DIGITAL IMAGE PROCESSING LAB
FINAL LAB PAPER REPORT
BSCE-VIII

By

M Abul Hassan (202101004)

Submitted to: **Sir Zia ur Rehman**

Date: June-11-2024

DEPARTMENT OF COMPUTER ENGINEERING
INSTITUTE OF SPACE TECHNOLOGY
KICSIT, KAHUTA CAMPUS

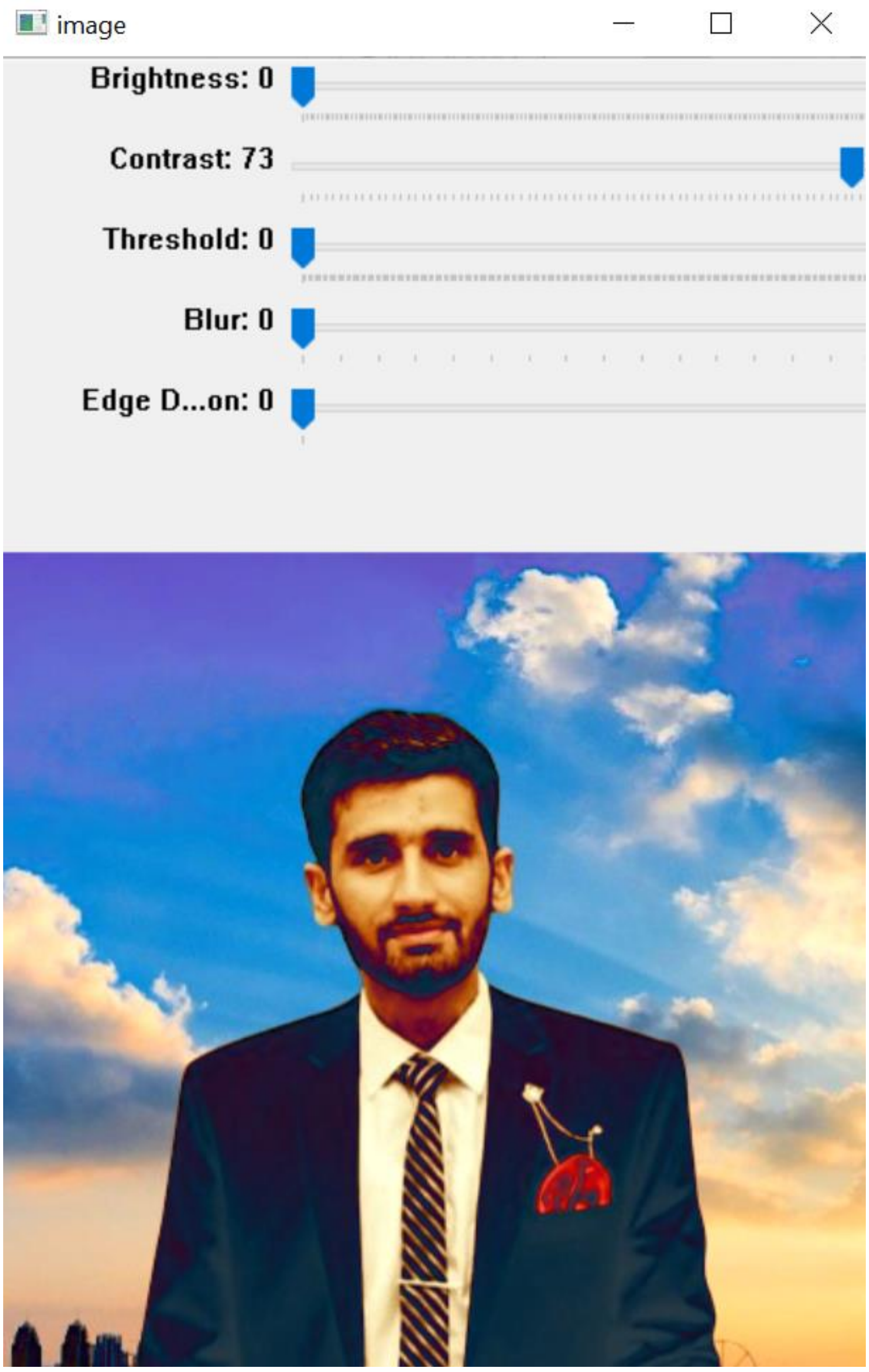
Q1:Produce the python program to explore the OpenCV tracker function with an example?

