A REPORT

ON

RESISTOR SCANNER APP

BY

Rishi Jain 2020A8PS1772G

AT

CEERI Pilani

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(JULY 2022)

A REPORT

ON

RESISTOR SCANNER APP

BY

Rishi Jain	2020A8PS1772G	Electronics and Instrumentation

Prepared in partial fulfilment of the
Practice School-I Course Nos.
BITS C221/BITS C231/BITS C241

AT
CEERI Pilani
A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(JULY 2022)

Acknowledgements

I want to thank CEERI Pilani for giving me this opportunity to work with one of the greatest teams of scientists in the country. I would also like to express my gratitude to our project mentors, Dr. Gaurav Purohit and Mr. Pramod Tanwar, for giving me this excellent opportunity to do a project on building the RESISTOR SCANNER APP platform and for helping me learn different tools required for building the platform.

I want to thank BITS Pilani for giving me this once-in-a-lifetime opportunity of working with such a prestigious institute Central Electronics Engineering Research Institute, Pilani. I want to thank the PS division for giving me this chance. I want to express my sincere thanks to my PS faculty, Dr. Sandeep Joshi, for supporting me, guiding me in my journey through PS1, and teaching me how to be successful in a professional environment and improve my research skills. He has been extremely patient and understanding in guiding me throughout the length of the Practice School-1 programme.

Abstract Sheet

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

(RAJASTHAN)

Practice School Division

Station:	CSIR:CEERI	**Centre** :PILANI	
Duration:	2 MONTHS	**Date of Start.** 30TH MAY	
Date of Submission:	22nd July 2022		
Title of the Project:	RESISTOR SCANNER APP		
ID No./Name/ Discipline of the student	2020A8PS1772G, Rishi Jain, Electronics & Instrumentation		
Name(s) and designation(s) of the expert(s):	Dr. Gaurav Purohit, Scientist and Mr. Pramod Tanwar**,** Principal Scientist, CSIR CEERI PILANI		
Name(s) of the PS Faculty:	Dr. Sandeep Joshi		
Key Words:	Artificial Reality, Blender, Circuit Design, OpenCV		
Project Areas:	3D-Modeling, Using AR and VR tools to design apps, Android app Development, OpenCV, Computer Vision		

Abstract: After learning Blender for the Pre-Mid-Semester part, I was given the task to create An App using which we can calculate the resistance of a colour-coded resistor just by pointing a camera at it. I decided to use Android Studio for designing and OpenCV to code the Algorithm for converting the image of the resistor to the value of its resistance. For designing the algorithm, I used the Imgproc and Core module of the OpenCV library. For processing the image, the algorithm crops the image first and then finds the relative positions of the colour bands by finding the contours and then calculate the resistance using the formula in Appendix – A.

Signature of Student

Date: 22/07/2022

Dr. Sandeep Joshi

Signature of PS Faculty

Date: 22/07/2022

Table of contents

Introduction

About the station (CEERI Pilani)

Central Electronics Engineering Research Institute (CEERI) is a national laboratory established first in Pilani to advance Research and Development in the field of Electronics. The laboratory has immensely contributed to the country's electronics and allied engineering growth. It has established state-of-the-art infrastructure in the R & D areas of Microwave Tubes, Plasma Devices, MEMS and Microsensors, Optoelectronics Devices, Microelectronic Processing and Fabrication, VLSI Design, LTCC Technology, Nano Structures, Power Electronics, Industrial Process Control, Agri-electronics, Instrumentation and Embedded Systems and serving the nation since its inception.
My mentors at the CEERI Pilani were Mr. Pramod Kumar Tanwar and Dr. Gaurav Purohit. Mr. Pramod Tanwar is a Principal Scientist at CEERI Pilani, and his area of interest are Cyber-Physical Systems, Internet of Things (IoT), Computer Architecture, Robotic Vision, FPGAs & GPUs. Dr. Gaurav Purohit is a Senior Scientist at CEERI Pilani. His areas of Interest are Digital Baseband Architecture for Software Defined Radio 6G/5G and B5G Wireless standards, Intelligent data analytics using AI/ML algorithms for Air quality monitoring, Tele-health, and drone tracking applications.

Problem statement

As the report's title suggests, the project aimed to design an app that can scan a colour-coded resistor and display its resistance. The task that was assigned was to create an android app using Android Studio and to create an image processing Algorithm that can scan the resistor and display its value.

Computer Vision and OpenCV

Computer Vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision allows them to see, observe and understand [1].

