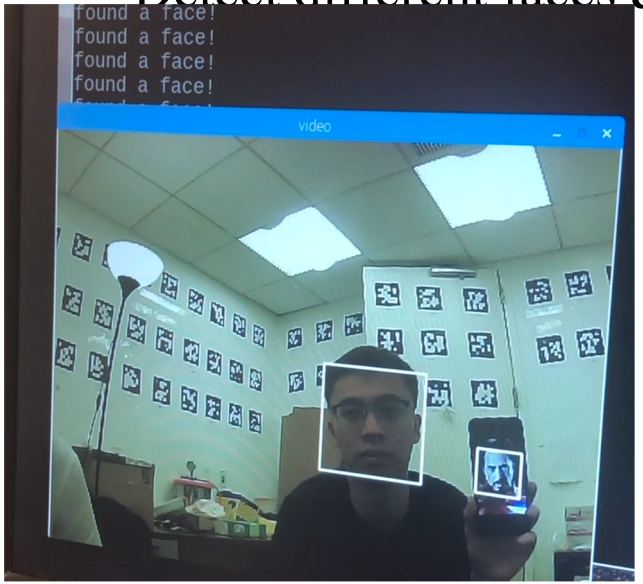
Vision Control Features on R-Pi

- Face Detection
- Distance Measurement
- Color Detection



Face Detection

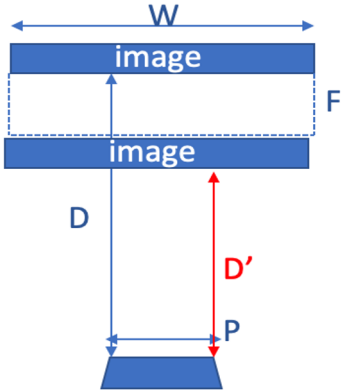
- Detect different faces & print signal(pixels location) in real-time




```python
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)

while (True):
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.3,
        minNeighbors = 3,
        minSize = (15,15)
        )
```

Advantage of using USB camera

Distance Detection



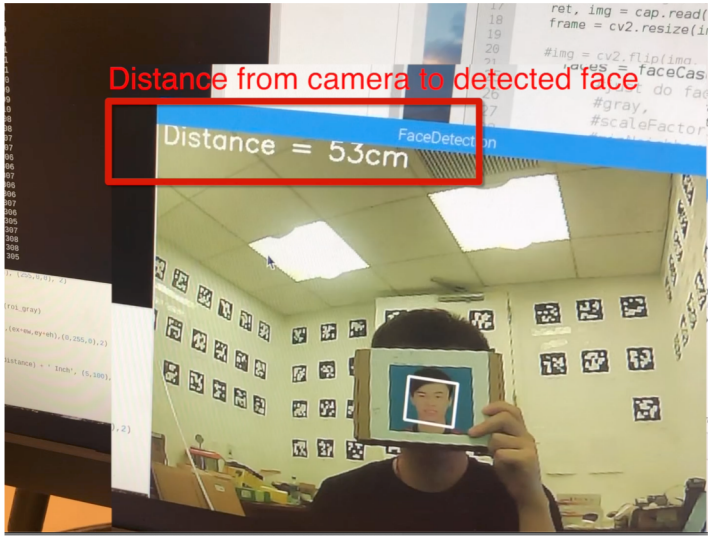
Triangle Similarity

$$F = (P * D) / W$$

$$D' = (F * W) / P$$

```
for (x,y,w,h) in faces:
    distance_i = ((2*3.14*180)/(w+h*360)*1000+3)/2.9
    #print(distance_i) #distance = distance_i * 2.54 (convert inches to cm)

    distance = math.floor(distance_i*2.54)
    # Create rectangle around faces
    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),2)
    #print(x)
    if x:
        print('found a face! At distance: ', distance, 'cm','x position is: ',x)
        x
```

