

**BURSA TECHNICAL UNIVERSITY-FACULTY OF ENGINEERING AND NATURAL
SCIENCES-INTERNATIONAL**

**REAL TIME FACE TRACKING SYSTEM WITH IMAGE PROCESSING &
RASPBERRY PI & SERVO MOTOR**

ENGINEERING DESIGN

Onur DURMUŞ

Ö160105005

Supervisor Teacher : Doç. Dr. Cemal HANILÇI

DECEMBER 2019

CONTENTS

FIGURE LIST.....	iv
1. PURPOSE OF THE PROJECT.....	v
2. INSTALLING OPENCV.....	1-4
3. RASPBERRY PI ARDUINO SERIAL COMMUNICATION.....	5-6
4. FACE DETECTION.....	7
5. FUNDAMENTAL OF THE PROJECT'S CODES.....	8-9
6. PICTURE OF THE PROJECT.....	10
7.WORKING IMAGE OF THE PROJECT.....	11
8. REFERENCES.....	12

PURPOSE OF THE PROJECT

Today, tracking objects and people is very important both in terms of military space and in terms of population control in populated countries. For this reason, real-time object and human tracking after image processing was introduced in the 2000s, the use of this technology will benefit countries and will continue to evolve much as it is a never technology.

FIGURE LIST

		<u>Page No</u>
Figure 1.0	Configuring Raspberry-Pi file system	1
Figure 1.1	Configuring Raspberry-Pi file system	1
Figure 2.0	Arduino-RaspberryPi serial connection	5
Figure 2.1	Activated I2C and Serial features of RaspberryPi	5
Figure 2.2	Activated I2C and Serial features of RaspberryPi	5
Figure 2.3	Installing and identify library for Arduino in Rasp terminal	6
Figure 3.0	Example of Haarcascade Classifier	7
Figure 4.0	Examples of Python codes	8
Figure 4.1	Examples of Python codes	8
Figure 4.2	Examples of Python codes	8
Figure 4.3	Examples of Python codes	9
Figure 4.4	Examples of Python codes	9
Figure 4.5	Examples of Python codes	9
Figure 5.0	Picture of Project	10
Figure 5.1	Working Image of the Project	11

INSTALLING OPENCV ON RASPBERRY PI

After installing Raspbian (type of Linux) operating system on Raspberry Pi, we enter the terminal and perform the following steps.

