# C:\Users\Admin\Downloads\BVM-Logo-PNG .pngB.V.M Engineering College (AN AUTONOMOUS INSTITUTION)

# (ELECTRONICS ENGINEERING DEPARTMENT)

**ACADEMIC YEAR: AY 2021-22**

**SUBJECT: 3EL05: EMBEDDED AND IOT SYSTEM DESIGN LABORATORY**

**Institute Vision:** “Produce globally employable innovative engineers with core values”

**Institute Mission:**

1. Re-engineering curricula to meet global employment requirements

2. Promote innovative Practices at all levels

3. Imbibe core values

4. Reform policies, systems and processes at all levels

5. Develop faculty and staff members to meet the challenges

**Department: Electronics Engineering**

**Vision:** “Produce globally employable, innovative Electronics engineers with core values”

**Mission:**

1. Promote Innovative Practices to strengthen teaching and learning process in electronics engineering
2. Develop faculty and staff to meet challenges in Electronics engineering
3. Adapt Engineering curricula to meet global requirements for Electronics engineering program me.
4. Reform policies, systems and processes at all levels
5. Imbibe core Values.

**Program Educational Objectives (PEOs):**

1. Design and Development of Hardware and Software Design for Embedded and IOT Systems.

A Project Report On

**2-Axis (X,Y) Face Tracking Gimble**

Mini Project

(3EL05)

**SUBMITTED BY:**

**Harsh Mehta (ID No. 19EL083)**

**IN PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE**

**OF**

**BECHELOR OF TECHNOLOGY**

**IN**

**Electronics Engineering**

****

**BIRLA VISHWAKARMA MAHAVIDALAYA**

**Engineering Collage, Vallabh Vidyanagar, 388120  
 [An Autonomous Institution]  
Electronics Department**

**Lab Teacher Course Coordinator & Guide**

**Prof. Anita Bhatt**

**ELECTRONICS ENGG DEPARTMENT**

**BIRLA VISHWAKARMA MAHAVIDYALAYA ENGG COLLEGE V V NAGAR**

****

**CERTIFICATE**

This is to certify that M/r. MEHTA HARSH ANILKUMAR **ID No**. 19EL083, of B.Tech. (Electronics Engineering) SEM-VI has satisfactorily completed the term work of the subject **EMBEDDED AND IOT SYSTEM DESIGN LABORATORY (3EL05)** prescribed by BVM an Autonomous Institution during the Academic Year **2021-2022**

**Lab Teacher Course Coordinator & Guide**

**Prof. Anita Bhatt**

**ELECTRONICS ENGG DEPARTMENT**

## **Acknowledgment**

Firstly, we offer thanks to our guide and Course coordinator Prof. Anita Bhatt Electronics Department, Birla Vishvakarma Mahavidyalaya, our Lab teacher: Prof. Anita BhattLecturer, Birla Vishvakarma Mahavidyalaya, Electronics Department, for their invaluable support, guidance and advice given throughout this semester and for helping to establish our direction.

We express our deep and sincere sense of gratitude to specially,Prof. Anita BhattElectronics Department, Birla Vishvakarma Mahavidyalaya, who has given us invaluable support and given opportunities to learn and develop researcher skills and has been unending source of inspiration to us. We also thankful to all our other faculty members for their consistence support and guidance.

**Harsh Mehta**

**Table of Content**

Vision mission…………………………………………………….......I

Title page ……………………………………………………………. II

Certificates……………………………………………………………IV

Acknowledgements ………………………………………...................V

Table of Contents…………………………………………...................VI

Index…………………………………………………………………...VII

Abstract…………………………………………………………………IX

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page No.** |
| **1** | Introduction  Overview, Purpose |  |
| **2** | Working of the Project  Component description, Description of Hard ware, |  |
| **3** | Software part  Software description, Flow chart |  |
| **4** | Interfacing, Experiments, Results, Outcomes  Schematic diagram, Interfacing, Results |  |
| **5** | Hard ware part  Simulation of hardware |  |
| **6** | Features, Future Work, application, conclusion |  |
| **7** | References |  |
| **8** | Appendix Program Code |  |

**ABSTRACT**

The facial recognition is a very useful tool incorporated in many modern devices to detect human faces for tracking, biometric and to recognize human activities. In this project, I have used the OpenCV's Harr cascade classifiers for detecting human faces and pan/tilt servo mechanism to track the user's face using Arduino UNO.

