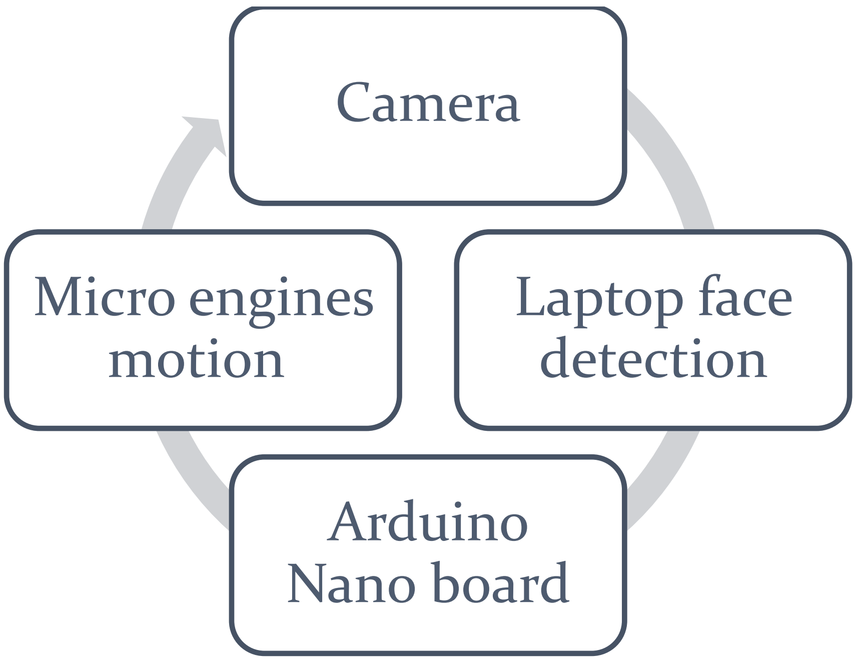
Face Recognition & Tracking Camera – Arduino Project

By Yuval Horowitz & Ariel Turnowski

1. Background on the mission

In this project we were required to build a system that combines the Arduino controller to perform simple operations using engine components together with the OpenCv library of Python which has a high computational capacity.



The system is built in a circular manner, as described in the diagram: the camera takes pictures without a break (about 45 frames per second). She sends the photos to the laptop, the computer has an external library in Python called OpenCv that performs recognition of the face in the photos using a grayscale algorithm, when recognizing the face it calculates the distance of the center of the face from the center of the frame. When the aforementioned distance passes a certain limit, he sends a command to the Arduino to move the servo motors in the opposite direction in the appropriate axis in order to follow the face.

2. Work steps

a) Learning and understanding how to operate the Arduino system using Python (sending commands and control using LEDs).

b) Writing code in Python that will run on the computer and with the help of the camera and the OpenCv library will perform face recognition and calculate the distance of the face from the center of the frame.

c) Writing a function in Arduino idle which receives values classified according to axes and values and moves the corresponding motor with the corresponding value.

d) Building the system itself - connecting all the components to each other and initial wiring.
e) Building a stand and mounting the system on it.

3. Hardware:

a) Web cam

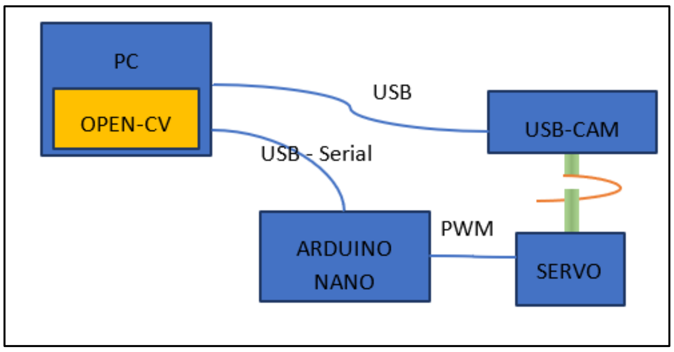
A camera with a USB connection that allows you to receive a video signal at a rate of about 45 frames per second.