Step #1: Expand filesystem on Raspberry Pi

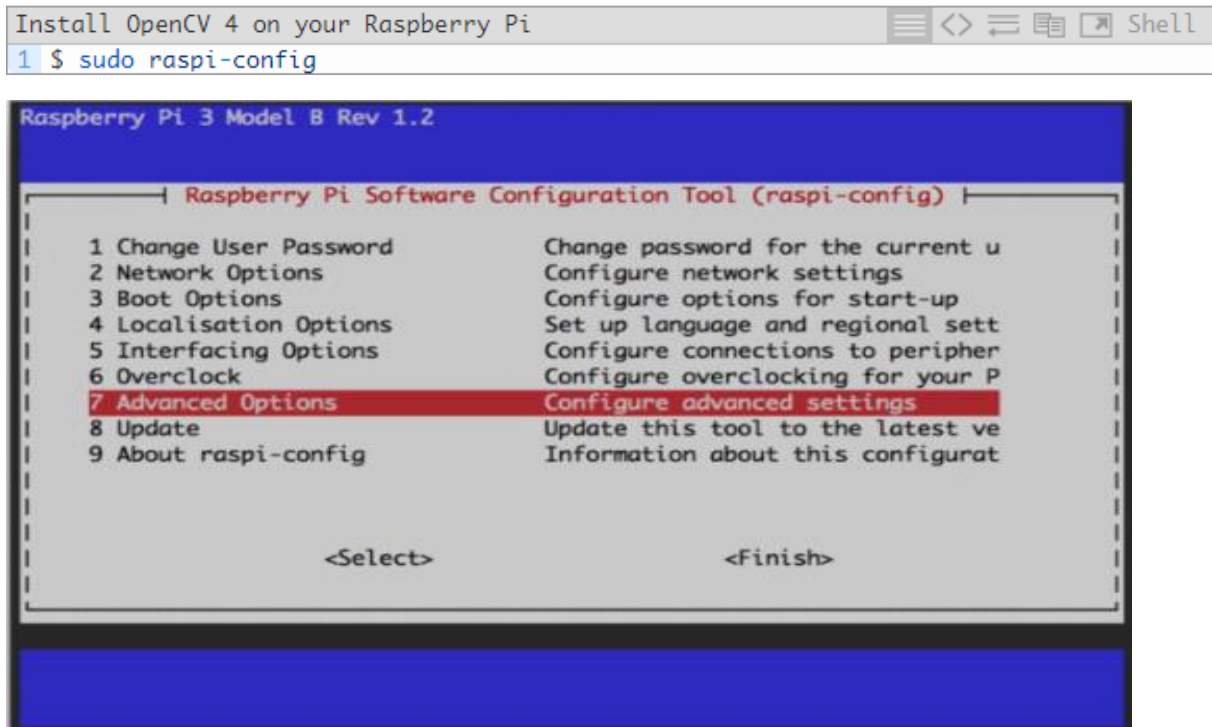


Figure 1.0

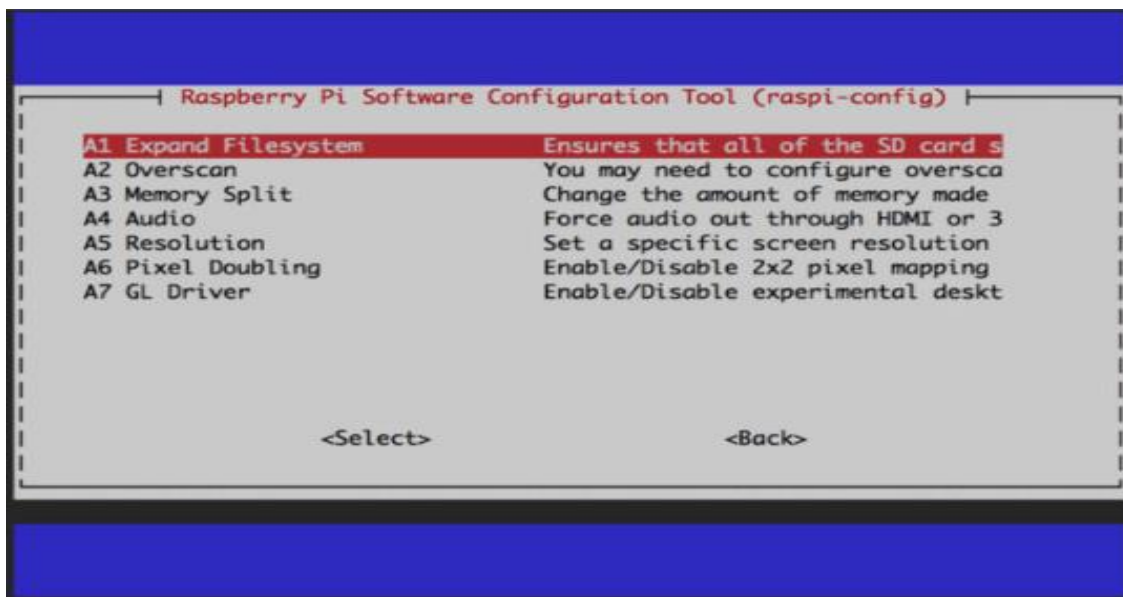


Figure 1.1

Step #2: Installing OpenCV 4 dependencies on Raspberry Pi

-Updating our system:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get update && sudo apt-get upgrade
```

-Installing developer tools including CMAKE:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install build-essential cmake unzip pkg-config
```

-Installing a necessary of image and video libraries, these are critical to being able to work with image and video files:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
2 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
3 $ sudo apt-get install libxvidcore-dev libx264-dev
```

-Installing GTK, our GUI backend:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install libgtk-3-dev
```

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install libcanberra-gtk*
```

-Followed by installing two packages which contain numerical optimizations for OpenCV:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install libatlas-base-dev gfortran
```

-Installing the Python3 development headers:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo apt-get install python3-dev
```

Step #3: Downloading OpenCV 4 for Raspberry Pi

-For OpenCV to work stably, we need to install it together with “contrib” modules.

