Pan & Tilt servo control

With face detection app

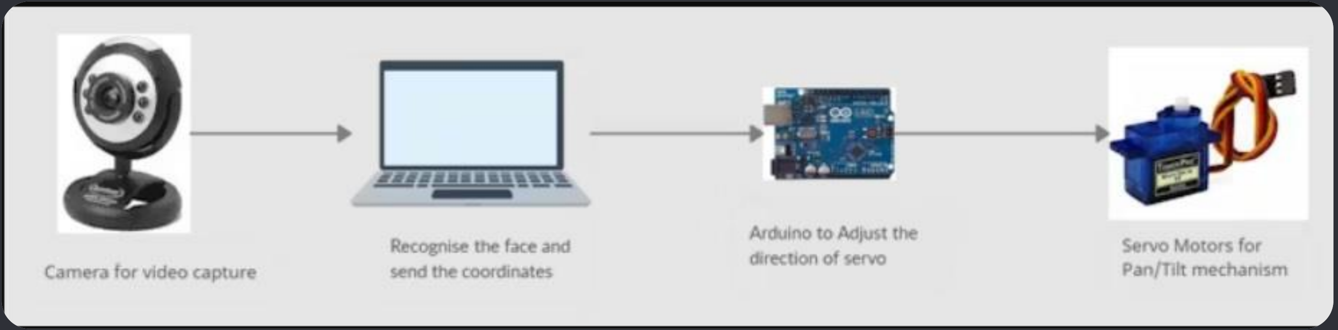
(Face Tracker)

Table of contents

1 Introduction

Introduction

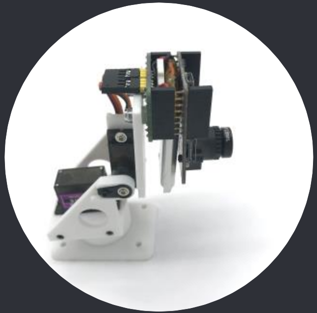
The facial recognition is a very useful tool incorporated in many modern devices to detect human faces for tracking, biometric and to recognize human activities. In this project, we have used the OpenCV's Harr cascade classifiers for detecting human faces and **pan/tilt servo mechanism** to track the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



2

Components

Components



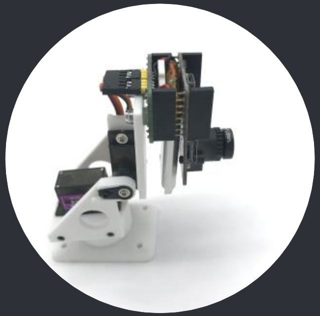
Pan-tilt mechanism

Arduino NANO

2 x Micro servos

Web Camera

Components



Pan-tilt mechanism

- A pan-tilt mechanism's mechanical components allow the system to achieve controlled motion along two axes – pan (horizontal) and tilt (vertical).

Components

Arduino NANO

- The Arduino Nano is a small, powerful, and versatile microcontroller board based on the ATmega328P microcontroller.
- **Operating Voltage: 5V**
  Input Voltage: 7-12V (recommended), 6-20V (limits)
  Digital I/O Pins: 22 (14 of which can function as PWM outputs)

Components

Micro servos

Micro servo motors are miniature versions of the standard servo motors, offering similar functionality but in a smaller form factor. They are commonly used in applications where space constraints and low weight are critical, such as small robotics, model aircraft, and animatronics.

Components

Web cam

A webcam is a digital camera that streams or captures images and video in real-time. Using the proceeded video in detection

Why Micro Servo Motors

- Compact size: Ideal for our project's limited space requirements, ensuring a sleek and unobtrusive design.
- Precise control: Delivers accurate and stable pan and tilt movements for effective face tracking.
- Quick response: Enables real-time adjustments to maintain alignment with detected faces.
- Simple integration: Easily incorporated into our system's mechanical design and electronic control.
- Reliability: Provides consistent and dependable performance throughout the project's operation.

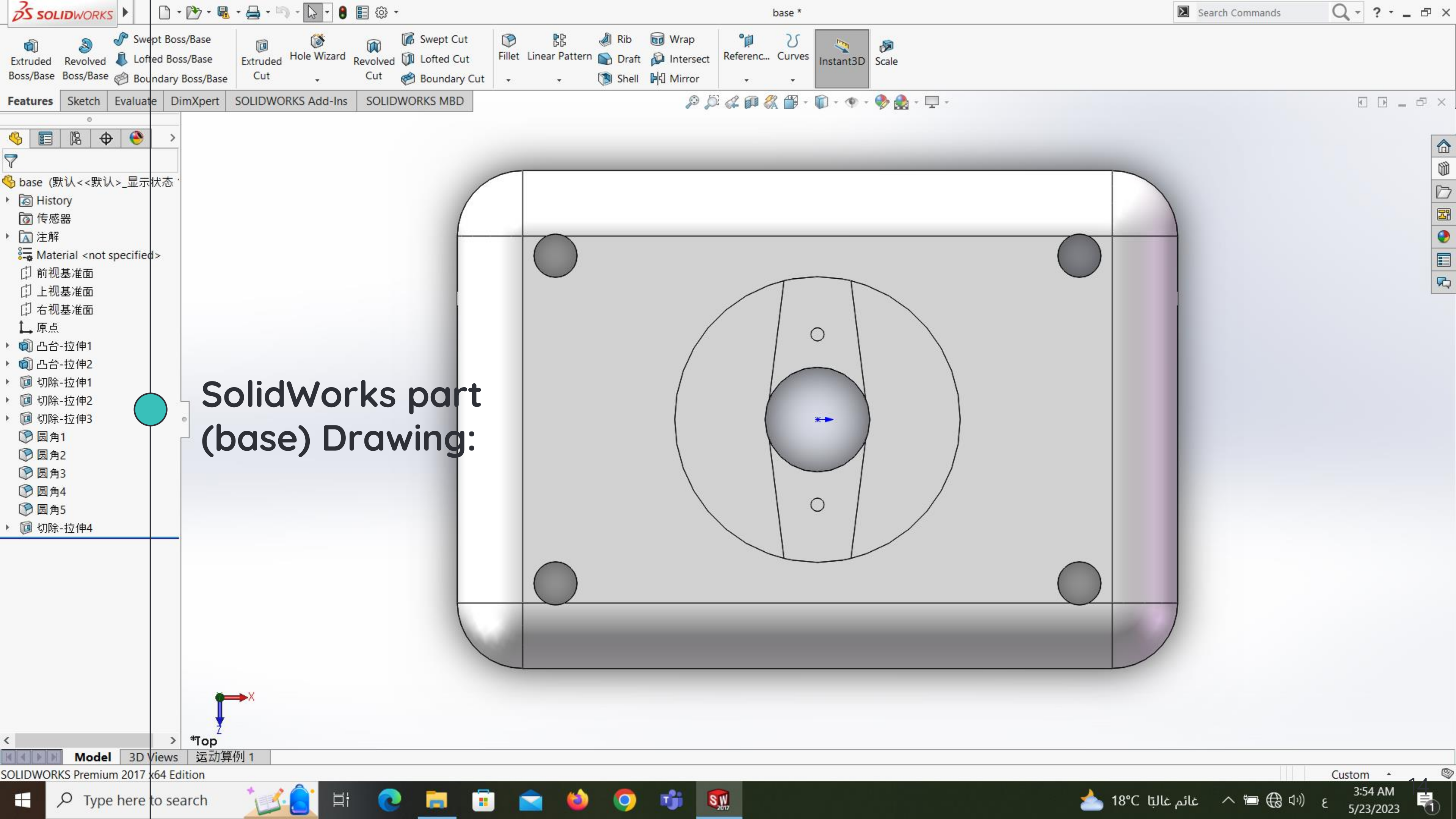
3 **Mechanical** Design

Mechanical Design

In this project, we will use SolidWorks for mechanical design to create a pan and tilt servo motor control system with face detection. The process will involve:

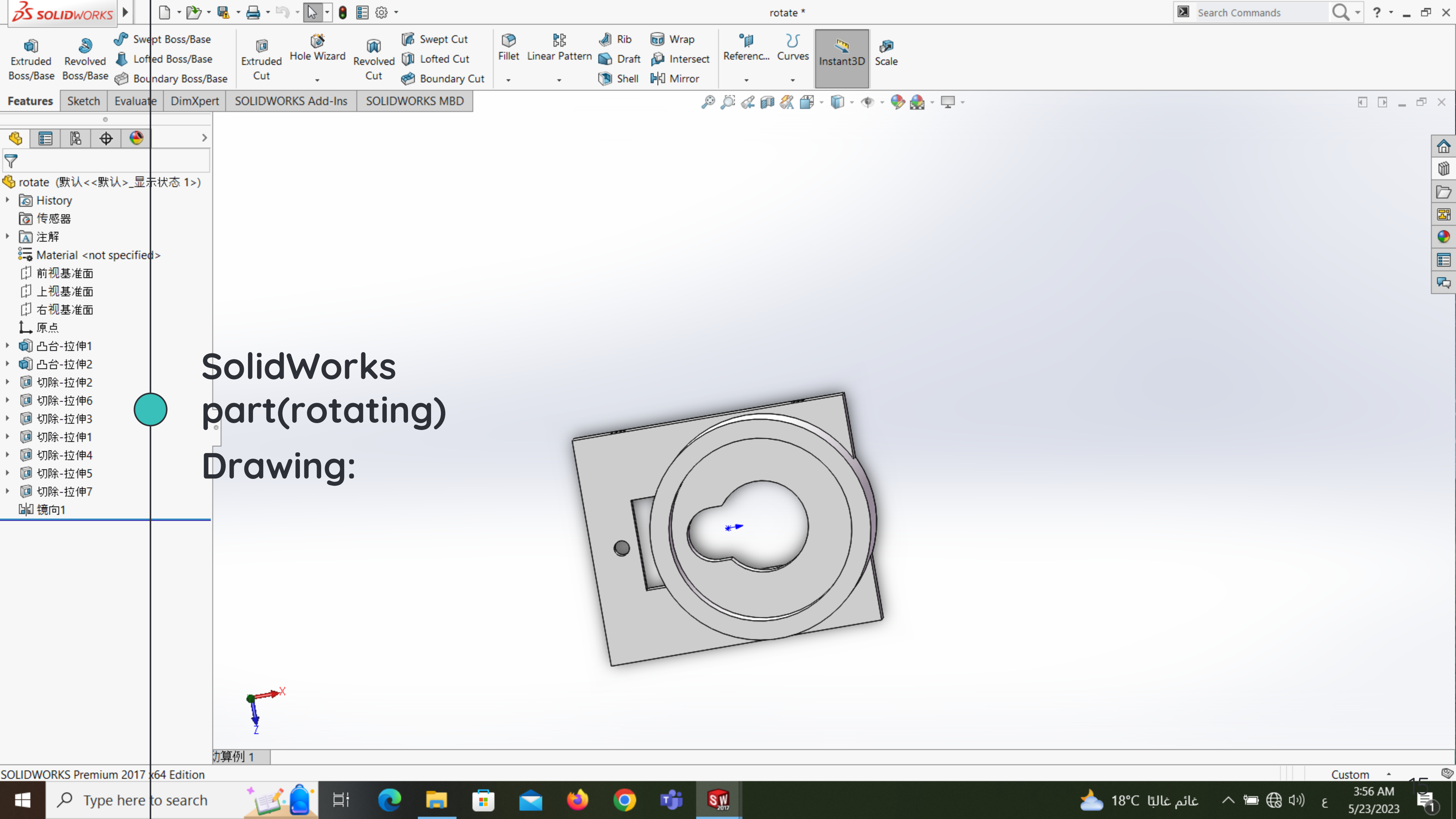
- Defining specifications for components, such as servo mounts, brackets, and structure.
- Creating 3D models of components in SolidWorks.
- Assembling the components in the software to form a complete system.

SolidWorks part (base) Drawing:

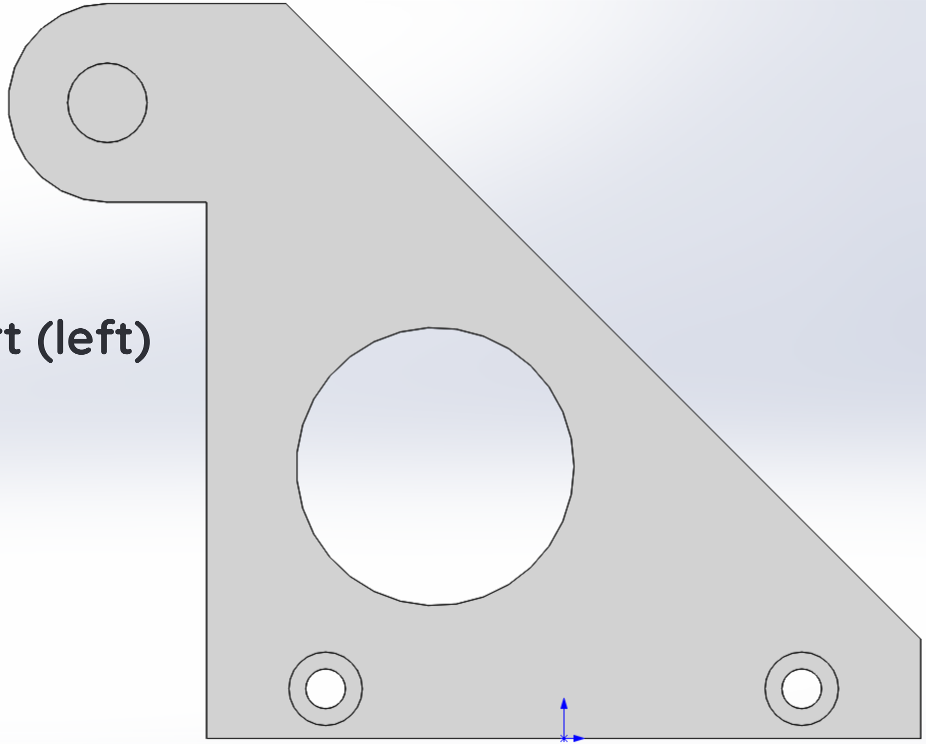


SolidWorks part(rotating)