b) A pair of FS90R SERVO motors

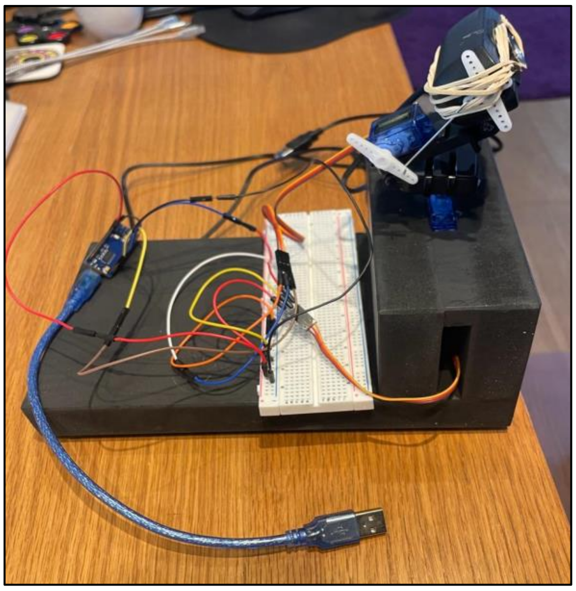
The motor receives a current of about 4.8-6 V, using an electric motor and gears it moves an axis. The motors are connected to the controller via voltage, GND and PWM cables, the purpose of which is to allow pulses to be given in order to move the motor for a short period of time. The two servo motors are used as the only two consumers in the electrical circuit of the system, when they are connected in parallel to a voltage of 5V and each is connected to a digital pin that allows PWM.

c) Arduino Nano controller

An easily programmable controller capable of simple interfacing with a large number of components. The controller is connected via USB to the computer.



מרשם סכמטי של המערכת


תצלום של מבנה המערכת

4. **Software**

a. Libraries used:

- Python: pyserial, OpenCv, numpy.
- Arduino: VarSpeedServo.h.

b. Software used:

- Arduino IDE.
- IDLE (Python).

c. External files:

- Built-in XML file - haarcascade_frontalface_alt, intended for face recognition in the interface with the OpenCv library.

d. Explanation of the algorithm:

First we defined reading and size of video (in pixels) which are read from the camera then we defined facial recognition that realizes the grayscale principle which is ideal for face recognition because by means of a matrix representation of pixels as numbers between 0 and 255 the boundaries of shapes can be distinguished, in the search for boundaries in the form of a face it can be classified the face from the rest of the picture - a matrix.

An information transmission frequency (Baud rate) of 250000 was defined in order to allow the system to react as quickly as possible.