```
Install OpenCV 4 on your Raspberry Pi
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip
3 $ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.0.0.zip
```

And then, unzip the downloaded files:

```
Install OpenCV 4 on your Raspberry Pi
1 $ unzip opencv.zip
2 $ unzip opencv_contrib.zip
```

Step #4: Configuring Python 3 virtual environment for OpenCV4

-Installing pip that a Python Package Manager:

```
Install OpenCV 4 on your Raspberry Pi
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python3 get-pip.py
```

-Making use of virtual environments for Python development: Virtual environments will allow our to run different versions of Python software in isolation on our system.

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo pip install virtualenv virtualenvwrapper
2 $ sudo rm -rf ~/get-pip.py ~/.cache/pip
```

```
Install OpenCV 4 on your Raspberry Pi
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 source /usr/local/bin/virtualenvwrapper.sh
```

```
Install OpenCV 4 on your Raspberry Pi
1 $ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
2 $ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
3 $ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
4 $ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

```
Install OpenCV 4 on your Raspberry Pi
1 $ source ~/.profile
```

Now it is ready to creating a virtual environment to hold OpenCV 4 and additional packages:

```
Install OpenCV 4 on your Raspberry Pi
1 $ mkvirtualenv cv -p python3
```

```
Install OpenCV 4 on your Raspberry Pi
1 $ workon cv
```

-Installing NumPy : The first Python package and only OpenCV prerequisite we'll install is NumPy. Numpy is array packages and this is too important modules for OpenCV.

```
Install OpenCV 4 on your Raspberry Pi
1 $ pip install numpy
```

Step #5: CMake and compiling OpenCV 4 for Raspberry Pi

```
Install OpenCV 4 on your Raspberry Pi
1 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
2   -D CMAKE_INSTALL_PREFIX=/usr/local \
3   -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
4   -D ENABLE_NEON=ON \
5   -D ENABLE_VFPV3=ON \
6   -D BUILD_TESTS=OFF \
7   -D OPENCV_ENABLE_NONFREE=ON \
8   -D INSTALL_PYTHON_EXAMPLES=OFF \
9   -D BUILD_EXAMPLES=OFF ..
```

And finally we can use “make” command for compiling OpenCV 4, this step takes about 6 hours for Raspberry Pi 3 Model B+.

```
Install OpenCV 4 on your Raspberry Pi
1 $ make -j4
```

And then installing OpenCV 4:

```
Install OpenCV 4 on your Raspberry Pi
1 $ sudo make install
2 $ sudo ldconfig
```

Step #6: Test your OpenCV 4 install on your Raspberry Pi