Code:

```
import cv2
import numpy as np

# Callback function for trackbar
def update_image(x):
    # Get the current positions of the trackbars
    brightness = cv2.getTrackbarPos('Brightness', 'image') - 100
    contrast = cv2.getTrackbarPos('Contrast', 'image') / 50.0
    threshold = cv2.getTrackbarPos('Threshold', 'image')
    blur = cv2.getTrackbarPos('Blur', 'image')
    edge = cv2.getTrackbarPos('Edge Detection', 'image')

    # Adjust brightness and contrast
    adjusted = cv2.convertScaleAbs(img, alpha=contrast, beta=brightness)

    # Apply Gaussian Blur if blur trackbar is not at 0
    if blur > 0:
        ksize = (2 * blur + 1, 2 * blur + 1)  # kernel size must be odd
        adjusted = cv2.GaussianBlur(adjusted, ksize, 0)

    # Apply threshold if threshold trackbar is not at 0
    if threshold > 0:
        _, adjusted = cv2.threshold(adjusted, threshold, 255,
cv2.THRESH_BINARY)

    # Apply edge detection if edge detection trackbar is not at 0
    if edge > 0:
        adjusted = cv2.Canny(adjusted, 50, 150)

    # Display the updated image
    cv2.imshow('image', adjusted)

# Load an image
img = cv2.imread('image.jpg')  # Replace with your image path
if img is None:
    print("Error loading image")
    exit()
```

```
# Create a window
cv2.namedWindow('image')

# Create trackbars for adjusting brightness, contrast, threshold, blur,
and edge detection
cv2.createTrackbar('Brightness', 'image', 100, 200, update_image)
cv2.createTrackbar('Contrast', 'image', 50, 100, update_image)
cv2.createTrackbar('Threshold', 'image', 0, 255, update_image)
cv2.createTrackbar('Blur', 'image', 0, 20, update_image)
cv2.createTrackbar('Edge Detection', 'image', 0, 1, update_image)

# Display the original image
cv2.imshow('image', img)

# Wait until user presses the ESC key
while True:
    if cv2.waitKey(1) & 0xFF == 27:  # 27 is the ASCII code for the ESC
key
        break

# Destroy all windows
cv2.destroyAllWindows()
```

OUTPUT:



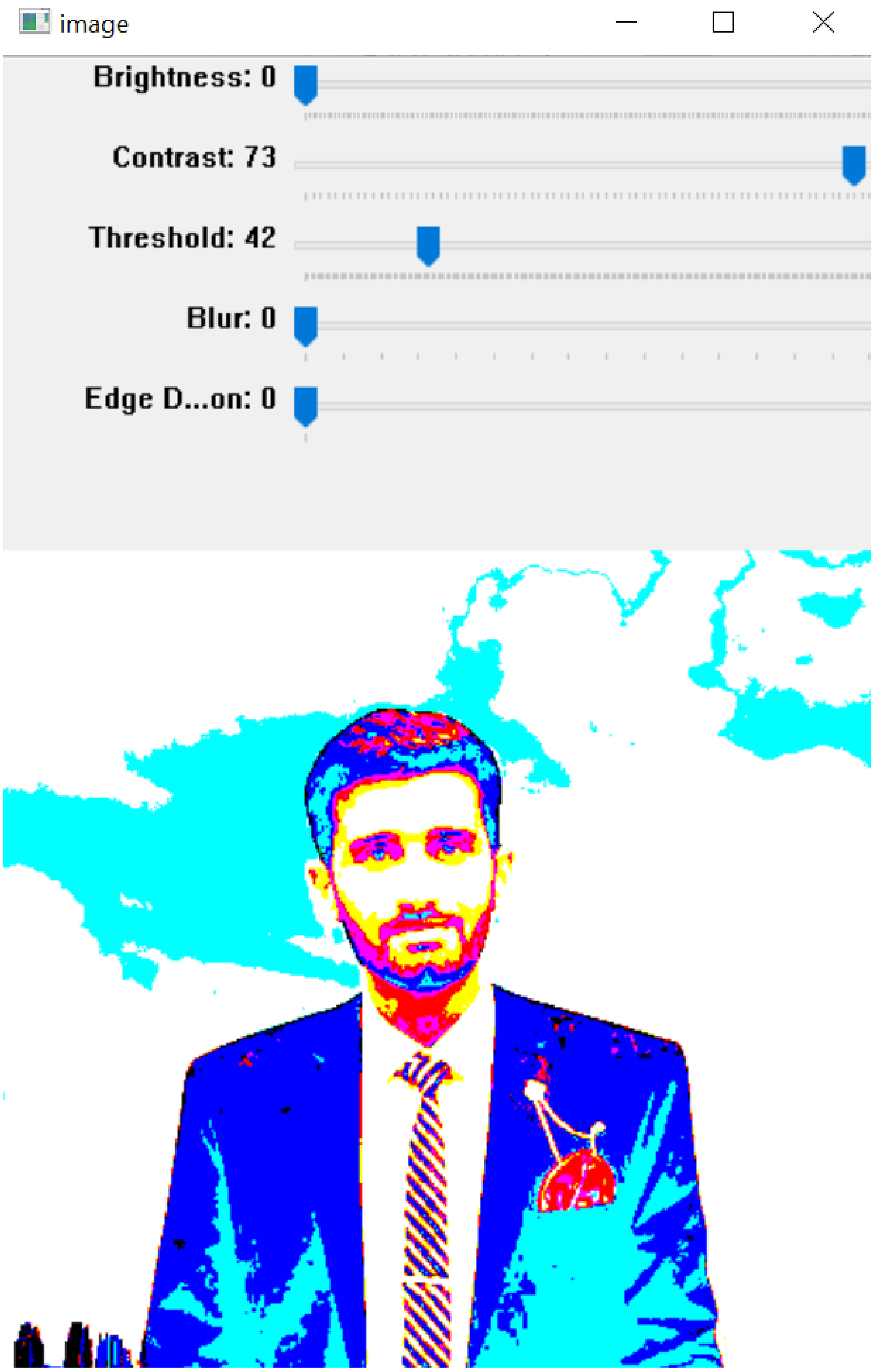
Brightness: 0

Contrast: 73

Threshold: 0

Blur: 0

Edge D...on: 0



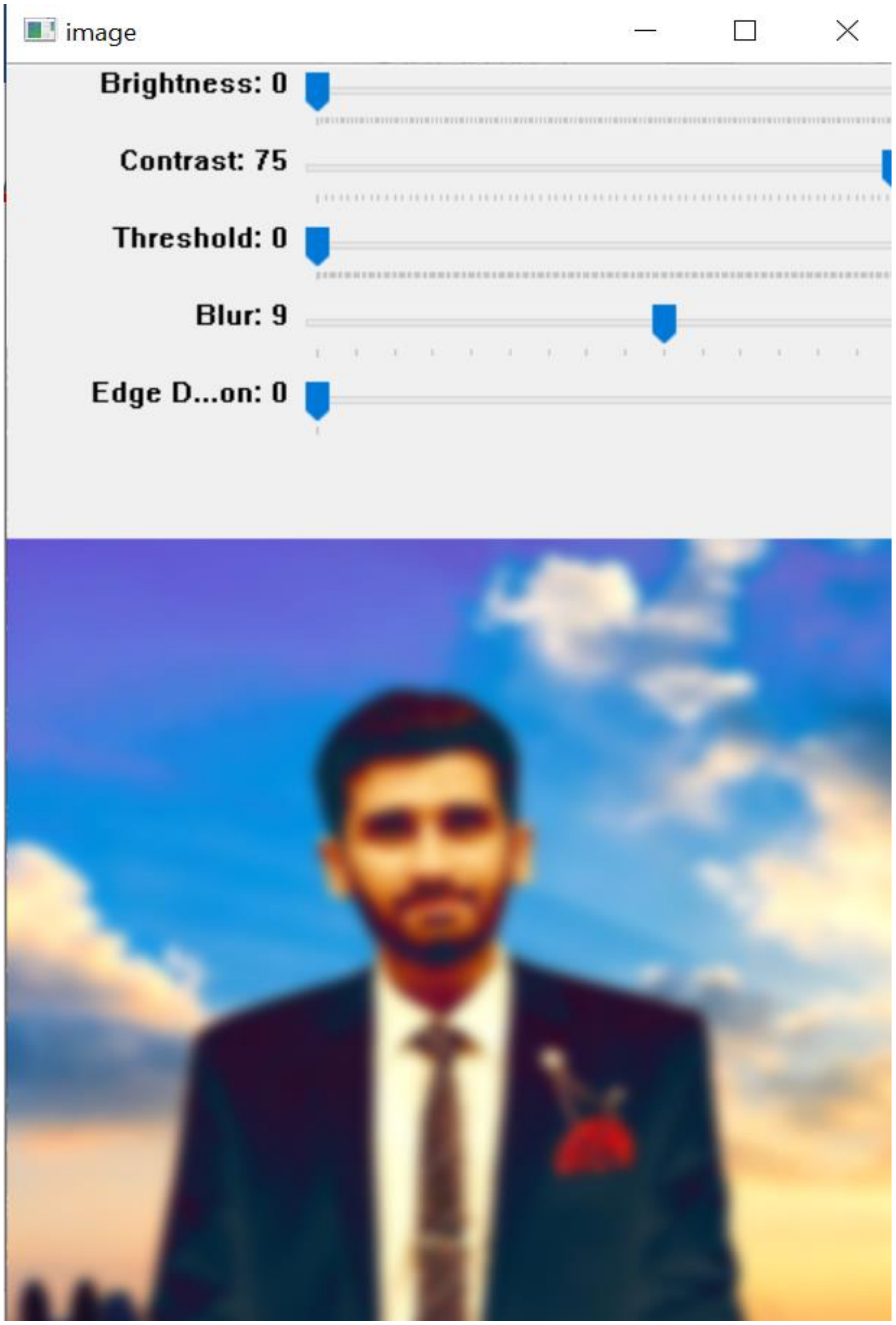
Brightness: 0

Contrast: 73

Threshold: 42

Blur: 0

Edge D...on: 0



image

Brightness: 0

Contrast: 75

Threshold: 0

Blur: 9

Edge D...on: 0

Q2:P roduce black image using python lib and using trackers explore how you can produce R,G,B color on the basis of value from trackbar?