**Gimble:**

Gimbals are typically used in aerial vehicles to attenuate

vibrations and stabilize the camera in the presence of angular

motion of the vehicle and other disturbances, through inertial

stabilization [1]–[3]. In addition, gimbals effectively augment

the Field of View (FOV) of the camera through angular

motion, which can either compensate for or be combined with

translational motion to achieve the desired shooting objective.

This ability to mitigate the FOV constraint becomes even more

critical when tracking of moving targets is involved [4]. In fact,

physical, autonomous target tracking using a camera/gimbal

combination is an extremely important functionality for vision-

enabled robotic systems, e.g., in autonomous cinematogra-

phy/intelligent shooting application

Gimbals are typically used in aerial vehicles to attenuate

vibrations and stabilize the camera in the presence of angular

motion of the vehicle and other disturbances, through inertial

stabilization [1]–[3]. In addition, gimbals effectively augment

the Field of View (FOV) of the camera through angular

motion, which can either compensate for or be combined with

translational motion to achieve the desired shooting objective.

This ability to mitigate the FOV constraint becomes even more

critical when tracking of moving targets is involved [4]. In fact,

physical, autonomous target tracking using a camera/gimbal

combination is an extremely important functionality for vision-

enabled robotic systems, e.g., in autonomous cinematogra-

phy/intelligent shooting application

Gimbals are typically used in aerial vehicles to attenuate

vibrations and stabilize the camera in the presence of angular

motion of the vehicle and other disturbances, through inertial

stabilization [1]–[3]. In addition, gimbals effectively augment

the Field of View (FOV) of the camera through angular

motion, which can either compensate for or be combined with

translational motion to achieve the desired shooting objective.

This ability to mitigate the FOV constraint becomes even more

critical when tracking of moving targets is involved [4]. In fact,

physical, autonomous target tracking using a camera/gimbal

combination is an extremely important functionality for vision-

enabled robotic systems, e.g., in autonomous cinematogra-

phy/intelligent shooting application

Gimbals are typically used in aerial vehicles to attenuate vibrations and stabilize the camera in the presence of angular motion of the vehicle and other disturbances, through inertial stabilization. In addition, gimbals effectively augment the Field of View (FOV) of the camera through angular motion, which can either compensate for or be combined with translational motion to achieve the desired shooting objective. This ability to mitigate the FOV constraint becomes even more critical when tracking of moving targets is involved. In fact, physical, autonomous target tracking using a camera/gimbal combination is an extremely important functionality for vision enabled robotic systems, e.g., in autonomous cinematography/intelligent shooting applications

**INTRODUCTION TO THE PROJECT**

**Computer Vision:**

First of all, to understand my project you have to understand computer vision. Face tracking and detection features in sequence is an important and fundamental problem in computer vision. This area of research has a lot of applications in face identification systems, model-based coding, gaze detection, human computer, interaction, teleconferencing, etc. human-computer interaction, teleconferencing, etc.

**Open CV**:

OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and a few C++ classes that implement some popular Image Processing and Computer Vision algorithms. OpenCV has cross-platform middle-to-high level API that consists of a few hundreds C functions. It does not rely on external libraries, though it can use some when it is possible. OpenCV is free for both non-commercial and commercial use. OpenCV provides transparent interface to Intel Integrated Performance Primitives (IPP). That is, it loads automatically IPP libraries optimized for specific processor at runtime, if they are available.