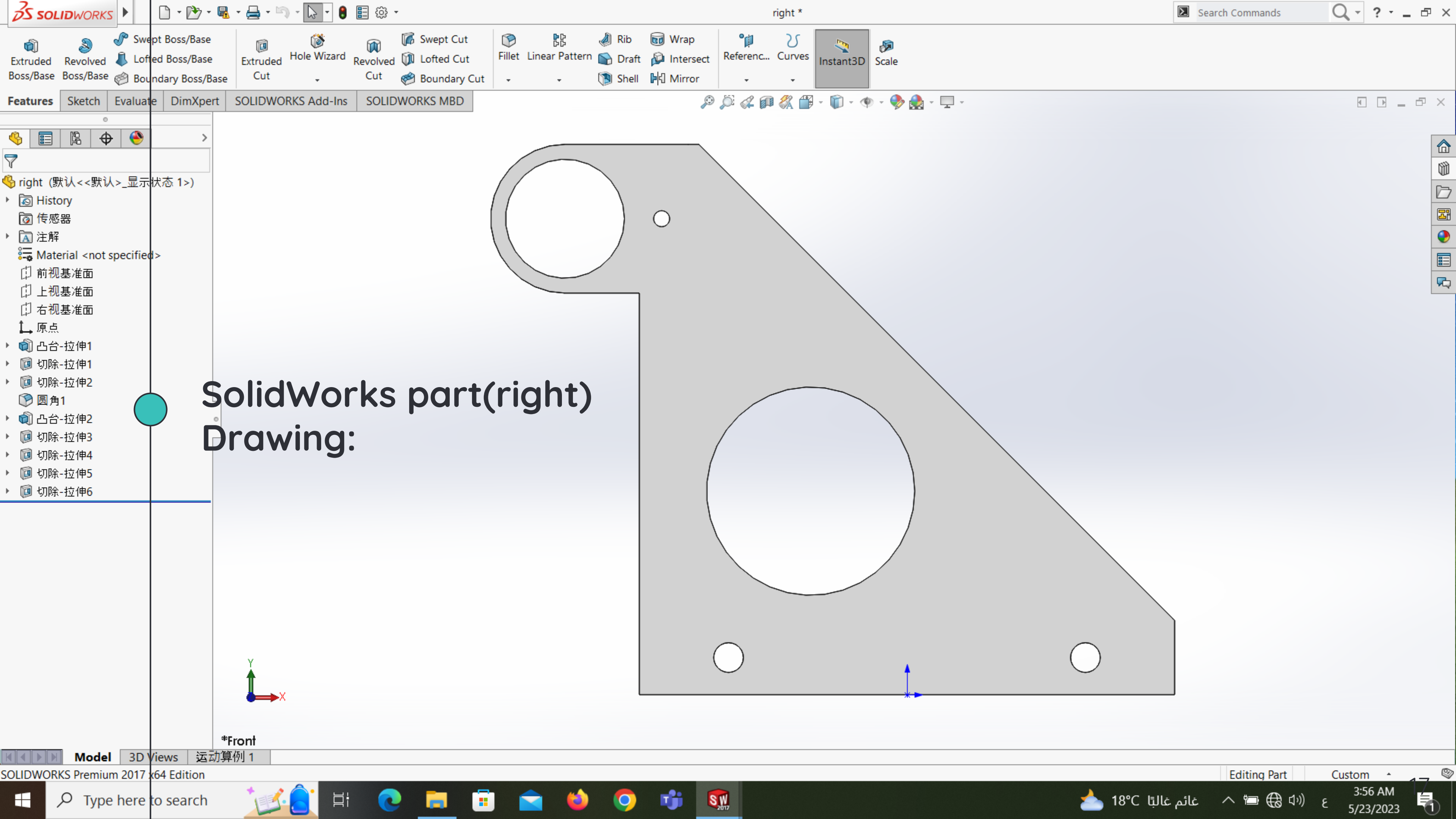
Drawing:



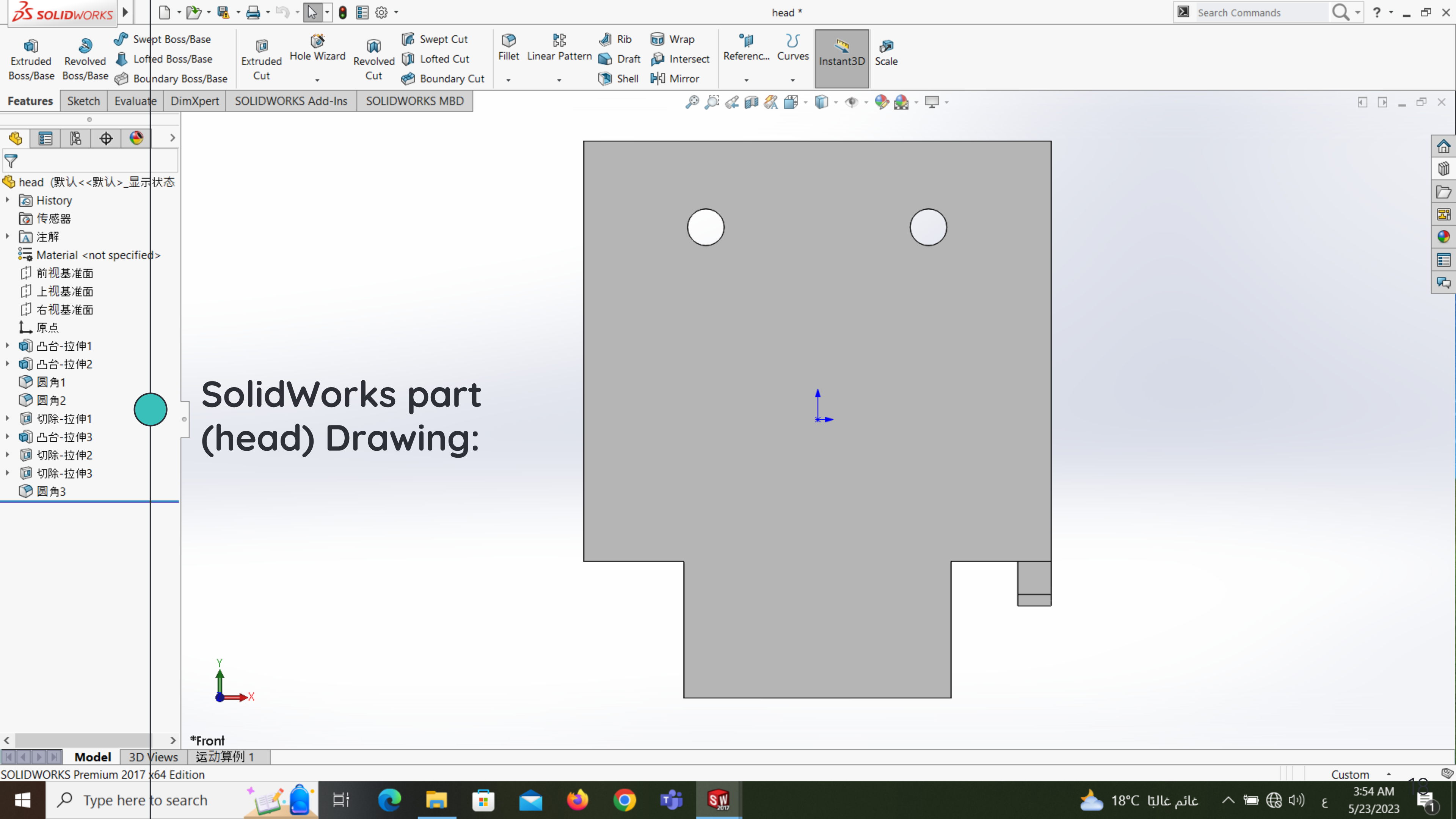
SolidWorks part (left) Drawing:



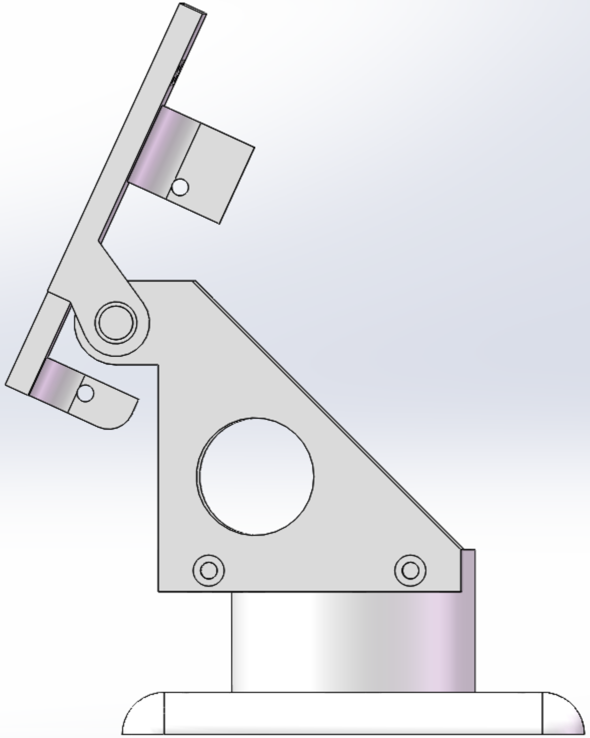
SolidWorks part(right) Drawing:



SolidWorks part (head) Drawing:



SolidWorks Assembly Drawing:

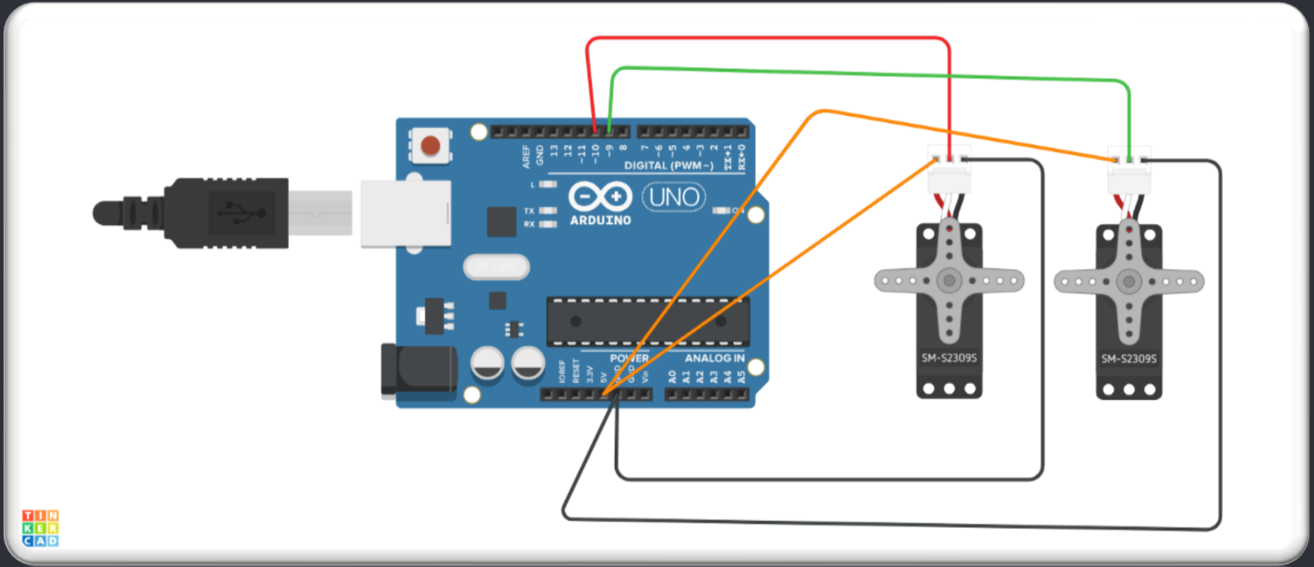


4 **Wiring** & Data Processing

Wiring & Data Processing

In our pan and tilt control system with face detection, we'll create a TinkerCAD circuit to simulate the wiring and connections between components

Schematic



OpenCV

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

Serial communication

Connection: USB cable between Arduino and device

Baud rate: Match for accurate data transfer

Data format: Consistent format for face position info

Data transfer: Continuously send coordinates and adjust servo motors

Error handling: Mechanisms to handle communication issues

Arduino Code

```
#include <VarSpeedServo.h>
VarSpeedServo servo1; VarSpeedServo servo2;
String inputString = "";         // a string to hold incom
unsigned int cont=0;

void setup()
{
  servo1.attach(9);
  servo2.attach(10);

  Serial.begin(250000);
  Serial.println("Ready");
}


void loop()
{

  signed int vel;
  unsigned int pos;

  if (Serial.available())
  {
    inputString = Serial.readStringUntil('!');
    vel = inputString.toInt();

    if(inputString.endsWith("x"))
    {
```

```
      if (vel > 2)
        servo1.write(180, vel, false);
      else if (vel < -2)
        servo1.write(0, -vel, false);
      else
      {
        pos = servo1.read();
        servo1.write(pos, 255, false);
      }
    }
    else if(inputString.endsWith("y"))
    {
      if (vel > 2)
        servo2.write(180, vel, false);
      else if (vel < -2)
        servo2.write(0, -vel, false);
      else
      {
        pos = servo2.read();
        servo2.write(pos, 255, false);
      }
    }
    else if(inputString.endsWith("o"))
    {
      cont++;
      if (cont >= 100)
      {
        pos = servo1.read();
        servo1.write(90, 20, true);
        pos = servo2.read();
```

Arduino Code

```
#include <VarSpeedServo.h>
VarSpeedServo servo1; VarSpeedServo servo2;
String inputString = "";         // a string to hold incom
unsigned int cont=0;

void setup()
{
  servo1.attach(9);
  servo2.attach(10);

  Serial.begin(250000);
  Serial.println("Ready");
}


void loop()
{

  signed int vel;
  unsigned int pos;

  if (Serial.available())
  {
    inputString = Serial.readStringUntil('!');
    vel = inputString.toInt();

    if(inputString.endsWith("x"))
    {
```

```
  if (vel > 2)
    servo1.write(180, vel, false);
  else if (vel < -2)
    servo1.write(0, -vel, false);
  else
  {
    pos = servo1.read();
    servo1.write(pos, 255, false);
  }
}
else if(inputString.endsWith("y"))
{
  if (vel > 2)
    servo2.write(180, vel, false);
  else if (vel < -2)
    servo2.write(0, -vel, false);
  else
  {
    pos = servo2.read();
    servo2.write(pos, 255, false);
  }
}
else if(inputString.endsWith("o"))
{
  cont++;
  if (cont >= 100)
  {
    pos = servo1.read();
    servo1.write(90, 20, true);
    pos = servo2.read();
```

```
{
  cont++;
  if (cont >= 100)
  {
    pos = servo1.read();
    servo1.write(90, 20, true);
    pos = servo2.read();
    servo2.write(70 , 20, true);
    cont = 0;

  }
  else
  {
    pos = servo1.read();
    servo1.write(pos, 255, false);
    pos = servo2.read();
    servo2.write(pos, 255, false);
  }


}
inputString = "";

  }
}
```