```
Install OpenCV 4 on your Raspberry Pi
1 $ workon cv
2 $ python
3 >>> import cv2
4 >>> cv2.__version__
5 '4.0.0'
6 >>> exit()
```

Installing OpenCV 4 finished.

Raspberry Pi – Arduino Serial Communication

Firstly, we use USB connected for connection:

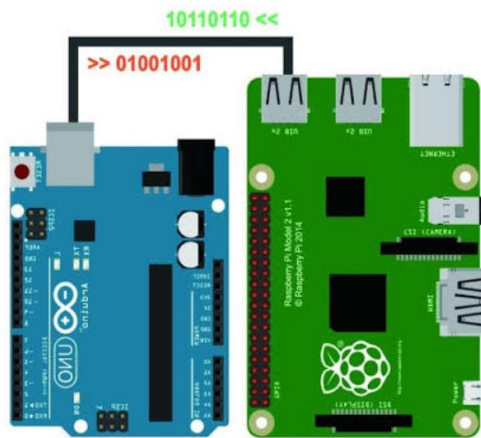


Figure 2.0

And then, we do some configure on Raspberry Pi:



Figure 2.1

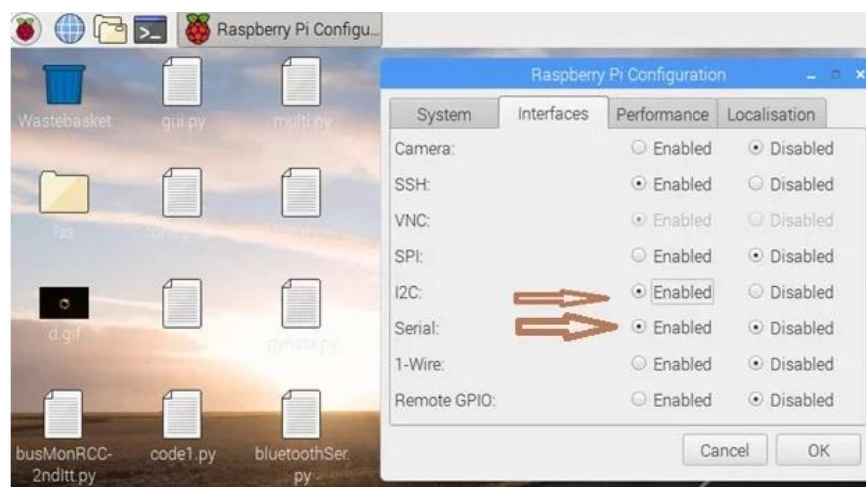


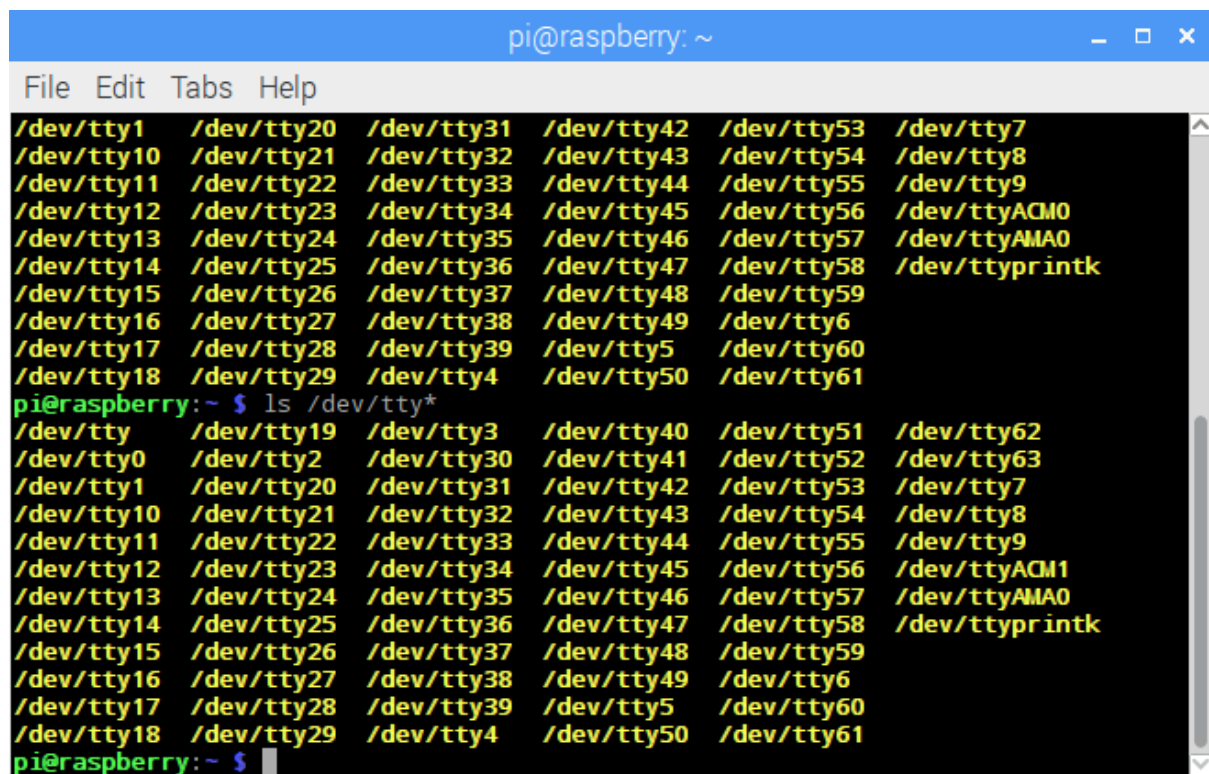
Figure 2.2

And then, we get in terminal and writing these codes:

```
sudo apt-get install python-serial
```

```
sudo pip install pyserial
```

```
ls /dev/tty*
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
/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/ttyACM0  
/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  
pi@raspberrypi:~$ 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  
pi@raspberrypi:~$
```

Figure 2.3

Arduino serial communication with Raspberry is now ready.

The main reason that we use Arduino in this project is that Arduino is more advantageous about pwm, than Raspberry Pi.

FACE&EYES DETECTION

For my Project, I used 'Haarcascade' library. Reason of i used that library, It is so much stably and easy way to recognizing face.

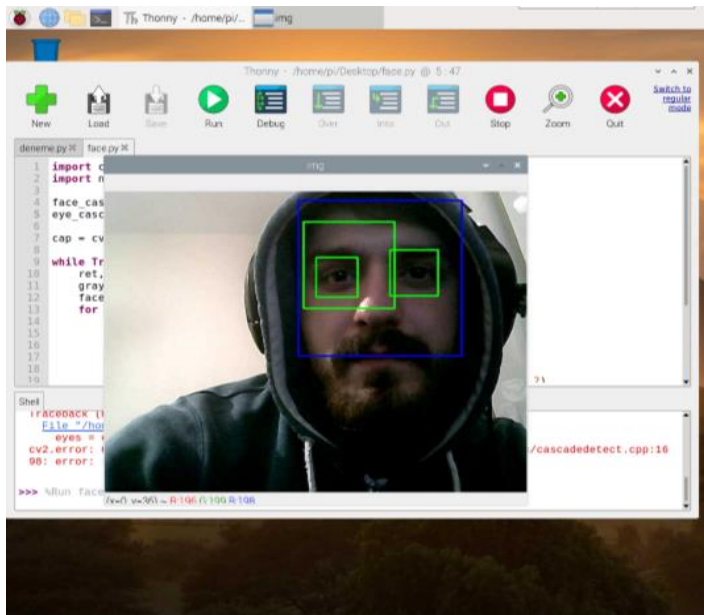


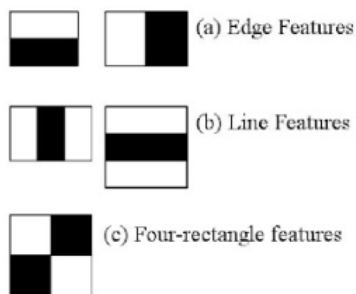
Figure. 3.0

HAARCASCADE WORKING LOGIC:

Edge Features: Indicates that a specific area on the image consists of a dark area and an edge property if a specific area consists of light colors.

Line Features: If the image consists of on, off, light colors, respectively, there is a line property.

Four-rectangle Features: If the dark and light tones are diagonally in the form of a square, it indicates the four-rectangle property.