**Image Processing**

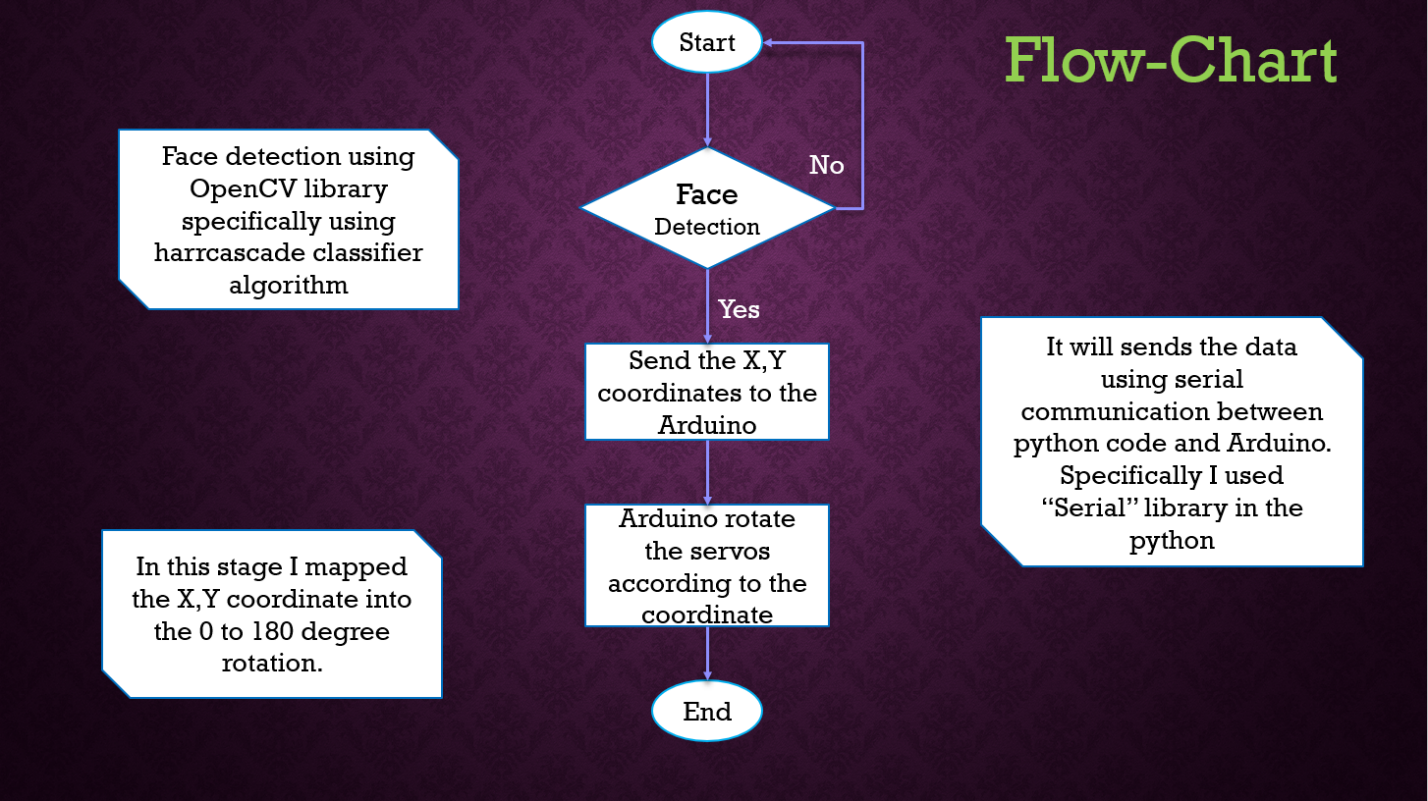
Computer manipulation of images. Some of the many algorithms used in image processing include convolution (on which many others are based), FFT, DCT, thinning (or skeletonisation), edge detection and contrast enhancement. These are usually implemented in software but may also use special purpose hardware for speed. Image processing contrasts with computer graphics, which is usually more concerned with the generation of artificial images, and visualisation, which attempts to understand (realworld) data by displaying it as an artificial image (e.g. a graph). Image processing is used in image recognition and computer vision. Silicon Graphics manufacture workstations which are often used for image processing. There are a few programming languages designed for image processing, e.g. CELIP, VPL, C++,PYTHON.

**Face Tracking:**

Face detection and tracking are important in video content analysis since the most important objects in most video are human beings. Research on face tracking and animation techniques has been improved due to its wide range of applications in security, entertainment industry, gaming, psychological facial expression analysis and human computer interaction. Recent advances in face video processing and compression have made face-to-face communication be practical in real world applications. However, higher bandwidth is still highly demanded due to the increasing intensive communication. Model based low bit rate transmission with high quality video offers a great potential to mitigate the problem raised by limited communication resources. However, after a decade’s effort, robust and realistic real time face tracking and generation still pose a big challenge. The difficulty lies in a number of issues including the real time face feature tracking under a variety of imaging conditions such as lighting variation, pose change, self-occlusion and multiple non-rigid features deformation and the real time realistic face modeling using a very limited number of feature parameters. Traditionally, the head motion is modeled as a 3D rigid motion with the local skin deformation, the linear motion tracking method cannot represent the rapid head motion and dramatic expression change accurately. The appearance-driven approach requires a significant number of training data to enumerate all the possible appearances of features. The model-based approach assumes the knowledge of a specific object is available, meanwhile the requirement of frontal facial views and constant illumination limited its application. All above tracking methods have shown certain limitations for accurate face feature tracking under complex imaging conditions. Different types of facial features, like skin color, edges, feature points, motion, have been used for face tracking. Skin color is tried for tracking face motion in X, Y direction and out-of-plane rotation in. It is often too simple to encode structural knowledge of face, it is thus good for coarse face tracking. An optical flow field has been adopted for face tracking. Dense motion information makes face tracking easier. A major constraint is that optical flow estimation is subject to the aperture effect and usually does not allow big movement. Salient facial feature points are better choice for accurate face tracking. The main shortcoming is that tracking of point features is easily impaired by noise and often the face appearing in video should be large enough to facilitate tracking. Face tracking can serve as a front end to further analysis modules, such as face recognition, face expression analysis, gaze tracking, and lip reading. Face tracking is also a core component to enable the computer to see the computer user in a Human Computer Interface system.