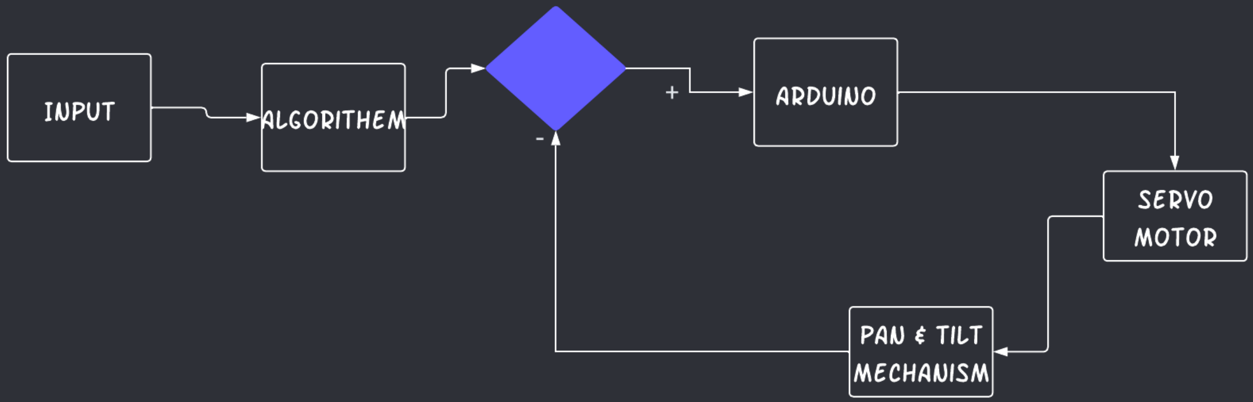
Python Code

```
import cv2
import serial
import numpy as np

def set_res(cap, x,y):
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, int(x))
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, int(y))

ser = serial.Serial('COM3', 250000)

cap = cv2.VideoCapture(1)

frame_w = 640
frame_h = 480
set_res(cap, frame_w,frame_h)

# Create the haar cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    cap.read()
    #cv2.imshow('original', frame)

    frame=cv2.flip(frame,1)
    #cv2.imshow('flipped', frame)

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    faces = np.array([])
    faces = face_cascade.detectMultiScale( gray,1.1,4)
        #flags = cv2.CV_HAAR_SCALE_IMAGE)


    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


    if ([i for i in faces]):
        face_center_x = faces[0,0]+faces[0,2]/2
        face_center_y = faces[0,1]+faces[0,3]/2
        #print(faces)
        err_x = 30*(face_center_x - frame_w/2)/(frame_w/2)
        err_y = 30*(face_center_y - frame_h/2)/(frame_h/2)
        ser.write((str(err_x) + "x!").encode())
        ser.write((str(err_y) + "y!").encode())
        print("X: ",err_x," ","Y: ",err_y)
    else:
        ser.write("o!".encode())

ser.close()
cap.release()
cv2.destroyAllWindows()
```

