Object Oriented Programming Term Project

İzzet Emre Demir

Project Environment:

- Python 3.8+
- Linux Mint 20 Cinnamon
- Microcase Mikrofonlu Hd Webcam Kamera 720P 30 FPS

Description of the project:

Develop a tennis ball detection software with OOP concepts

Project Structure:

green_ball_tracker.py : Tennis ball tracker classes here

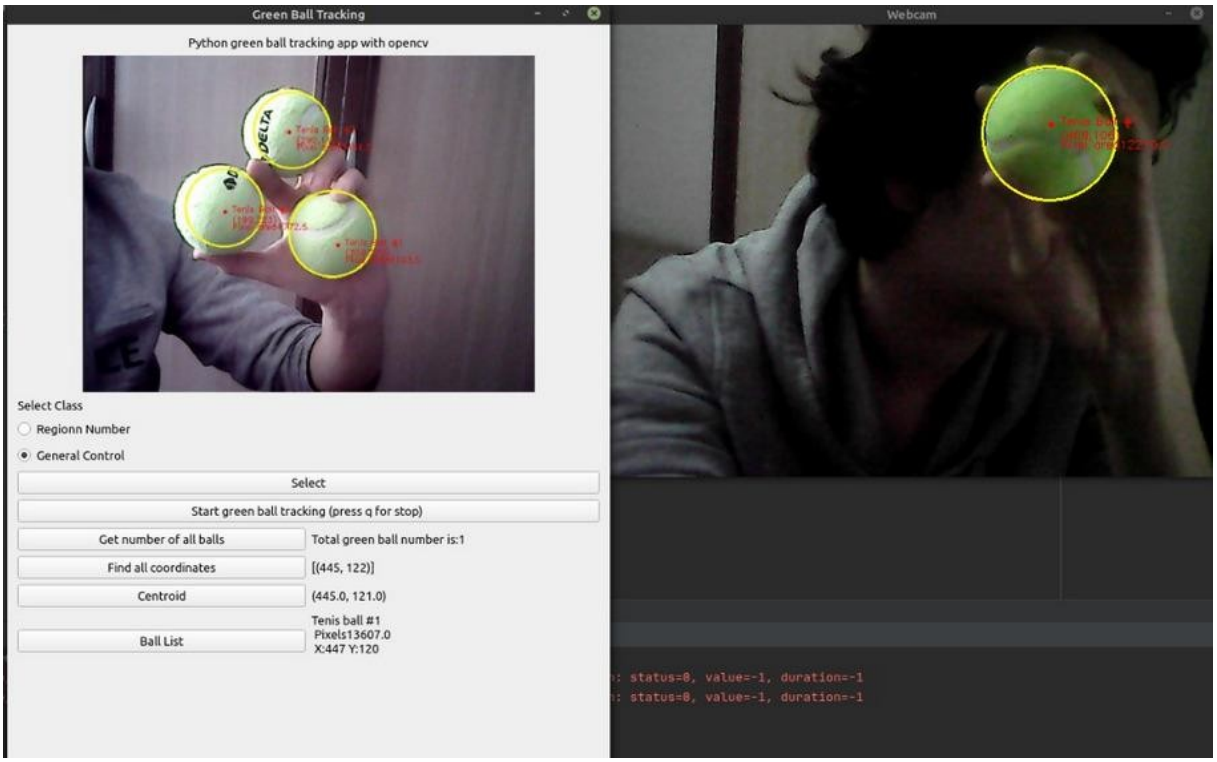
- Tennis_ball_detect Class
- Region_number Class
- General_control Class