Using these features we can detect many objects on the image such as line, edge, Face, Eye, tool etc. The haar-cascade method has been trained many times in advance to know The Shape of the face. For example, a system looking for a face first looks for two eyes. If there's an eye, he'll see if there's a nose. If the nose is there, it is the structure that gives the desired results by checking to see if there is a brow

FUNDAMENTAL LOGIC OF CODES

After the above OpenCV and serial installations, we will see the basic logic of code construction step by step. I am using 'Thonny IDE Python' program in Raspberry Pi.

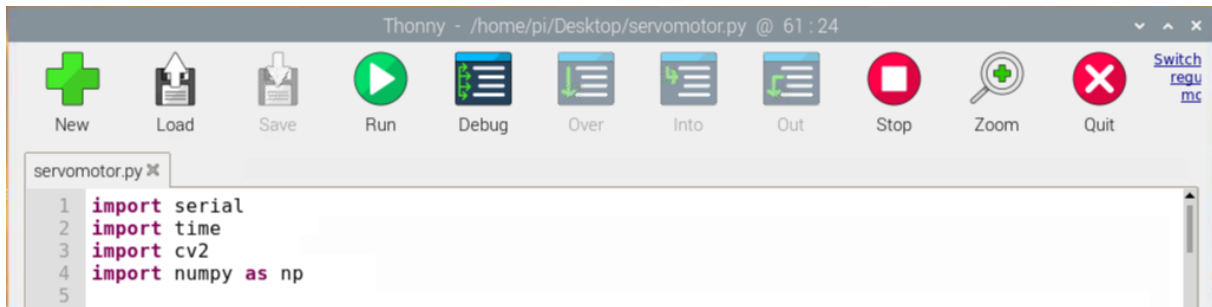


Figure 4.0

First of all, we import the 'serial' library for Arduino Raspberry Pi communications. Then, we import 'cv2' library for image processing. And we import 'numpy' library that is so important library.



Figure 4.1

Then we use 'Haarcascade' classifier in line6. Then we activate USB Camera in line 7. Also, we identify Arduino with line 8. Then we some set camera frame.

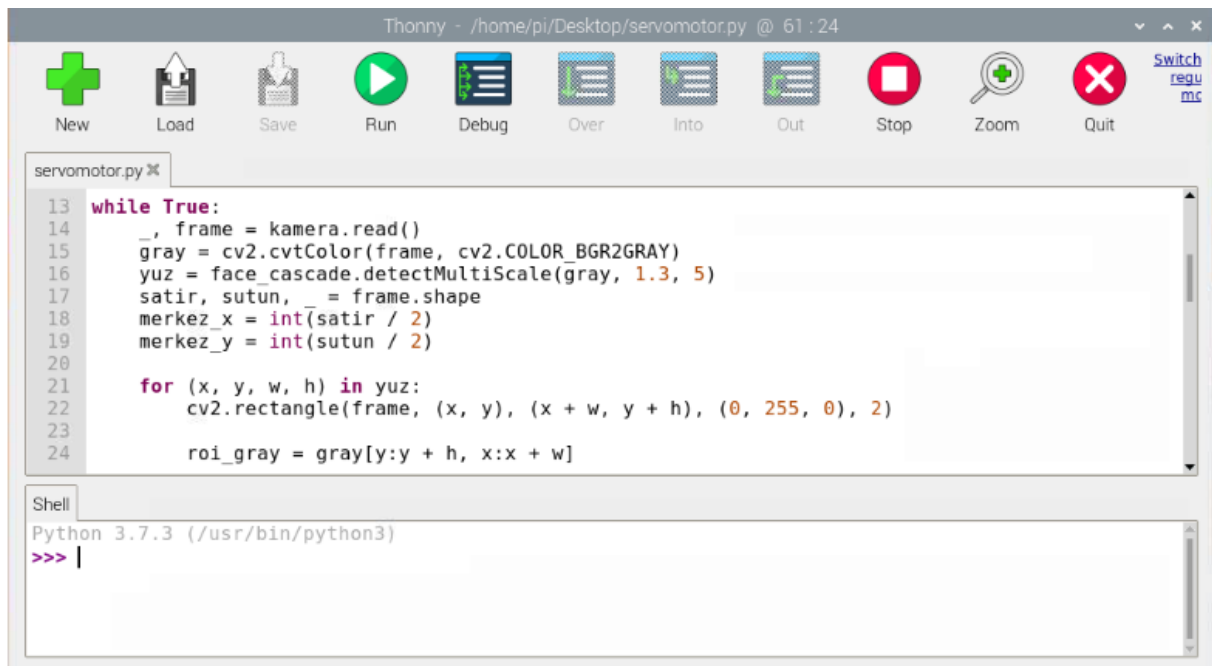


Figure 4.2

In Figure 4.2, we create while True cycle for faces and identify center of rows & columns of camera frame. Then, we now create a 'for' loop to enclose the recognised face into a rectangle.