**INTRODUCTION WITH FLOW CHART:**

The block diagram of the project and the description of the blocks in the block diagram are considered. Block diagram is shown below.



**HARDWARE COMPONENTS**

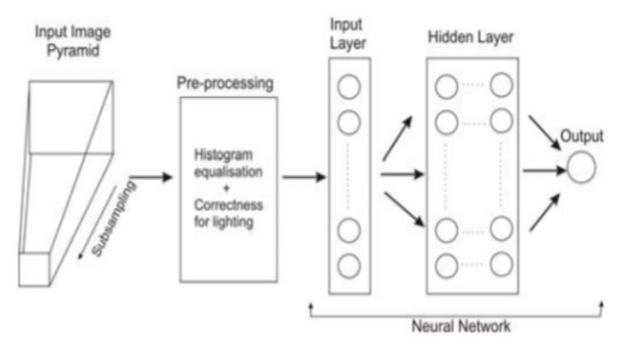
* Arduino Uno
* Servo Motors
* USB Camera
* Bread Board
* 3D printed gimbal Components
* Screw Driver
* Jumper Wires

**SOFTWARE COMPONENTS**

* Arduino IDE
* PyCharm IDE

**INTRODUCTION TO FACE DETECTION:**

Face detection involves separating image windows into two classes; one containing faces (tarning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin color and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

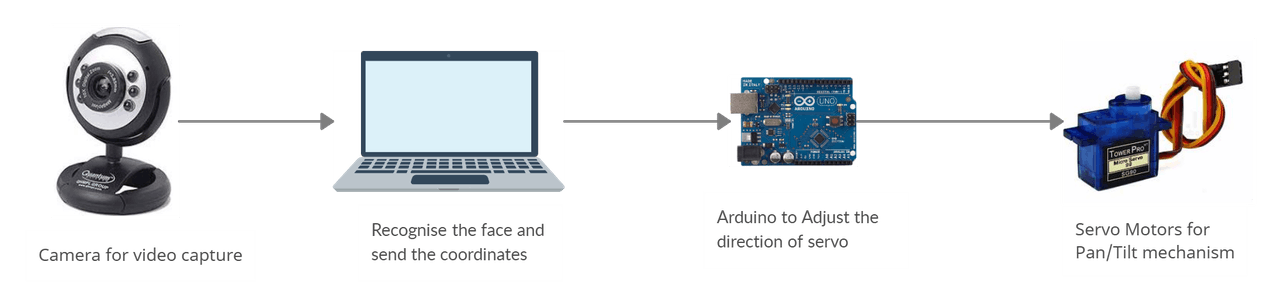


The face detection system can be divided into the following steps: -

1. **Pre-Processing**: To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.
2. **Classification**: Neural networks are implemented to classify the images as faces or nonfaces by training on these examples. We use both our implementation of the neural network and the Matlab neural network toolbox for this task. Different network configurations are experimented with to optimize the results.
3. **Localization**: The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:- Position Scale Orientation Illumination

**WORKING OF OUR PROJECT**

* Facial detection identifies and localizes human faces and ignores any background objects such as curtain, windows, trees, etc. OpenCV uses Harr cascade of classifiers where each frame of the video is passed through stages of classifiers and if the frame passes through all the classifiers, the face is present else the frame is discarded from the classifier i.e. the face is not detected.
* The OpenCV returns the cartesian coordinates of the image upon detection along with the height and width. From these coordinates, the center coordinates of the image can be calculated using x+width/2 and y+height/2



* These coordinates are passed to the Arduino UNO using the pyserial library when the face is detected. The servo's connected to the Arduino provides a pan/tilt mechanism where the camera is connected to one of the servo. When the co-ordinates of the face is away from the center, then the servo will align by 2 degrees(increment or decrement)to bring it towards the center of the screen.