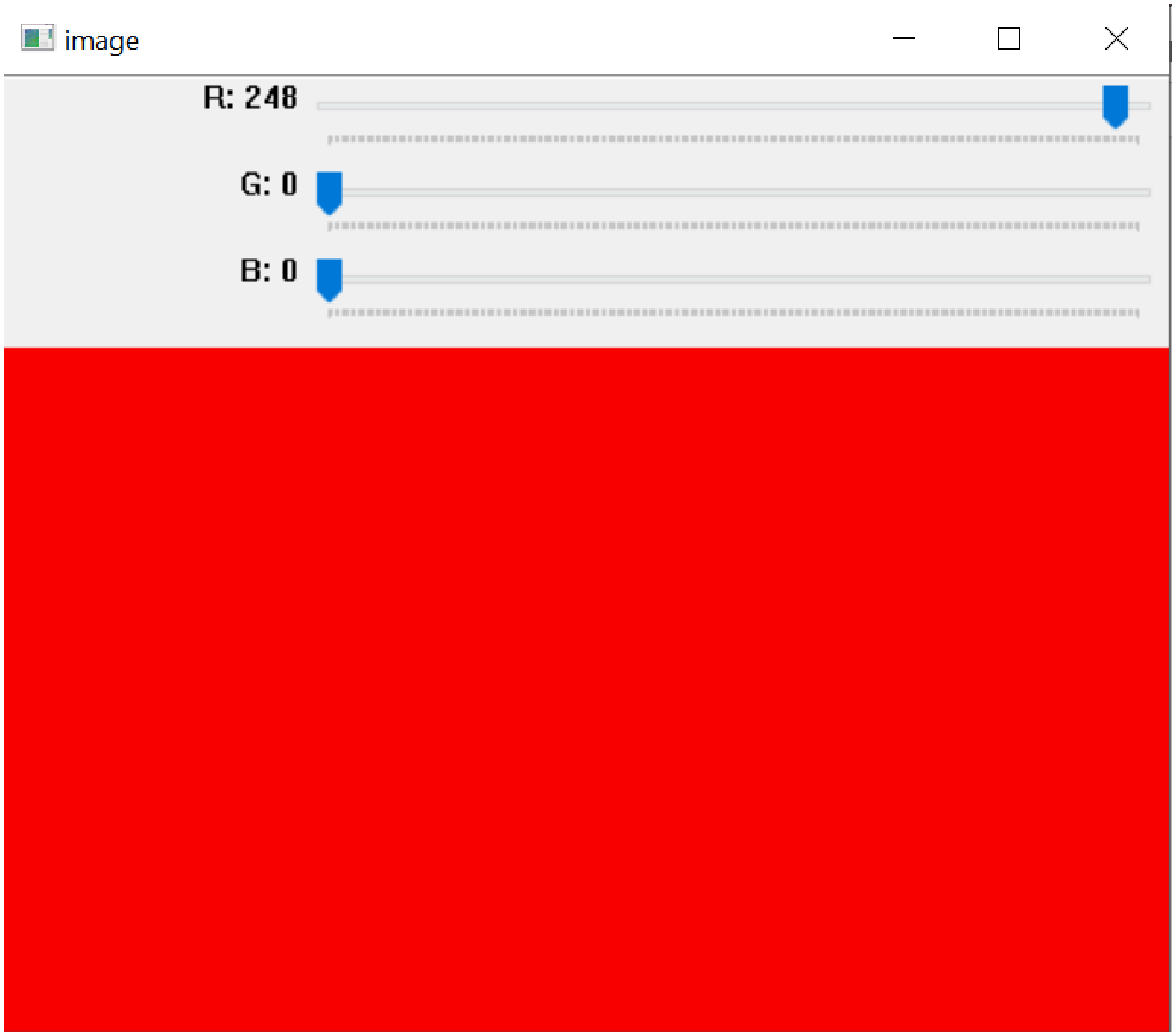
Code:

```python
import cv2
import numpy as np

# Callback function for trackbar (does nothing but required by
createTrackbar)
def nothing(x):
    pass

# Create a black image
img = np.zeros((300, 512, 3), np.uint8)

# Create a window
cv2.namedWindow('image')

# Create trackbars for color change
cv2.createTrackbar('R', 'image', 0, 255, nothing)
cv2.createTrackbar('G', 'image', 0, 255, nothing)
cv2.createTrackbar('B', 'image', 0, 255, nothing)

while True:
    # Get the current positions of the trackbars
    r = cv2.getTrackbarPos('R', 'image')
    g = cv2.getTrackbarPos('G', 'image')
    b = cv2.getTrackbarPos('B', 'image')

    # Update the image color
    img[:] = [b, g, r]

    # Display the image
    cv2.imshow('image', img)

    # Break the loop when 'ESC' key is pressed
    if cv2.waitKey(1) & 0xFF == 27:  # 27 is the ASCII code for the ESC
key
```