```

30  #####
31  orta_x = int((x + x+w)/2)
32  orta_y = int((y + y+h)/2)
33  #####

```

Figure 4.3

In this line, we create 'X' and 'Y' lines passing through the center point of the face enclosed in a rectangle.

```

43  if orta_x > merkez_x +25:
44      baslangic_pozisyonu -= 2
45      ser.write(str(baslangic_pozisyonu) + 'a').encode('utf-8'))
46      # time.sleep(0.03)
47  if orta_x < merkez_x -25:
48      baslangic_pozisyonu += 2
49      ser.write(str(baslangic_pozisyonu) + 'a').encode('utf-8'))
50      # time.sleep(0.03)
51
52  break
53
54  cv2.imshow("frame",frame)
55

```

Shell

Python 3.7.3 (/usr/bin/python3)

>>>

Figure 4.4

In Figure 4.4, we create 'if' loop. Then it means, if the center point of the face enclosed in a rectangle shifts to the right from the center point of the camera frame, the servo motor moves to the left, and the servo motor moves to the right if it shifts to the left.

ARDUINO CODE:

```

ServoArdu | Arduino 2:1.0.5+dfsg2-4.1
File Edit Sketch Tools Help
[Icons]
ServoArdu
#include<Servo.h>
Servo x ;
char x;
long int v;
void setup() {
x.attach(4);
Serial.begin(9600);
x.write(90);
}

void loop() {
if(Serial.available()>2){
x =Serial.read();
v =Serial.parseInt();
if(x == 'a'){
x.write(v);
}
delay(10);
}
}

```

Figure 4.5

PICTURE OF PROJECT



Figure 5.0

WORKING IMAGE OF THE PROJECT

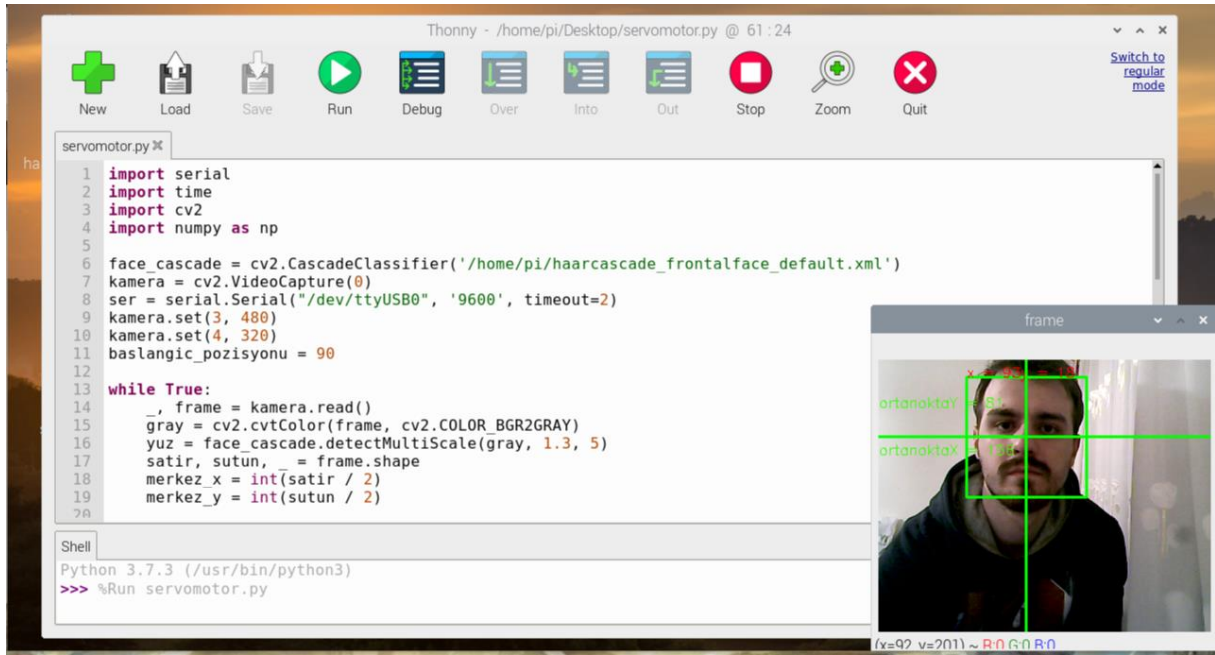


Figure 5.1

REFERENCES

- [1] Hands-On Image Processing with Python:Expert Techniques for Advanced, Sandipan Dey
- [2] <<https://github.com/opencv/opencv/tree/master/data/haarcascades>>
- [3] <<https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/>>
- [4] <<https://pysource.com/>>
- [5] <<https://stackoverflow.com/>>
- [6] < <https://www.researchgate.net/>>