### **Detecting the face**

* I have used 'haarcascade\_frontalface\_default.xml' which is a pre- trained model for detecting human faces and can be downloaded from Git-Hub. Upon downloading, the xml file can be loaded using *cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')*
* The function used for face detection *is cv2.CascadeClassifier.detectMultiScale()* with the 'scale factor' value as 1.1(default) and 'minNeighbour' value as 6. This returns the cartesian coordinates of the image along with the height and width. Increasing the 'minNeighbour' can improve facial detection but sacrifices in execution speeds which would lead to a delayed response from the servo. Thus, the value 6 seemed optimal.
* In-order to have a precise facial recognition, a plain background would be recommended as I faced some false detection due to the curtains in the background.

### **Calculating the Coordinates**

* OpenCV returns the face coordinates in terms of pixel values. By default, the video resolution is set to 640\*480. The coordinates describe the top-left pixel values(x and y) along with the height and width. I have used the center coordinates of the face for reference and can be calculated using x+width/2 and y+height/2 and can be seen as a green dot. These coordinates are sent to the arduino for moving the angle of the camera.
* The square in the center of the frame in white describes the region within which the center of the face i.e the green dot must be. If it is outside the squared region when the face is moved, then the servo will align the camera to bring it inside the region.

### **Sending Serial data to Arduino**

* I found this part challenging as I tried many ways to send the coordinates sequentially to the arduino but the response was slow. After spending hours figuring it out, I began looking for similar projects online until I found this project. It made me aware of the Serial function Serial.parseInt() which takes integer inputs from an incoming serial of bytes. My approach towards sending the serial data is similar to the one used in that project.
* The python sends the center coordinates in a single string. For ex: "X100Y200", the value 100 after X represents center x-coordinate and 200 represents center-y coordinate.

### **Arrangement of Servo**

* I have attached the horizontal moving servo on the shaft of the vertical moving servo in which the camera is mounted. All the attachments are made using simple 3-D printed materials.

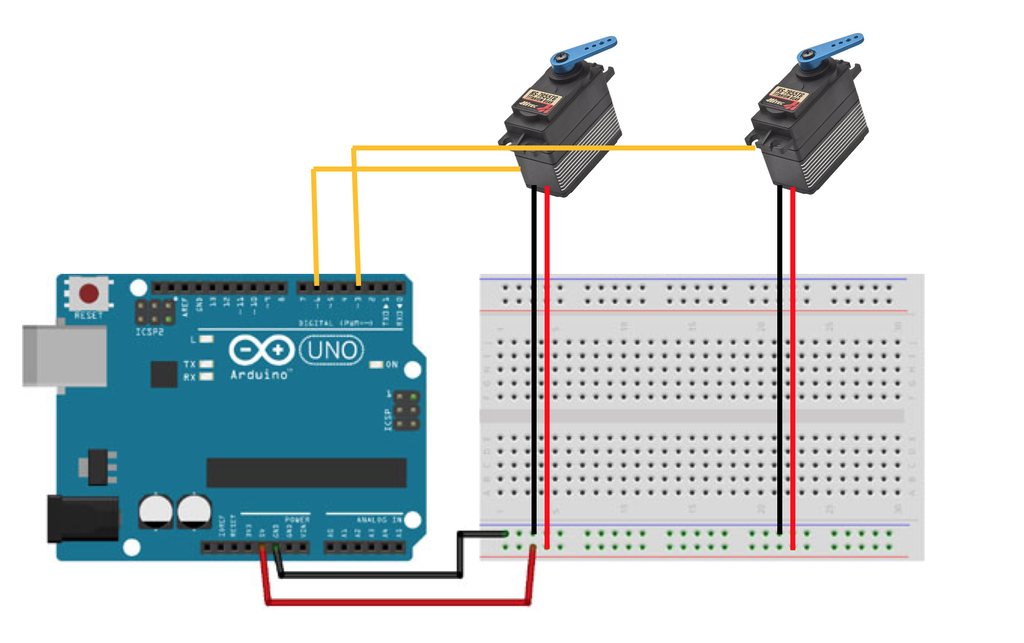
### **Libraries and Installations**

* This project requires **pyserial**and **opencv** libraries which I have downloaded using pip. My current python and OpenCV version is 3.8 and 4.4.0, so make sure you have a similar or a higher version. Also make sure that the XML file for face detection is saved in the same directory which contains the python script.