Camera with OpenCV detecting moving face and returning distance



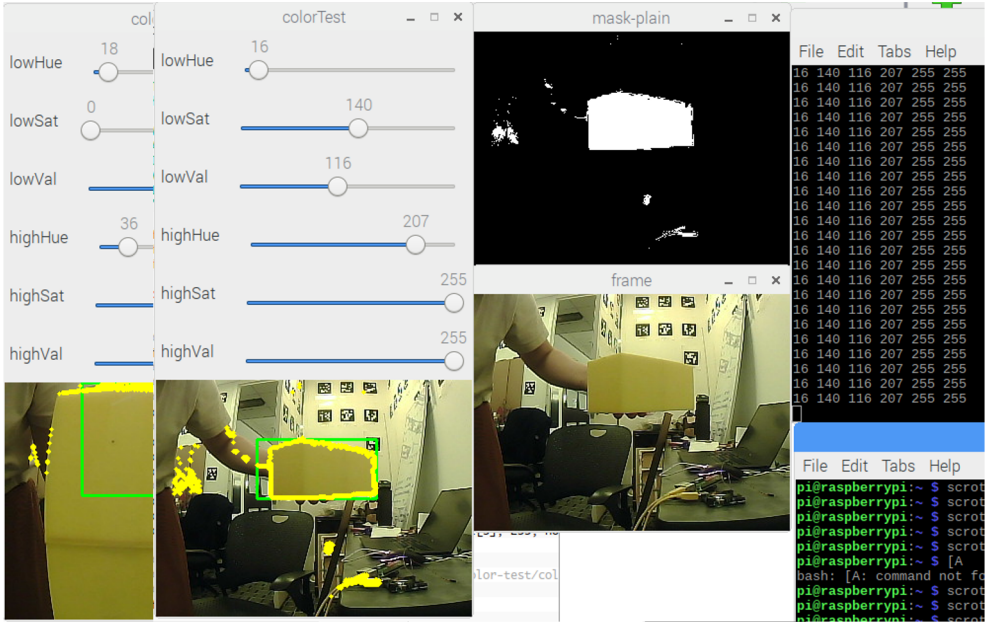
Color Detection

- Detect color ---> contour the color ---> draw certain shape on the biggest objective colored area

```
cv2.namedWindow('colorTest')
# Lower range colour sliders.
cv2.createTrackbar('lowHue', 'colorTest', icol[0], 255, nothing)
cv2.createTrackbar('lowSat', 'colorTest', icol[1], 255, nothing)
cv2.createTrackbar('lowVal', 'colorTest', icol[2], 255, nothing)
# Higher range colour sliders.
cv2.createTrackbar('highHue', 'colorTest', icol[3], 255, nothing)
cv2.createTrackbar('highSat', 'colorTest', icol[4], 255, nothing)
cv2.createTrackbar('highVal', 'colorTest', icol[5], 255, nothing)
```

```
frame = cv2.imread('colour-circles-test.jpg')

while True:
    # Get HSV values from the GUI sliders.
    lowHue = cv2.getTrackbarPos('lowHue', 'colorTest')
    lowSat = cv2.getTrackbarPos('lowSat', 'colorTest')
    lowVal = cv2.getTrackbarPos('lowVal', 'colorTest')
    highHue = cv2.getTrackbarPos('highHue', 'colorTest')
    highSat = cv2.getTrackbarPos('highSat', 'colorTest')
    highVal = cv2.getTrackbarPos('highVal', 'colorTest')
```