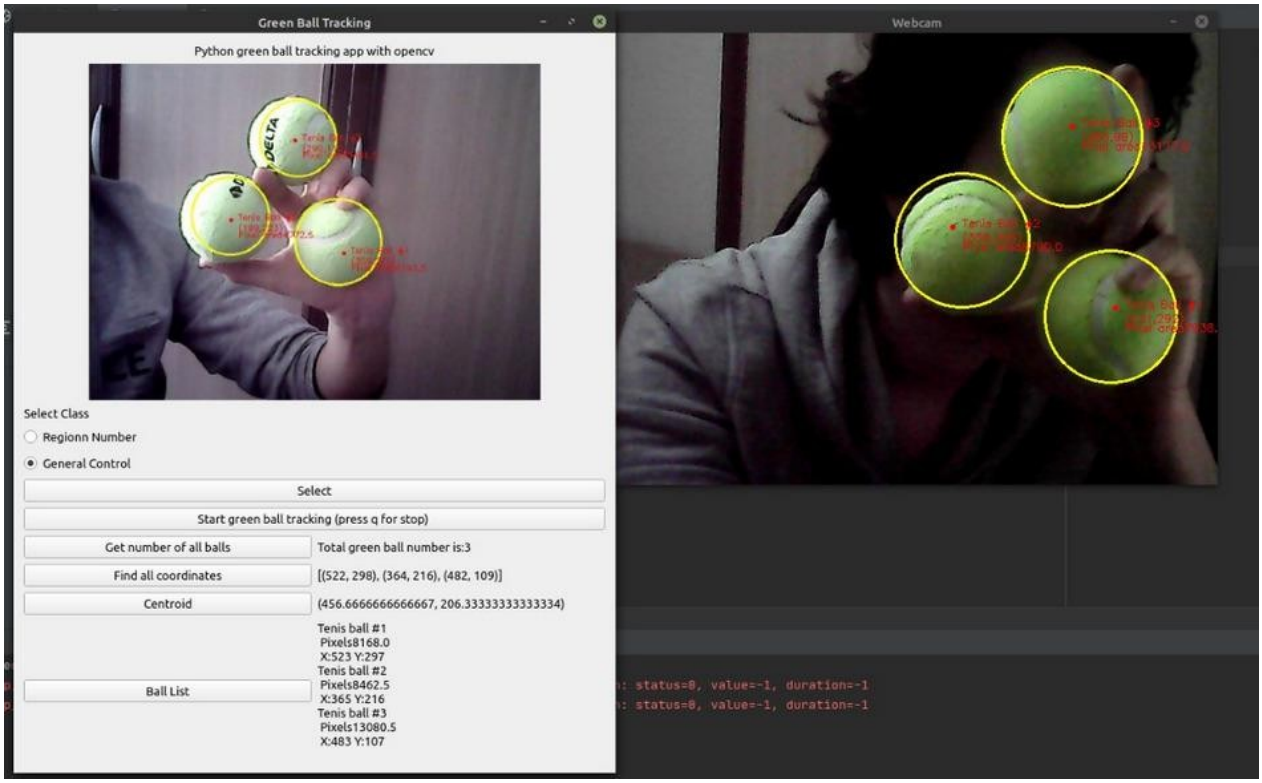
main.py : gui interface part here

- Window Method

Interface:

In main.py file I defined a Window function and use PyQt5 library for gui.





Installation:

sudo apt-get update
```
pip install -r requirements.txt
python3 main.py
```

Task Requirements:

A. Define an abstract super class named tennis_ball_detect. In this class the number of the balls and their coordinates should be defined as abstract methods.

```
@abstractmethod
def get_balls_number(self):
    pass
@abstractmethod
def get_ball_coordinates(self):
```

B. Define a region_number subclass which inherits from tennis_ball_detect super class and calculates the number of regions. In this class, referring to the previously defined abstract methods, calculate the number of the balls and their coordinates. Use encapsulation in order to prevent any change in the identified ball numbers and their coordinates. Also in this class define a method named centroid that calculates the centroid coordinates of each region.

```
class Region_number(Tennis_ball_detect):

    __regions = []
    def __init__(self):
        __regions = self.__regions
        super().__init__()

    # Function for gettion number of all tenis balls
    def get_balls_number(self):
        return len(self.get_ball_list())

    # Return all tenis ball coordinates
    def get_ball_coordinates(self):
        coordinate_list = []
        for i in self.get_ball_list():
            coordinate_list.append((i.x,i.y))
        if(len(coordinate_list) != 0):
            return coordinate_list
        else:
            return None

    def get_regions(self):
        return self.__regions

    def set_regions(self,r_list):
        self.__regions = r_list

    #The function that calculates the center coordinates of 5 different regions
    def centroid(self):
        h  = self.windowHeight
        w =self.windowWidth
        r_list =[]
        for i in range(5):
            r_list.append(((w/10)+(+w/5)*i,h/2))
```

```
        self.set_regions(r_list)
        return (r_list)
```