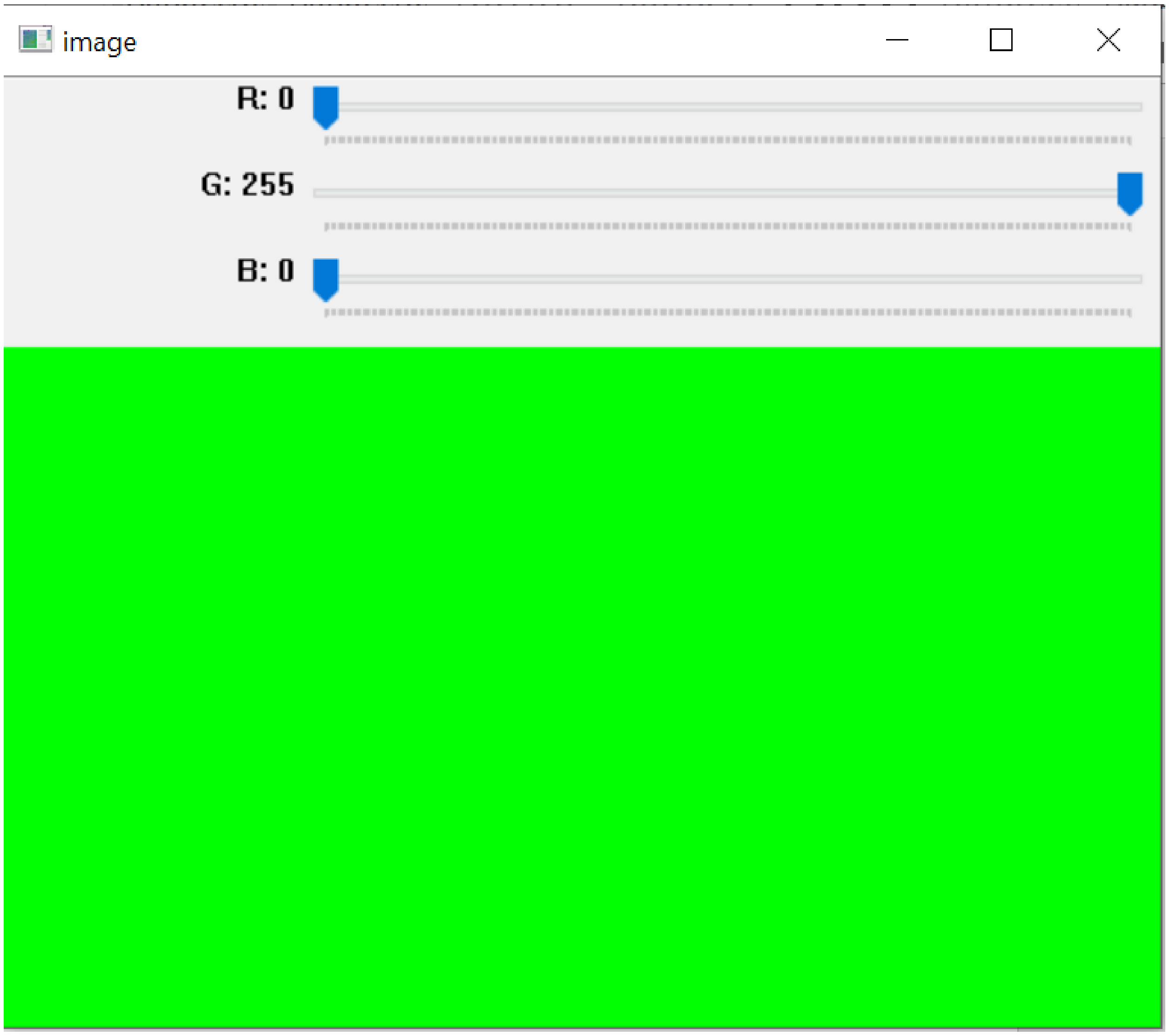
```
        break

# Destroy all windows
cv2.destroyAllWindows()
```

OUTPUT:

R:



G:

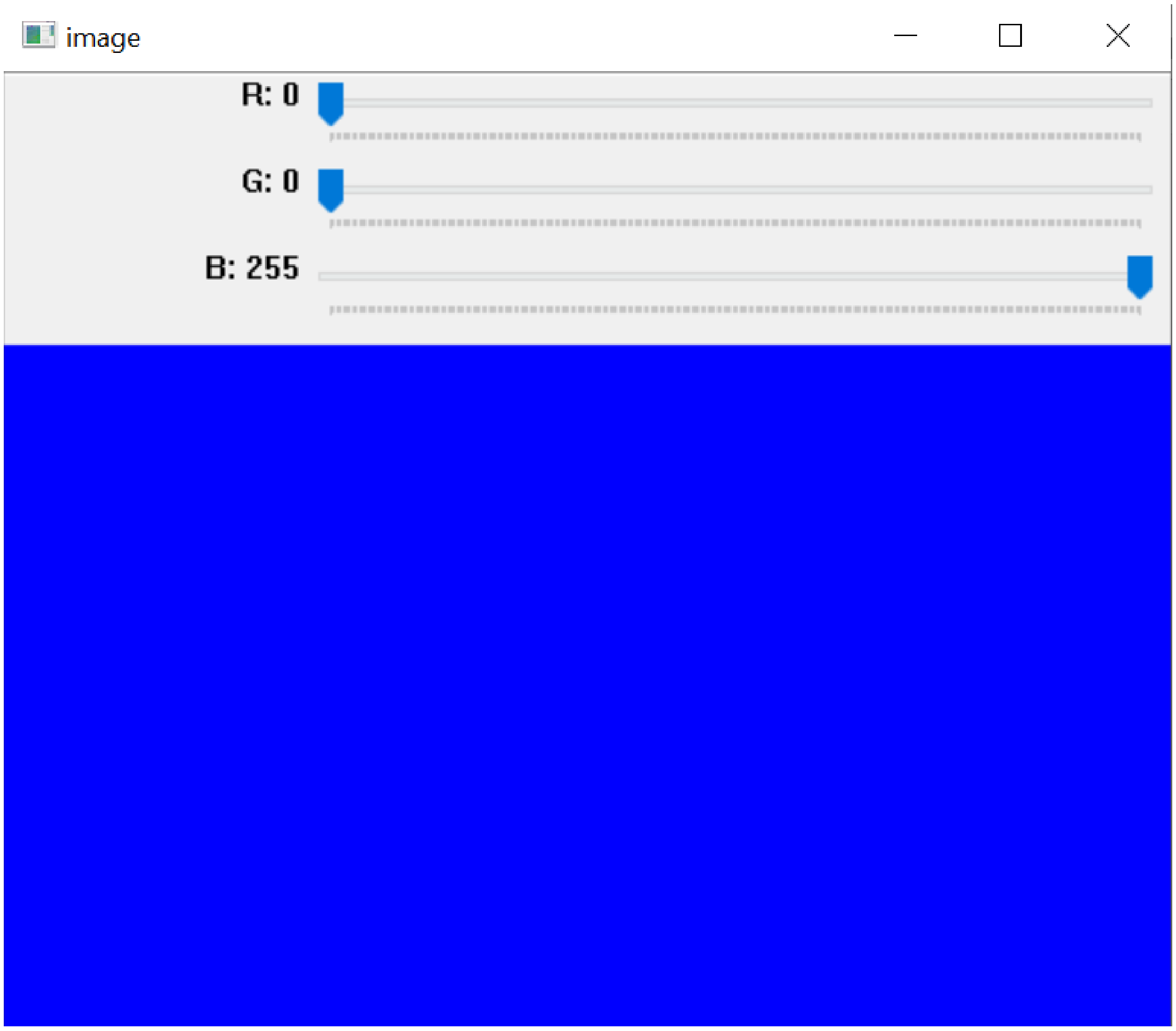


R: 0

G: 255

B: 0

B:



RGB Mixture:

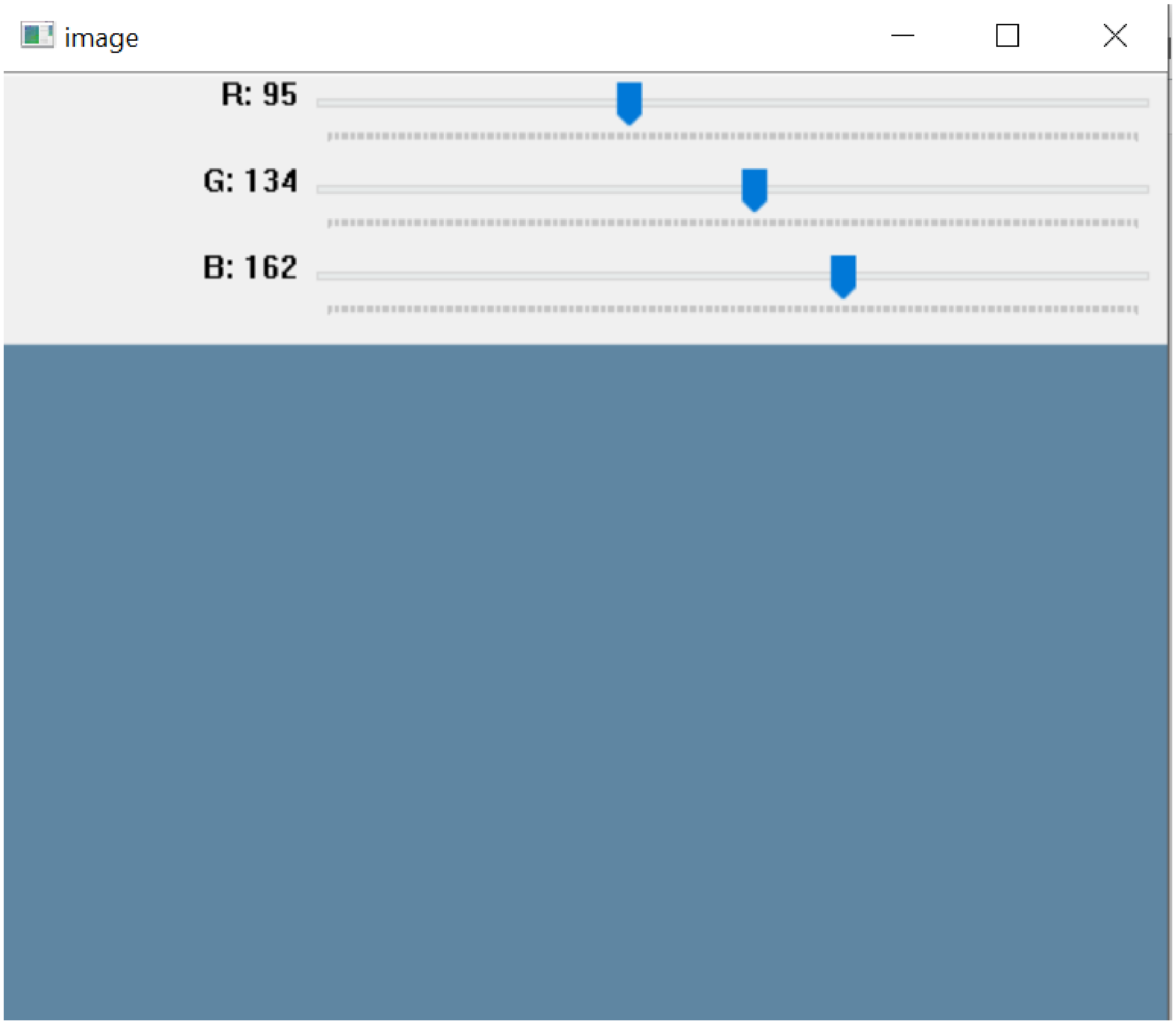


image
R: 95
G: 134
B: 162

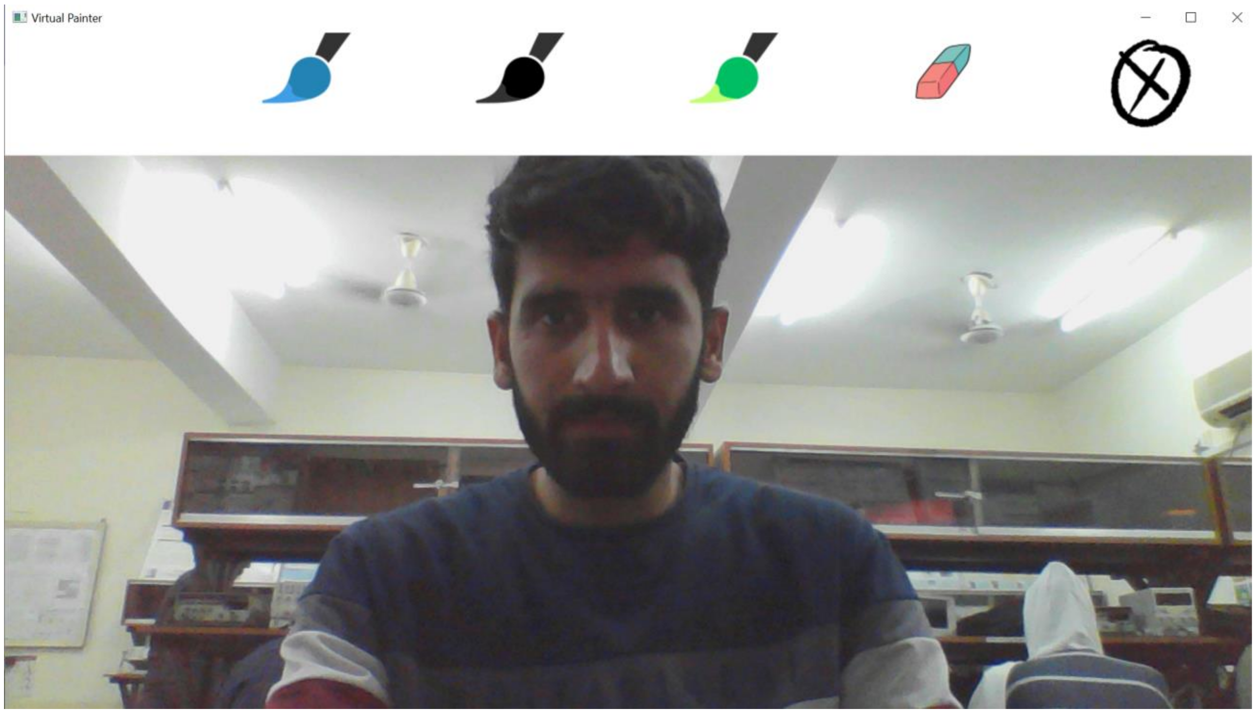
Q3:Try to draw the block diagram of your dip lab project and identify the core block of it and implement only that using python programming language?

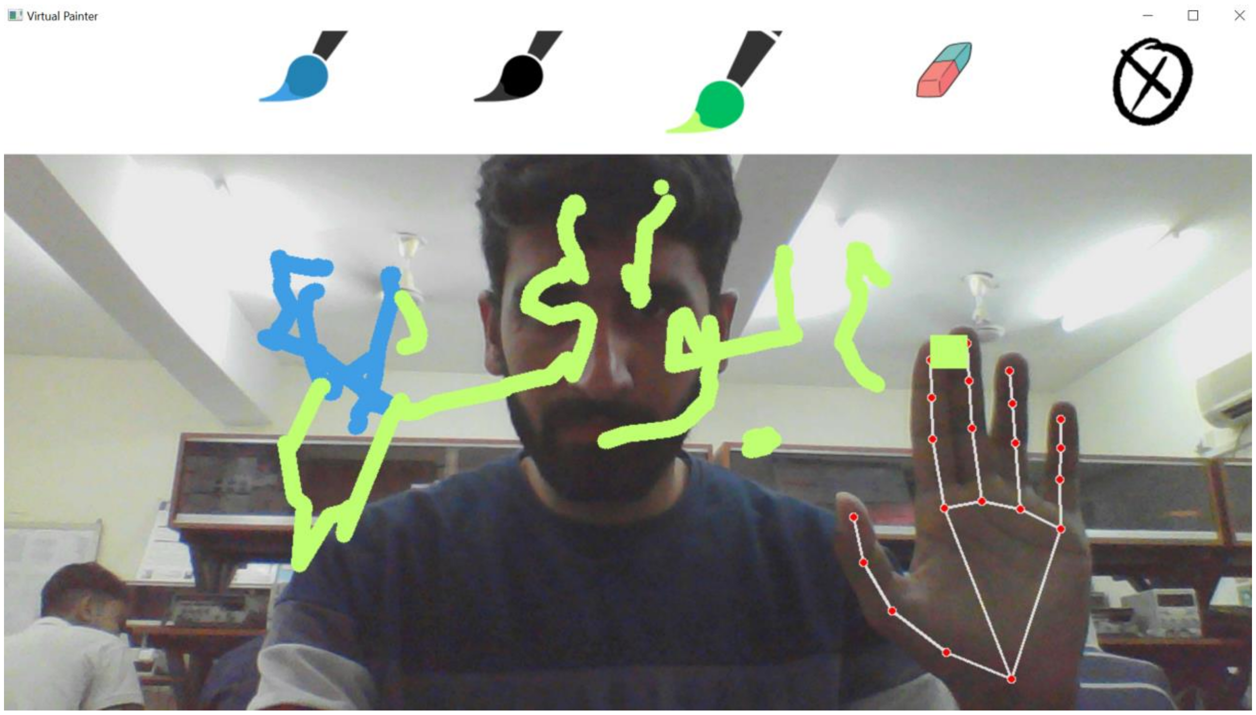
Code:

//Virtual painter:

```
import cv2
import numpy as np
import os
import mediapipe as mp

folderPath = "assets"
myList = os.listdir(folderPath)
overLayList = []

for imPath in myList:
    image = cv2.imread(f'{folderPath}/{imPath}')
    overLayList.append(image)

header = overLayList[0]
brushThickness = 15
eraserThickness = 50
drawColor = (255, 255, 255)
cap = cv2.VideoCapture(1)
cap.set(3, 1280)
cap.set(4, 1280)

initHand = mp.solutions.hands
mainHand = initHand.Hands()
draw = mp.solutions.drawing_utils

def handLandmarks(colorImg):
    landmarkList = []
    landmarkPositions = mainHand.process(colorImg)
    landmarkChek = landmarkPositions.multi_hand_landmarks
    if landmarkChek:
        for hand in landmarkChek:
            for index, landmark in enumerate(hand.landmark):  # Change
here
                draw.draw_landmarks(img, hand, initHand.HAND_CONNECTIONS)
```

```python
                landmarkList.append([index, int(landmark.x*1280),
int(landmark.y*720)])
    return landmarkList

imgCanvas = np.zeros((720, 1280, 3), np.uint8)

xp, yp = 0, 0
while True:
    success, img = cap.read()

    # Find Hand Landmarks
    finger_landmarks = handLandmarks(img)

    # Chack which fingers are up
    if finger_landmarks:
        # Tip of index finger
        x1, y1 = finger_landmarks[8][1:]
        # Tip of middle finger
        x2, y2 = finger_landmarks[12][1:]

        # If selection mode "2 fingers are up"
        if finger_landmarks[8][2] < finger_landmarks[7][2] and
finger_landmarks[12][2] < finger_landmarks[11][2]:
            cv2.rectangle(img, (x1, y1-25), (x2, y2+25), drawColor,
cv2.FILLED)
            xp, yp = 0, 0
            if y1 < 125:
                if 900 < x1 < 1020:
                    # blue
                    header = overLayList[4]
                    drawColor = (230, 158, 63)
                if 700 < x1 < 800:
                    # black
                    header = overLayList[3]
                    drawColor = (51, 51, 51)
                if 500 < x1 < 600:
                    # green
                    header = overLayList[2]
                    drawColor = (114, 255, 193)
                if 300 < x1 < 400:
                    # eraser
                    header = overLayList[1]
                    drawColor = (0, 0, 0)
                if x1 < 200:
                    imgCanvas = np.zeros((720, 1280, 3), np.uint8)

        # If the drawing mode "index finger is up"
        if finger_landmarks[8][2] < finger_landmarks[7][2] and
finger_landmarks[12][2] > finger_landmarks[11][2]:
            cv2.circle(img, (x1, y1), 15, drawColor, cv2.FILLED)

            if xp == 0 and yp == 0:
                xp, yp = x1, y1

            if drawColor == (0, 0, 0):
                cv2.line(img, (xp, yp), (x1, y1), drawColor,
eraserThickness)
                cv2.line(imgCanvas, (xp, yp), (x1, y1), drawColor,
eraserThickness)
            elif drawColor != (255, 255, 255):
                cv2.line(img, (xp, yp), (x1, y1), drawColor,
brushThickness)
                cv2.line(imgCanvas, (xp, yp), (x1, y1), drawColor,
brushThickness)

            xp, yp = x1, y1

    imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR_BGR2GRAY)
    _, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH_BINARY_INV)
    imgInv = cv2.cvtColor(imgInv, cv2.COLOR_GRAY2BGR)
    img = cv2.bitwise_and(img, imgInv)
    img = cv2.bitwise_or(img, imgCanvas)

    # Flip the frame and setting the header image
    img = cv2.flip(img, 1)
    img[0:125, 0:1280] = header

    cv2.imshow("Virtual Painter", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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


Virtual Painter


Virtual Painter

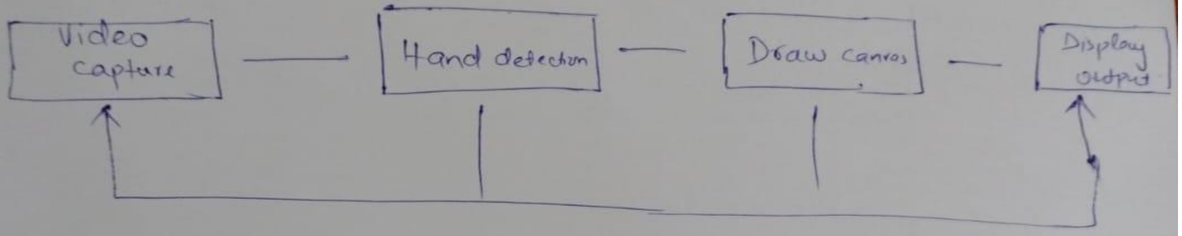
Block diagram:

Question :- 3

Block diagram :-

project :- virtual painter using OpenCV

Diagram :-



Flow :-

→ capture video using web cam

→ Hand detection block detects and track hand landmarks using mediapipe

→ Draw canvas block maintain canvas where user can draw using the fingers

→ Display output block shows the final output, including the drawing canvas and any additional elements.