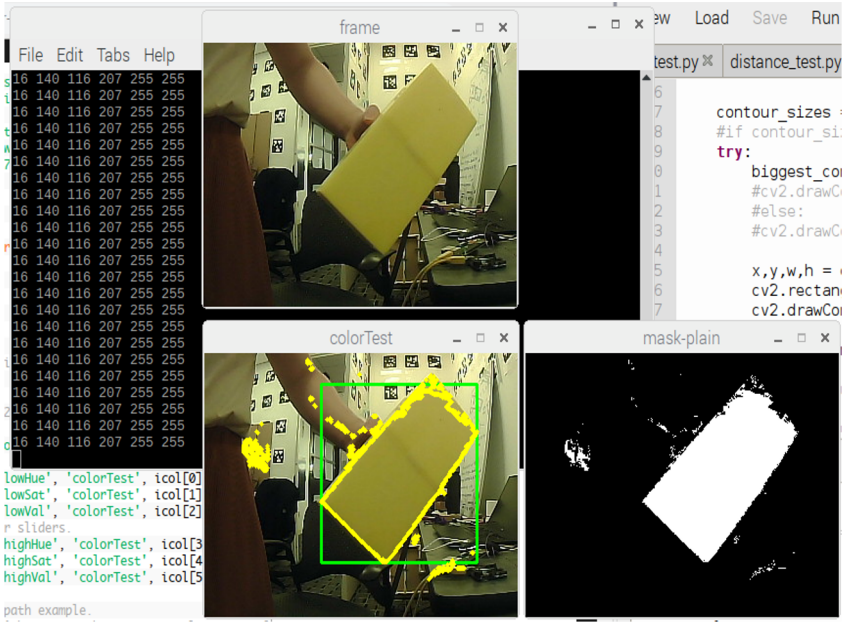
[2]



Color Detection

- Detect color ---> contour the color ---> draw certain shape on the biggest objective colored area

```
#if contour_sizes != []:
try:
    biggest_contour = max(contour_sizes, key=lambda x: x[0])[1]
    #cv2.drawContours(frame, biggest_contour, -1, (0,255,0), 3)
    #else:
    #cv2.drawContours(frame, contours, -1, (0,255,0), 3)

    x,y,w,h = cv2.boundingRect(biggest_contour)
    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.drawContours(frame, contours, -1, (0,255,255), 3)
except:
    cv2.drawContours(frame, contours, -1, (0,255,255), 3)
# Show final output image
cv2.imshow('colorTest', frame)
```

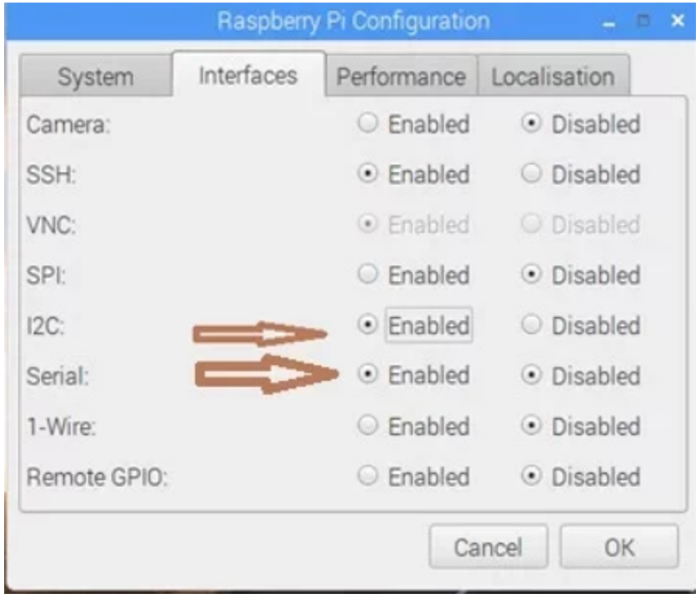


R-Pi and Arduino Communication

```
import serial                                    # import serial library
arduino = serial.Serial("/dev/ttyACM1", 9600)    # create serial object named arduino
```

ls/dev/tty*

```
pi@raspberry:~ $ ls /dev/tty*
/dev/tty    /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0   /dev/tty2   /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1   /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyACM1
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyAMA0
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyprintk
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
```