C. Define a general_control subclass which inherits from tennis_ball_detect super class. In this class define a method named centroid that calculates the overall centroid coordinates of the green pixels on the camera vision (Polymorphism).

```
class General_control(Tennis_ball_detect):
    __regions = []

    def __init__(self):
        __regions = self.__regions
        super().__init__()

    # Function for gettion number of all tenis balls
    def get_balls_number(self):
        return len(self.get_ball_list())

    # Return all tenis ball coordinates
    def get_ball_coordinates(self):
        coordinate_list = []
        for i in self.get_ball_list():
            coordinate_list.append((i.x, i.y))
        if (len(coordinate_list) != 0):
            return coordinate_list
        else:
            return None

    # The function that calculates the center coordinates of overall green balls
    def centroid(self):
        x = 0
        y = 0
        for i in self.get_ball_list():
            x +=i.x
            y += i.y
        return x/len(self.get_ball_list()),y/len(self.get_ball_list())
```

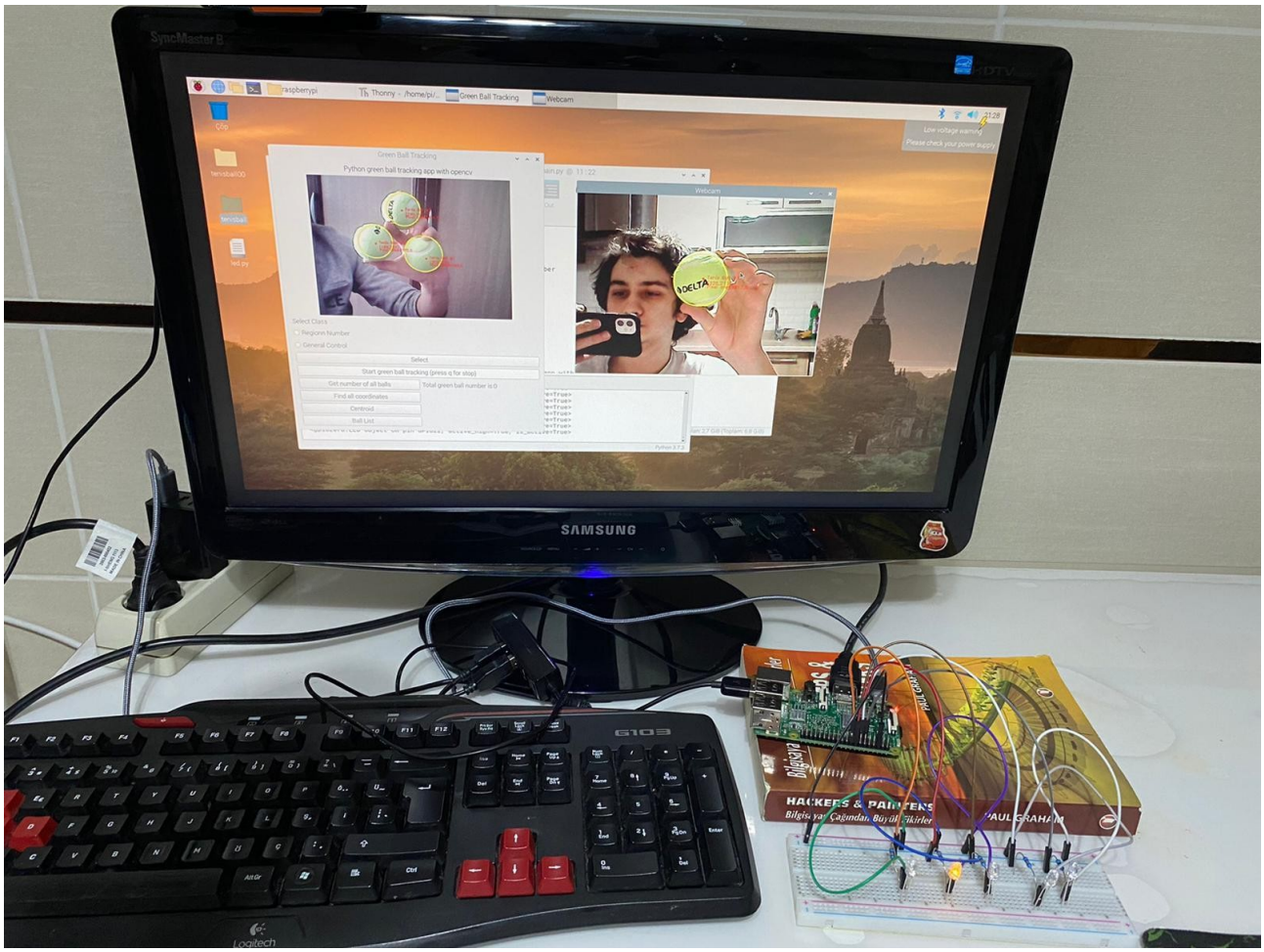
Raspberry Pi Implementation:

There is a raspberry pi folder in project folder.
There are 5 region represented by a led. In raspberry pi implementation turns on led which represents region of the tennis ball.

regions.

1	2	3	4	5

Some additional codes were required for this.
In green_ball_tracker.py we forked Tennis_ball_detect class and changed track_balls function.
With this code we can control leds via gpio pins.

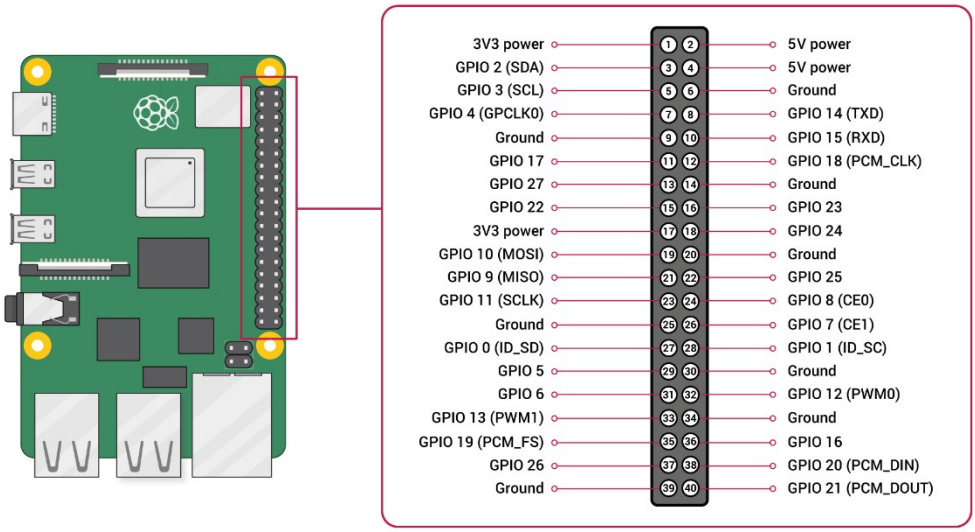


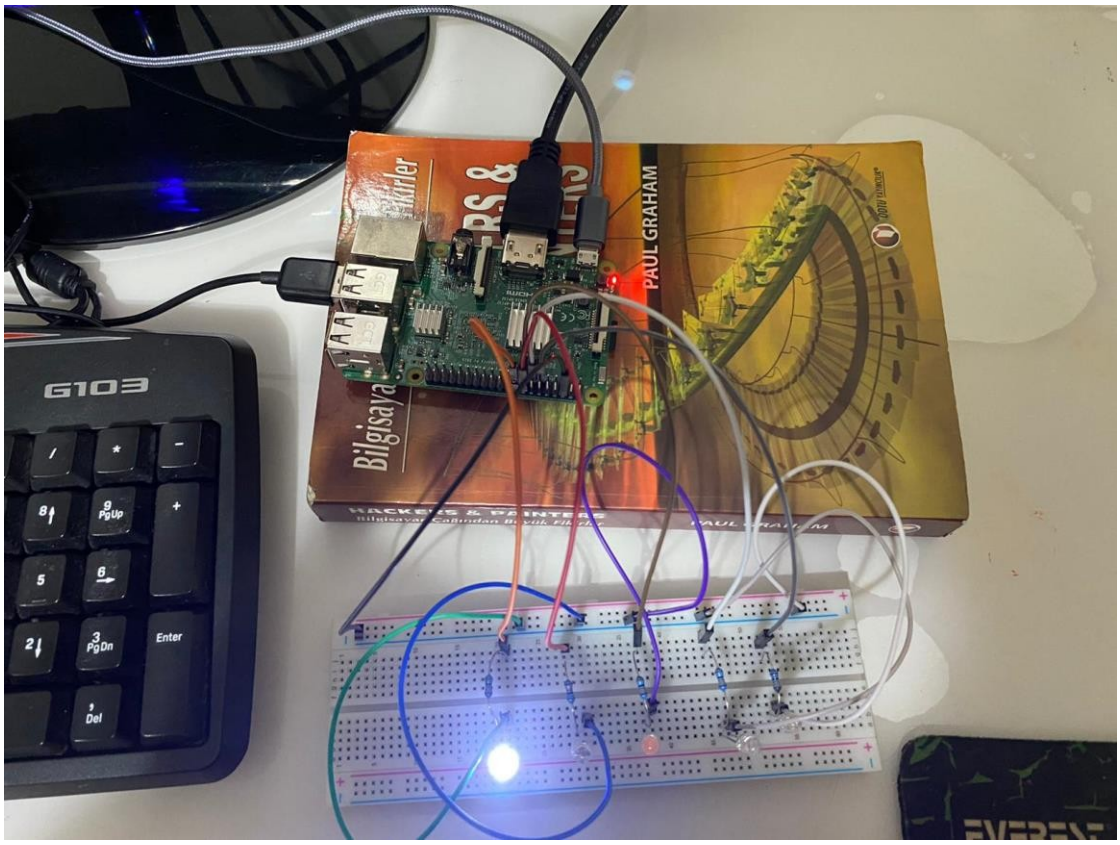
First import gpio library in init function and define variables:

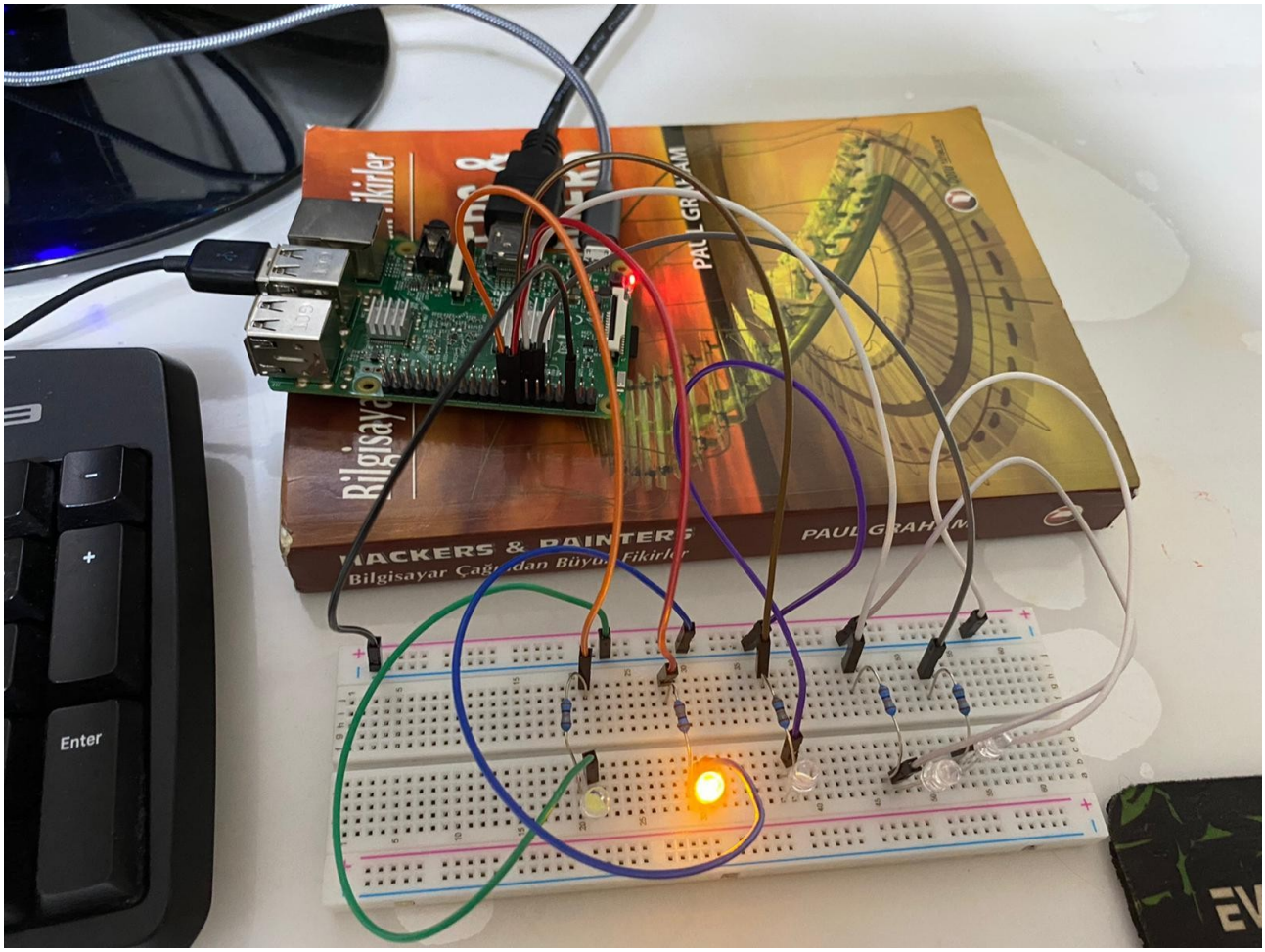
```
from gpiozero import LED
self.led1 = LED(17)
self.led2 = LED(27)
self.led3 = LED(22)
self.led4 = LED(23)
self.led5 = LED(24)

### Raspberry Pi Part
# Get Reagion Number
reagion = 0
try:
    x = 0
    y = 0
    for i in self.get_ball_list():
        x += i.x
        y += i.y
    center = x / len(self.get_ball_list()), y / len(self.get_ball_list())

    h = self.windowHeight
    w = self.windowWidth
    for i in range(5):
        if (center[0] > (w / 5) * i) and center[0] < ((w / 5) * (i + 1)):
            reagion = i + 1
except:
    pass

try:
    active_led = None
    active_reagion = None

    def change_led(led):
        global active_reagion
        if (active_led):
            active_led.off()
        led.on()
        active_reagion = reagion

    if (active_reagion != reagion):
        if (reagion == 1):
            change_led(self.led1)
        elif (reagion == 2):
            change_led(self.led2)
        elif (reagion == 3):
            change_led(self.led3)
        elif (reagion == 4):
            change_led(self.led4)
        elif (reagion == 5):
            change_led(self.led5)
except:
    pass
```

GPIO Diagram:

Led 1 Connects GPIO 17
Led 2 Connects GPIO 27
Led 3 Connects GPIO 22
Led 4 Connects GPIO 23
Led 5 Connects GPIO 24







Conclusion:

I developed tennis ball tracking project with using OOP concepts I use opencv for image processing and pyqt5 for interface. After that I implemented it for Raspberry pi and use GPIO pins. Deployment on Raspberry pi can be harder than pc because of additional libraries of project requirements.