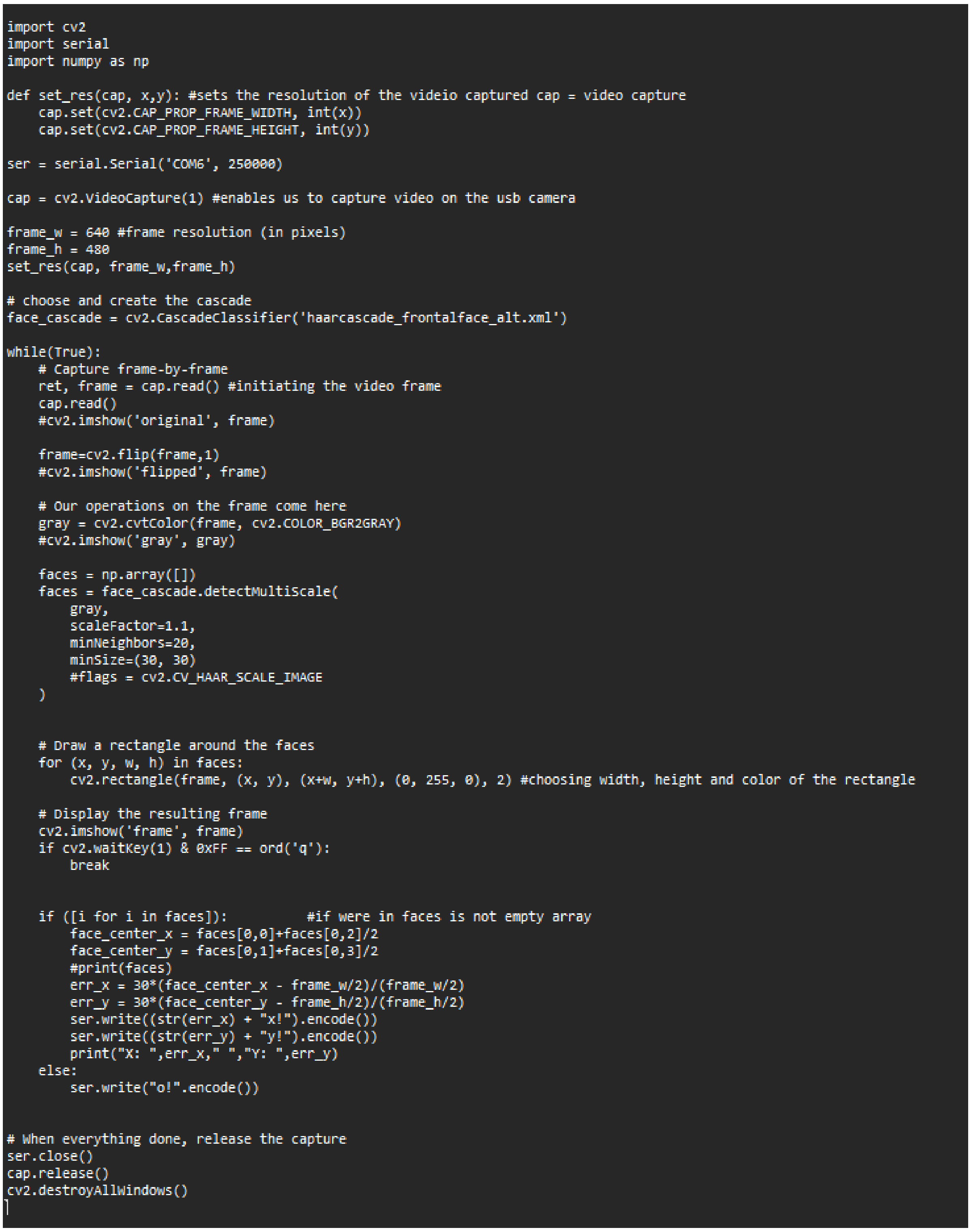
After that, if the algorithm detects a face in a certain frame, it calculates the position of the center of the face and its relative position from the center of the frame in both the X axis and the Y axis and sends them to the Arduino, if it does not detect a face, it sends the Arduino a character that does not move the camera.

5. Summary

The project emphasizes the interfaces between the Arduino, the computer and the camera while using the enormous power of Python and its libraries in order to perform complicated calculations and transfer simple results to the Arduino in order to perform operations that apparently sound relatively complicated. In relation to the previous project, the electrical circuits were very simple and there was no need to add electronic components in order to change the various currents and voltages in the circuit. On the other hand, the software interface between Python and Arduino was a jump in terms of our knowledge of hardware design.