R-Pi and Arduino Communication

```
void loop() {
  // read the sensor
  IMU.readSensor();
  // display the data
  Serial.print("#AccelX:");
  Serial.print(IMU.getAccelX_mss(),6);
  Serial.println(":");
  Serial.print("#AccelY:");
  Serial.print(IMU.getAccelY_mss(),6);
  Serial.println(":");

  Serial.print("#GyroZ:");
  Serial.print(IMU.getGyroZ_rads(),6);
  Serial.println(":");
  delay(200);
}
```

```
1  import serial
2  ser = serial.Serial("/dev/ttyACM0", 115200)
3  while True:
4    acc_x=ser.readline()
5    acc_y=ser.readline()
6    gyro_z=ser.readline()
7    my_list = [acc_x, acc_y, gyro_z]
8    my_list = sorted(my_list)
9    print(my_list)
10
11   x1 = float(my_list[0].strip().decode("utf-8")[8:-1])
12   y1 = float(my_list[1].strip().decode("utf-8")[8:-1])
13   z1 = float(my_list[2].strip().decode("utf-8")[7:-1])
14   data_list = [x1,y1,z1]
15   print (data_list)
16
17
```

‘b #AccelX: ‘0.1234567 \r\n’

R-Pi and Arduino Communication

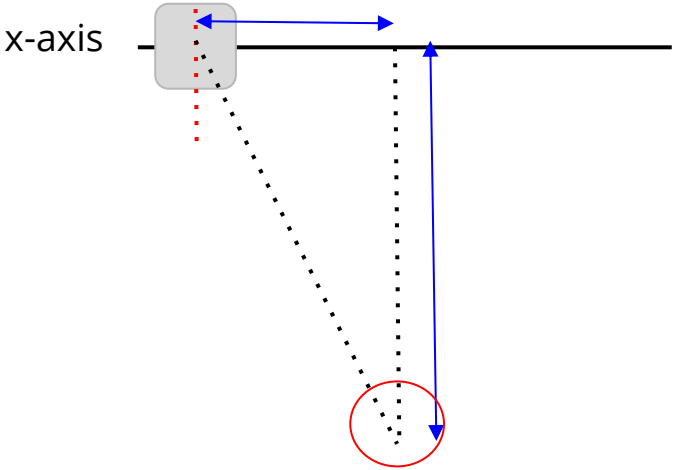
NYU TANDON SCHOOL OF ENGINEERING

```
import serial                                                    # import serial library
arduino = serial.Serial("/dev/ttyACM1", 9600)    # create serial object named arduino
while True:                                                      # create loop

        command = str(input ("Servo position: "))        # query servo position
        arduino.write(str.encode(command))                    # write position to serial port
        reachedPos = str(arduino.readline())             # read serial port for arduino echo
        print   (reachedPos)
```

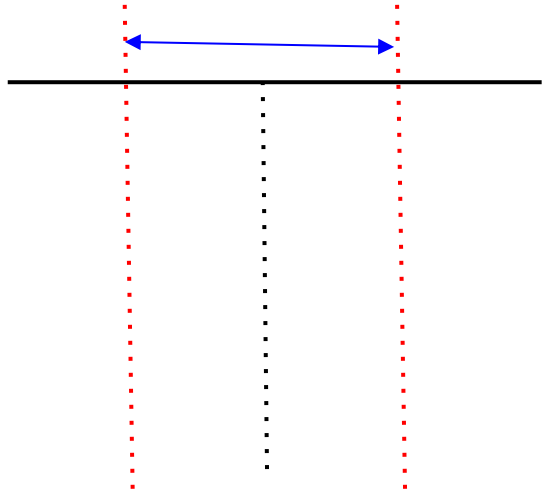
```
void loop()
{
  if(Serial.available())  // if data available in serial port
    {
    inByte = Serial.readStringUntil('\n'); // read data until newline
    pos = inByte.toInt();   // change datatype from string to integer
    myservo.write(pos);     // move servo
    Serial.print("Servo in position: ");
    Serial.println(inByte);
    }
}
```

Future Step

- For direction sensing & increase accuracy



Off-center angle could be detected using camera and OpenCV



Narrow the range of area, in which signal of detected object is taken

Demo