The Application

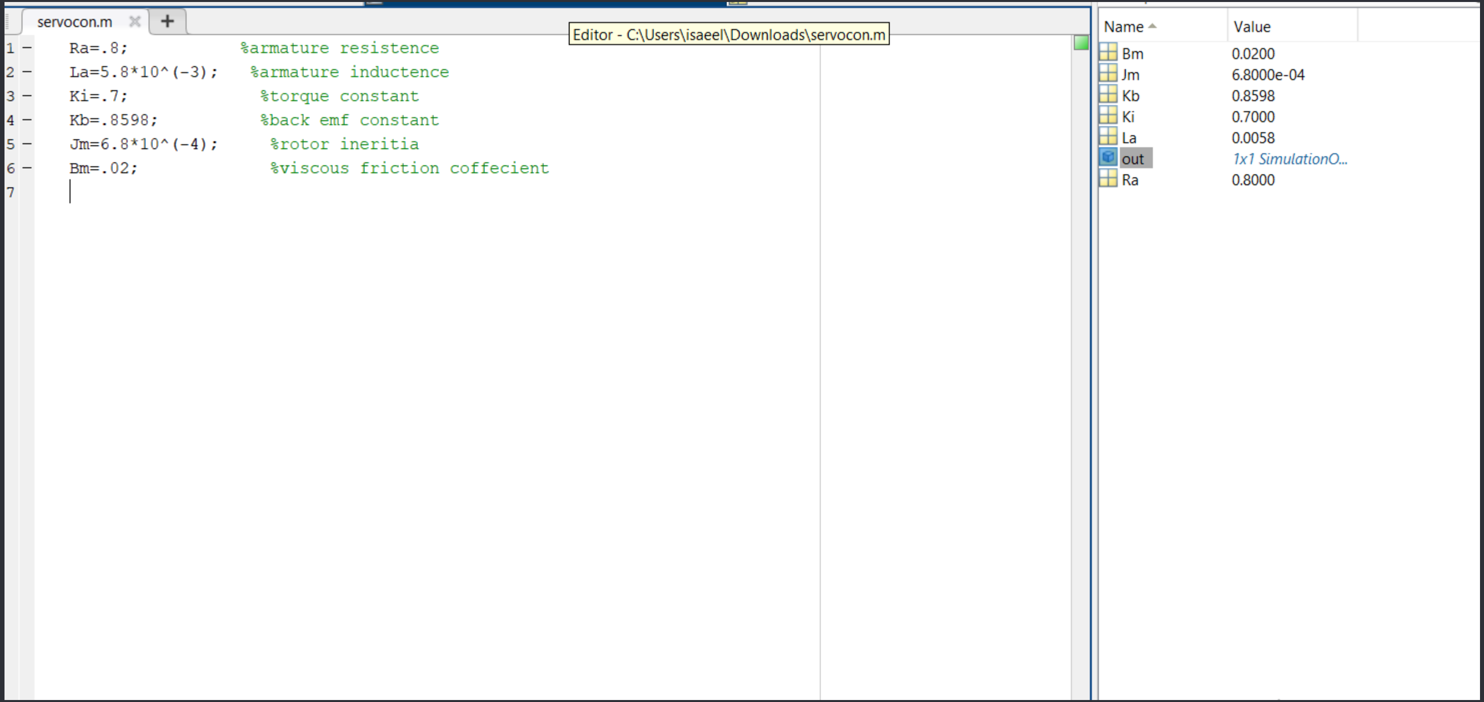
System block diagram



5 MATLAB Simulation

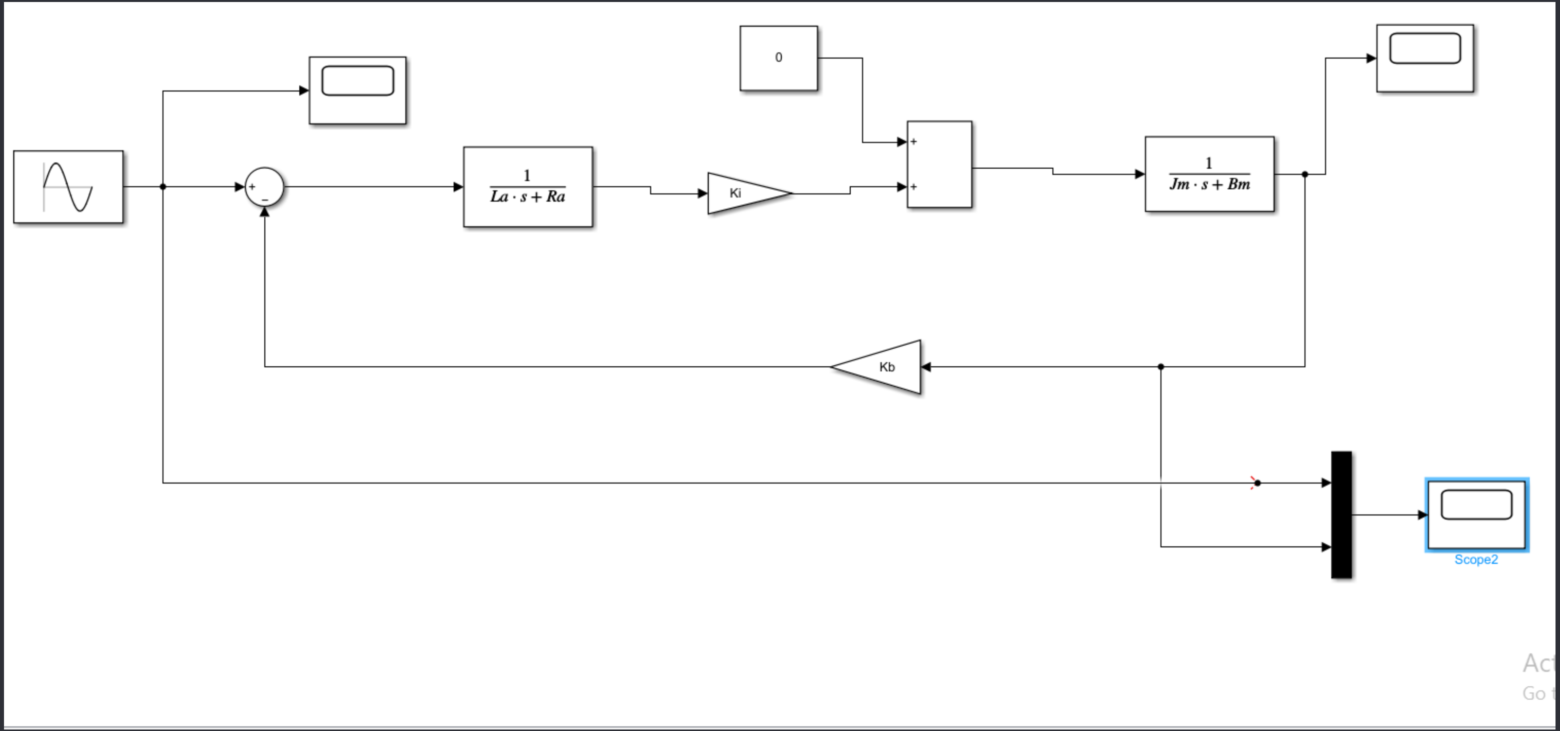
MATLAB Simulation

variables



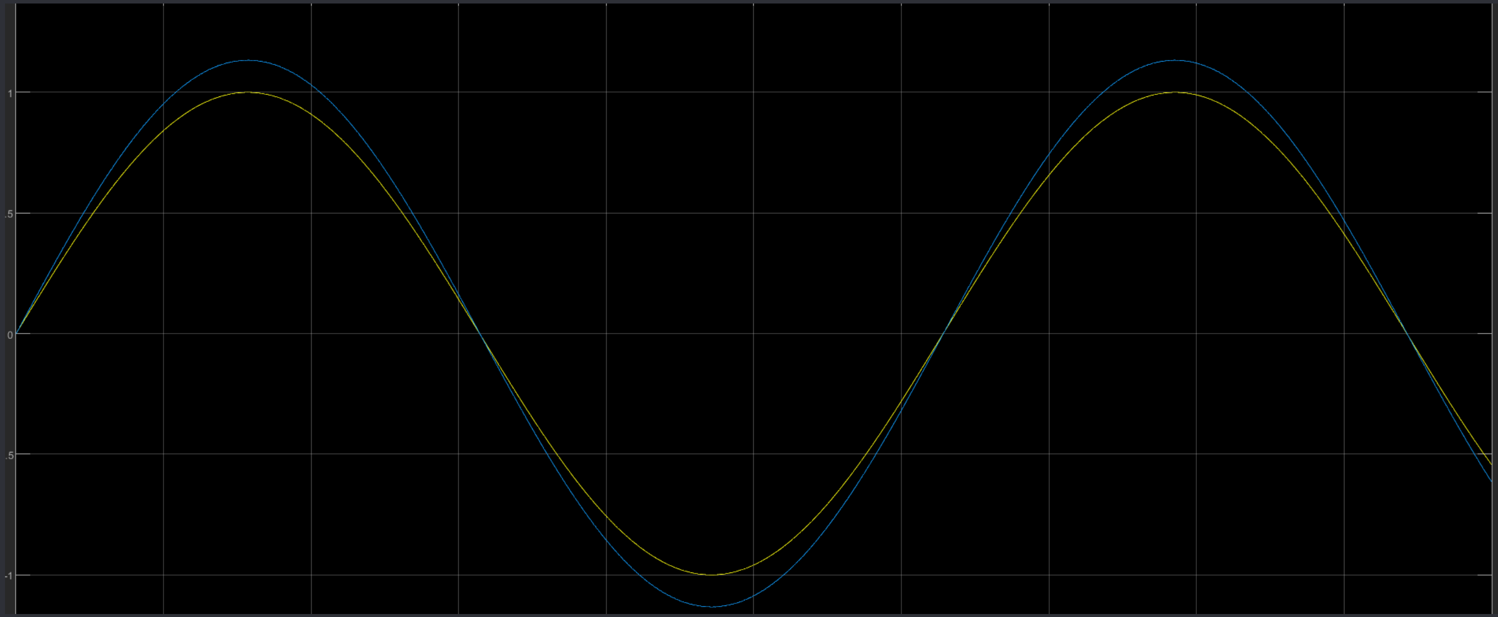
MATLAB Simulation

ang vel



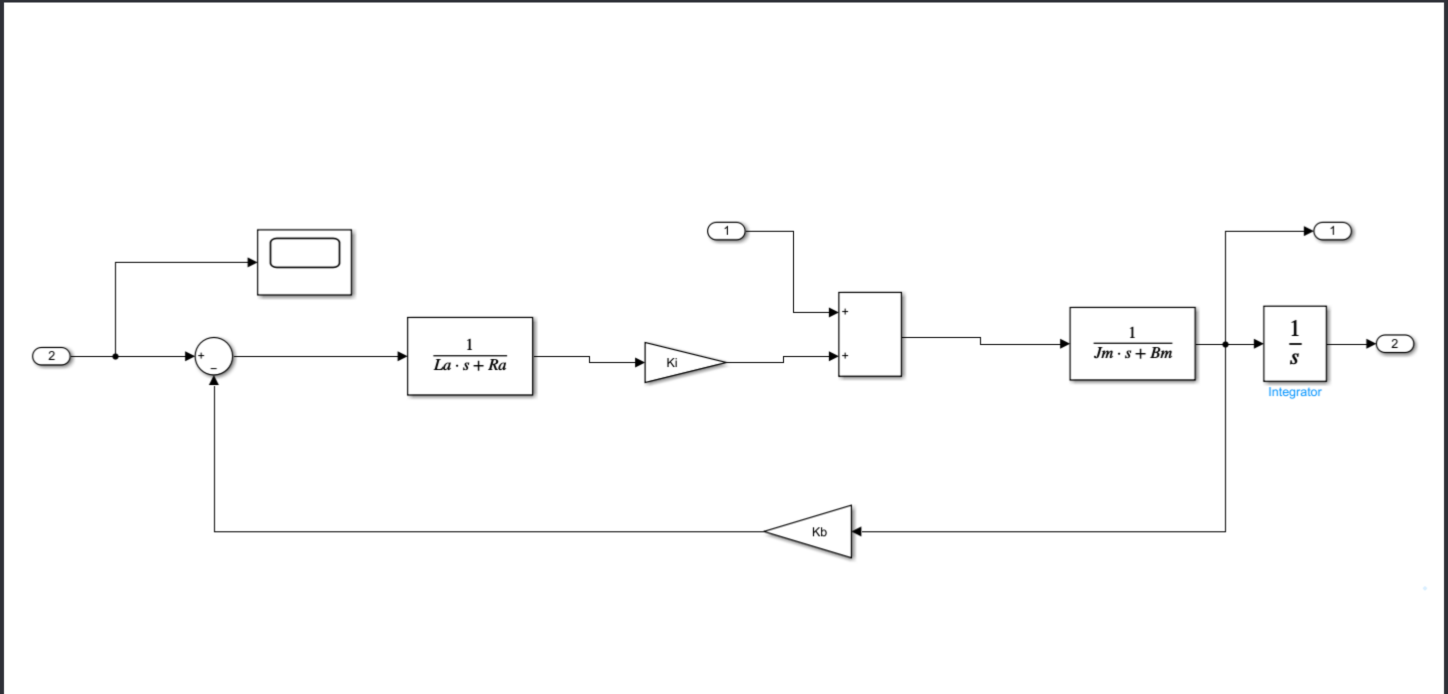
MATLAB Simulation

ang vel output



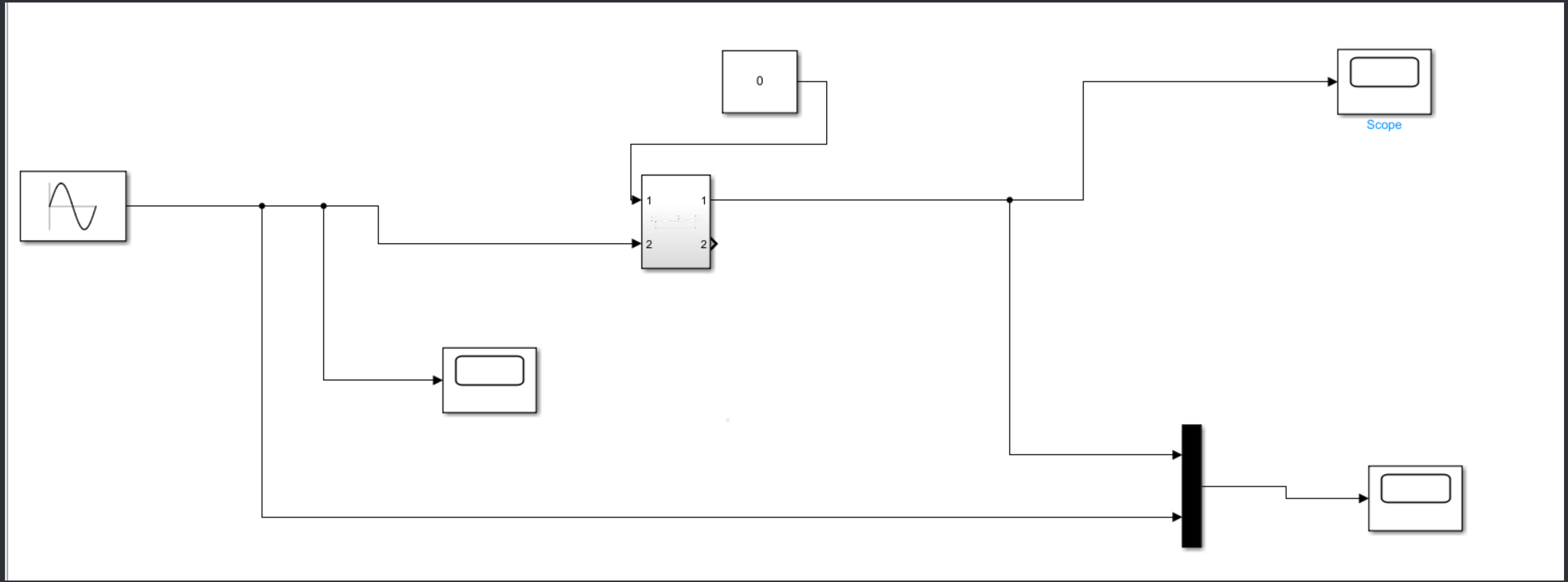