Here is a video captured by the camera tracking my face. (To view double click on this Icon)



**Schematic Diagram**



**Hardware Implementation**





**WORKING VIDEO OF PROJECT:**

Double click on icon to open the video.



**CONCLUSIONS:**

Face recognition and tracking is a still challenging problem in the field of computer vision. It has received a great deal of attention over the past years because of its several applications in various domains. Although there is strong research effort in this area, face recognition and tracking system are far from ideal to perform adequately in all situations from real world.

**APPLICATION**

* Person Monitoring in Riots
* Easy to record the lectures (NPTEL Video Capturing)
* Phycological facial expression analysis
* Entertainment Industries

**FUTURE WORK**

After this project I have planned to implement 3rd axis (Z axis) within this project only. So from this my project running on 3-axis (X,Y,Z).

**References**

[1] Z. Hurak and M. Rezac, “Image-based pointing and tracking for inertially stabilized airborne camera platform,” IEEE Transactions on Control Systems Technology, vol. 20, no. 5, pp. 1146–1159, Sept. 2012.

[2] F. Königseder, W. Kemmetmüller, and A. Kugi, “Attitude estimation using redundant inertial measurement units for the control of a camera stabilization platform,” IEEE Transactions on Control Systems Technology, vol. 24, no. 5, pp. 1837–1844, Sept. 2016.

[3] M. K. Masten, “Inertially stabilized platforms for optical imaging systems,” IEEE Control Systems Magazine, vol. 28, no. 1, pp. 47–64, Feb. 2008.

[4] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, “Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor,” IEEE Robotics and Automation Letters, vol. 2, no. 3, pp. 1762–1769, July 2017.

[5] A. Torres-González, J. Capitán, R. Cunha, A. Ollero, and I. Mademlis, “A multidrone approach for autonomous cinematography planning,” Iberian Robotics Conference (ROBOT), 2017.

**APPENDIX PROGRAM CODE**

The program code is written in the embedded c and python language.

**Arduino Code:**

#include<Servo.h>

Servo x, y;

int width = 640, height = 480; // total resolution of the video

int xpos = 90, ypos = 90; // initial positions of both Servos

void setup() {

Serial.begin(9600);

x.attach(9);

y.attach(10);

// Serial.print(width);

//Serial.print("\t");

//Serial.println(height);

x.write(xpos);

y.write(ypos);

}

const int angle = 2; // degree of increment or decrement

void loop() {

if (Serial.available() > 0)

{

int x\_mid, y\_mid;

if (Serial.read() == 'X')

{

x\_mid = Serial.parseInt(); // read center x-coordinate

if (Serial.read() == 'Y')

y\_mid = Serial.parseInt(); // read center y-coordinate

}

/\* adjust the servo within the squared region if the coordinates

is outside it

\*/

if (x\_mid > width / 2 + 30)

xpos += angle;

if (x\_mid < width / 2 - 30)

xpos -= angle;

if (y\_mid < height / 2 + 30)

ypos -= angle;

if (y\_mid > height / 2 - 30)

ypos += angle;

// if the servo degree is outside its range

if (xpos >= 180)

xpos = 180;

else if (xpos <= 0)

xpos = 0;

if (ypos >= 180)

ypos = 180;

else if (ypos <= 0)

ypos = 0;

x.write(xpos);

y.write(ypos);

}

}

**Python Code:**

import cv2  
import serial,time  
face\_cascade= cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')  
cap=cv2.VideoCapture(1)  
ArduinoSerial=serial.Serial('com3',9600,timeout=0.1)  
time.sleep(1)  
  
while cap.isOpened():  
 ret, frame= cap.read()  
 frame=cv2.flip(frame,1)  
 gray = cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)  
 faces= face\_cascade.detectMultiScale(gray,1.1,6) #detect the face  
 for x,y,w,h in faces:  
 string='X{0:d}Y{1:d}'.format((x+w//2),(y+h//2))  
 print(string)  
 ArduinoSerial.write(string.encode('utf-8'))  
 #plot the center of the face  
 cv2.circle(frame,(x+w//2,y+h//2),2,(0,255,0),2)  
  
 cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),3)  
  
 cv2.imshow('img',frame)  
  
 # press q to Quit  
 if cv2.waitKey(10)&0xFF== ord('q'):  
 break  
cap.release()  
cv2.destroyAllWindows()