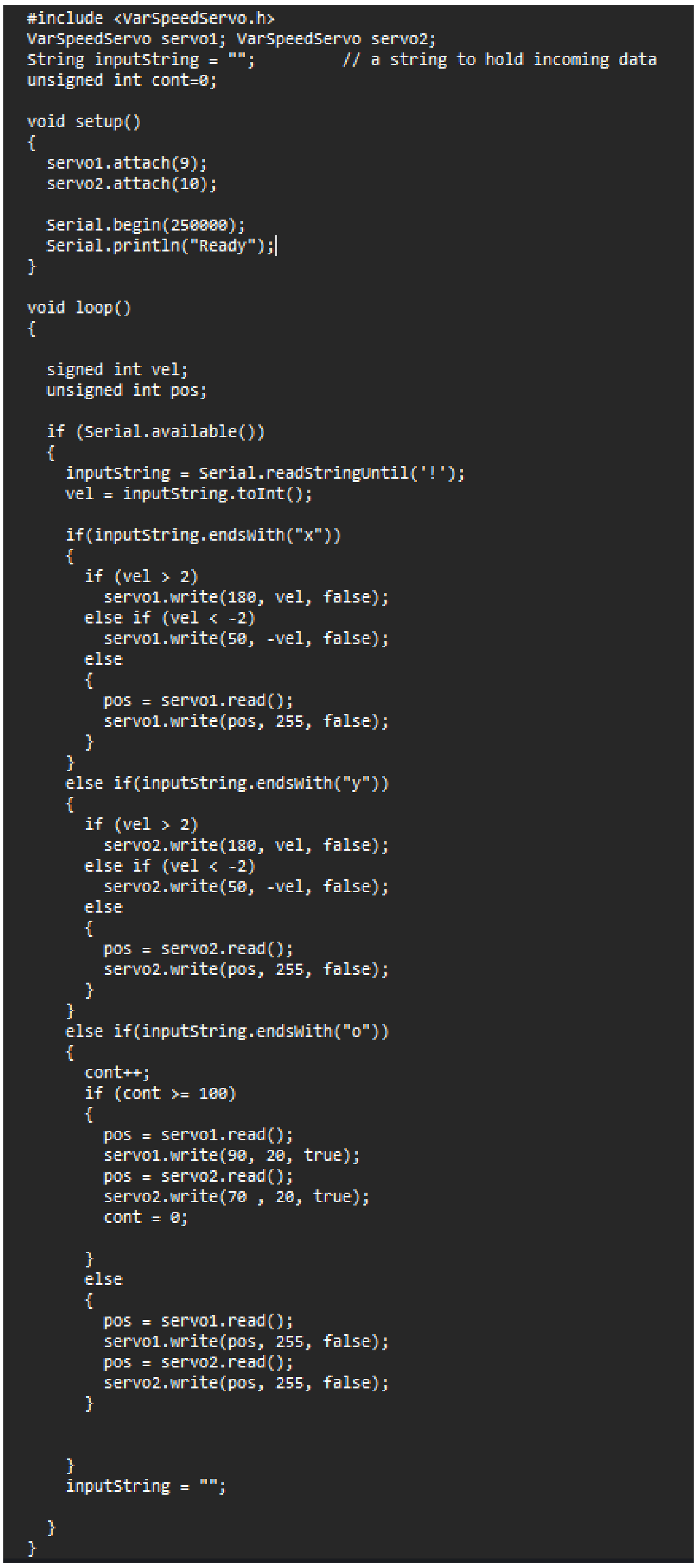
Appendices

1. Python Code



```python
import cv2
import serial
import numpy as np

def set_res(cap, x,y): #sets the resolution of the videio captured cap = video capture
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, int(x))
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, int(y))

ser = serial.Serial('COM6', 250000)

cap = cv2.VideoCapture(1) #enables us to capture video on the usb camera

frame_w = 640 #frame resolution (in pixels)
frame_h = 480
set_res(cap, frame_w,frame_h)

# choose and create the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read() #initiating the video frame
    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)
    #cv2.imshow('gray', gray)

    faces = np.array([])
    faces = face_cascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=20,
        minSize=(30, 30)
        #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) #choosing width, height and color of the rectangle

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


    if ([i for i in faces]):          #if were in faces is not empty array
        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())


# When everything done, release the capture
ser.close()
cap.release()
cv2.destroyAllWindows()
```

2. Arduino IDE Code



```
#include <VarSpeedServo.h>
VarSpeedServo servo1; VarSpeedServo servo2;
String inputString = "";         // a string to hold incoming data
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(50, -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(50, -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();
        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 = "";

  }
}
```