MATLAB Simulation

displacement



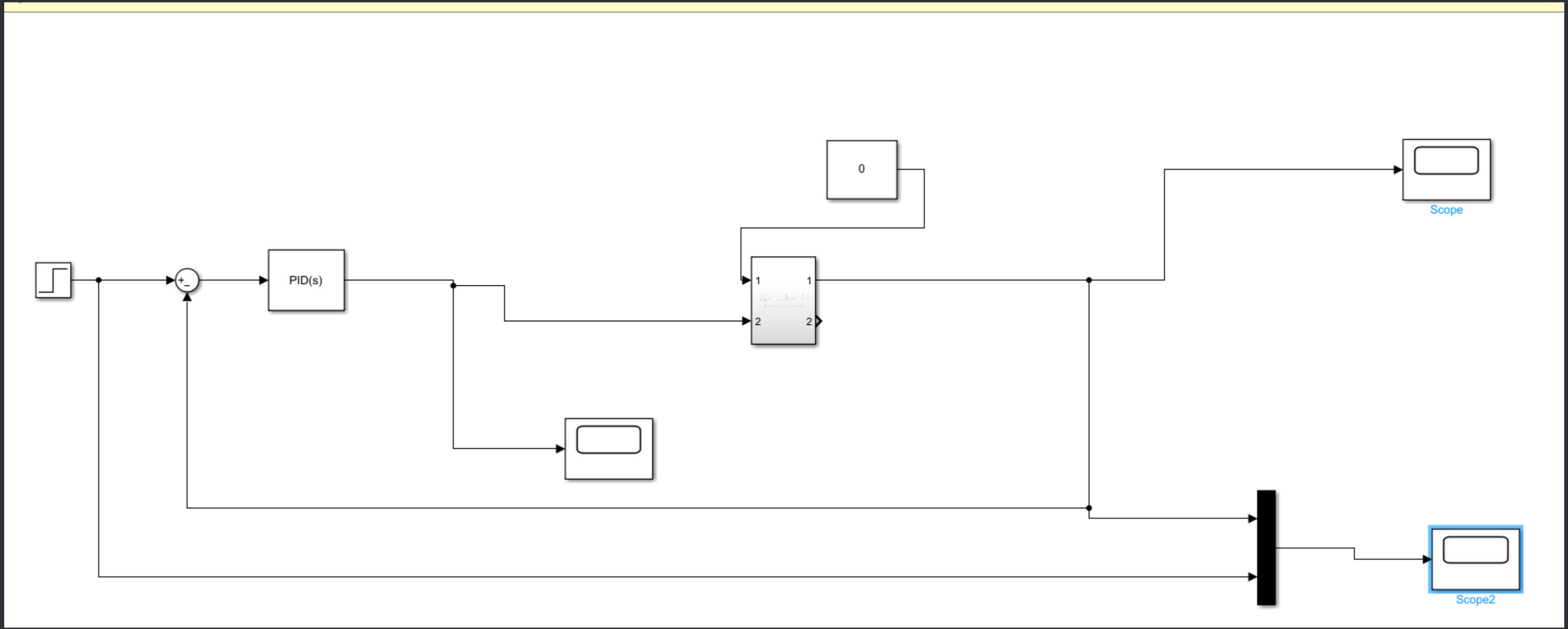
MATLAB Simulation

displacement



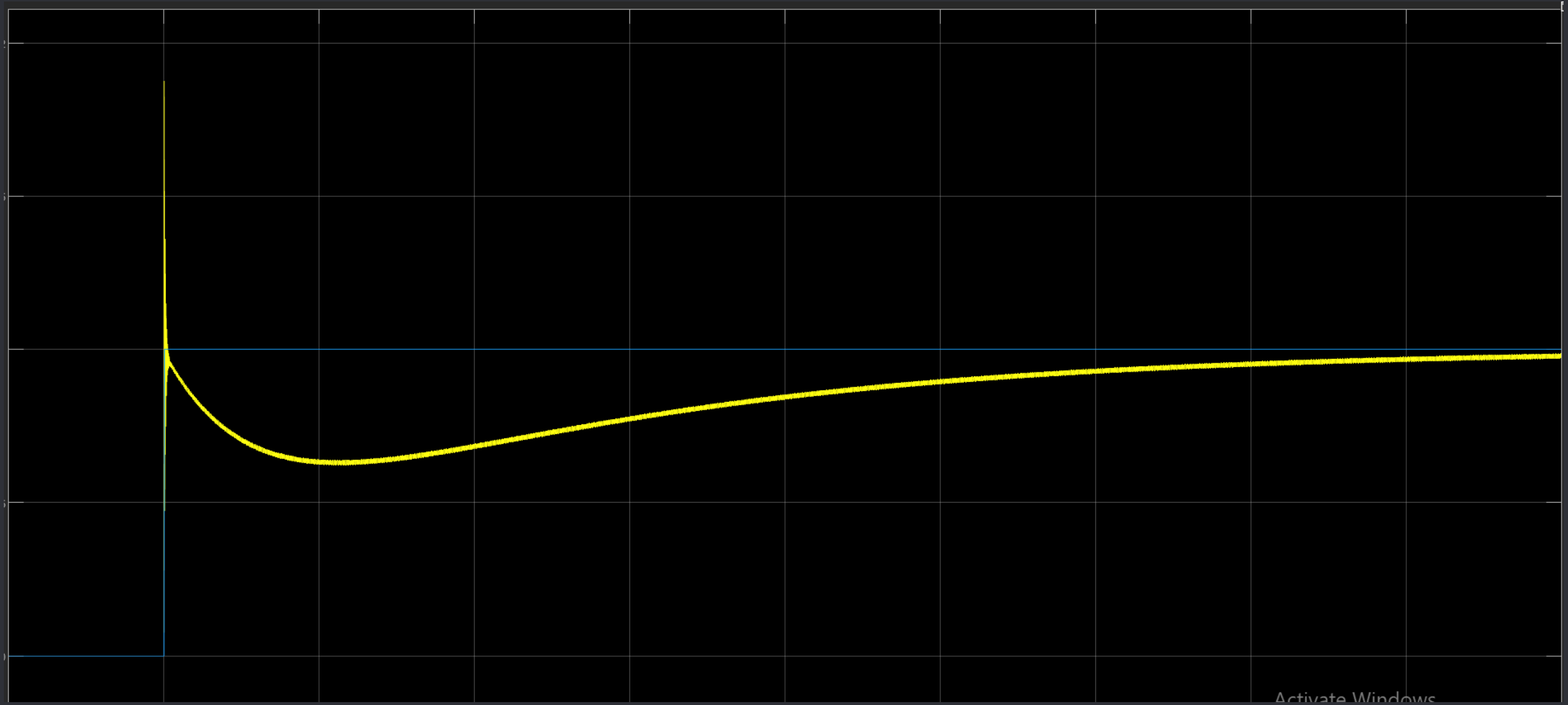
MATLAB Simulation

PID



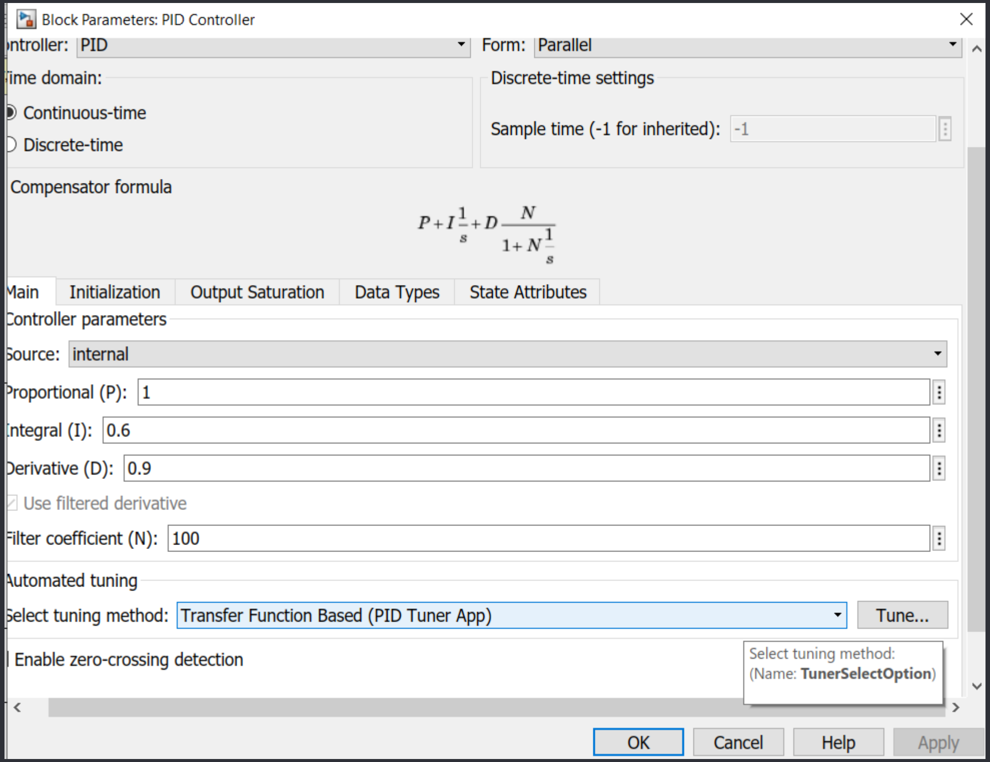
MATLAB Simulation

PID



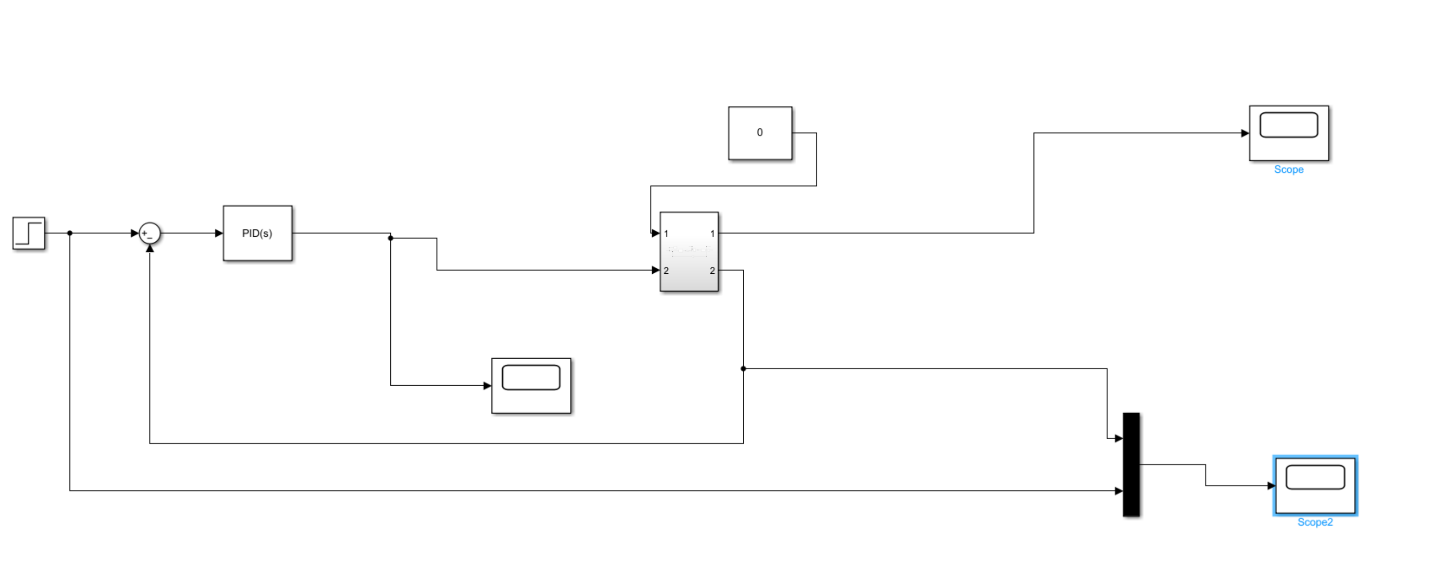
MATLAB Simulation

PID



MATLAB Simulation

PID



6 Conclusion

Conclusion

In this project, we successfully developed a pan and tilt servo control system with face detection, integrating mechanical design, wiring, coding, and serial communication. The system effectively tracks faces in real-time, adjusting the servo motor positions to maintain alignment with the detected subject.

Thanks!

ANY QUESTIONS?