Computer Vision works similar to human vision; one disadvantage that computer vision has over human vision is that humans have a lifetime of training and context and are far superior in telling objects apart using a massive list of features. On the other hand, computer vision needs extensive training to recognise even small features like edges.

A Good computer Vision model needs a lot of data and trains itself on the data over and over until it discerns distinction and ultimately recognises images. Two essential technologies used to accomplish this are Deep Learning and Convolutional Neural Network (CNN). Deep learning uses algorithmic models that enable a computer to teach itself about the context of visual data. A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels.

Figure 1: OpenCV logo [2]

OpenCV is a huge open-source library for Computer Vision, Machine Learning, and image processing. Now, it plays a major role in real-time operation, which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When integrated with various libraries, such as NumPy, python can process the OpenCV array structure for analysis. We use vector space to identify image patterns and their multiple features and perform mathematical operations on these features [3]. OpenCV can help in object detection, pattern recognition, etc. It has applications in 2D and 3D feature toolkits, Facial recognition systems, Gesture recognition, Objection detection, etc. OpenCV is written in C++, but there are bindings in Python and Java. OpenCV features GPU acceleration for real-time operation.

Main Text

Blender

Figure 2: Blender Logo [4]

Blender is a free and open-source 3D computer graphics software tool used to create animated films, visual effects, art, 3D-printed models, motion graphics, interactive 3D applications, virtual reality, and video games. Blender's features include 3D modelling, UV mapping, texturing, digital drawing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animation, match moving, rendering, motion graphics, video editing, and compositing.

The project was presented to us as an AR/VR project by our mentors, so during the initial days of our Practice School-I, we were told to learn to use Blender. For learning Blender, I have used two resources, Blender's official documentation and a tutorial on YouTube. The tutorial on YouTube which I used to learn Blender is Blender 3.0 Beginner Donut Tutorial. This tutorial was made available by Andrew Price on his YouTube channel named Blender Guru.

I started with an introduction to the user interface of Blender, and I learned small things like how to interact with any object and how to move about in the workspace. Then for the coming few days learned tools like Modifiers, Sculpting, Painting, Texturing, Shading, Geometry Nodes, etc.

I also learned about different rendering engines shipped with Blender, i.e., Cycles and Eevee. The first Digital asset that I made was a doughnut.

Figure 2: Rendered version of the Doughnut that I made.

The layout of the App

The project required us to build an android app; I decided to use Android Studio for that. Other alternatives were present for making the app, e.g., Eclipse, coding in python and converting that to an android app using Buildozer and Python for Android. I decided to use Android Studio because it is straightforward to use. Android Studio is the official IDE for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development [5]. The advantage of using Android Studio lies in the fact that it has such a great collection of tools that are present to cater for the needs of an android app developer.

Coming to the app I built, for the start-up page of the app, I started with an empty activity and then added a ListView on top of the Constraint View (Present by default). Then I populated the list view with buttons to start the two modules present in the app as instructed by Mr. Pramod Tanwar. The ‘Resistor Scanner’ and the ‘Minimum Resistor Calculator’ can be accessed using the button named ‘Start’ present on their respective ListView.

Figure 3: Screenshot of the Start-up page of the App

The ‘Resistor Scanner’ can be launched by clicking on the ‘START’.

A new activity will launch with a camera view.

Figure 5: Screenshot of working Resistor Scanner

Working of the Scanner

The basic functioning of the Scanner depends on the OpenCV module specially designed for the Android platform. OpenCV for android can be downloaded from opencv.org and can be imported to Android Studio in just a few steps. Using OpenCV with java on android can be a bit more complex than using it with python, especially for a beginner, due to the lack of resources and documentation to help users. I used the JavaCamera2View and CameraBridgeViewBase Classes to take input from the camera. CameraBridgeViewBase is a basic class that implements the interaction with Camera and the OpenCV library. The input from the camera is taken through the enableView() function and sent to an image processing class where the colours on the resistor are detected. After detecting the colours on the resistor, the value of the resistance is calculated. To see the calculation involved, refer to Appendix A.

The algorithm used to calculate the resistance

To calculate the resistance of a colour-coded resistor, it is not only required to detect the colour of the bands present on the resistor but also to estimate its relative position along the X-axis (refer to the Appendix A). The colour of the bands can be found directly by looking at the array representation of any image, leaving us the task of estimating the relative location of the colour bands. Any image can be represented by a 3-D array where a triplet of integer represents each pixel; in RGB space, an integer value in the triplet represents the intensity of red, blue, or green colour in that pixel.

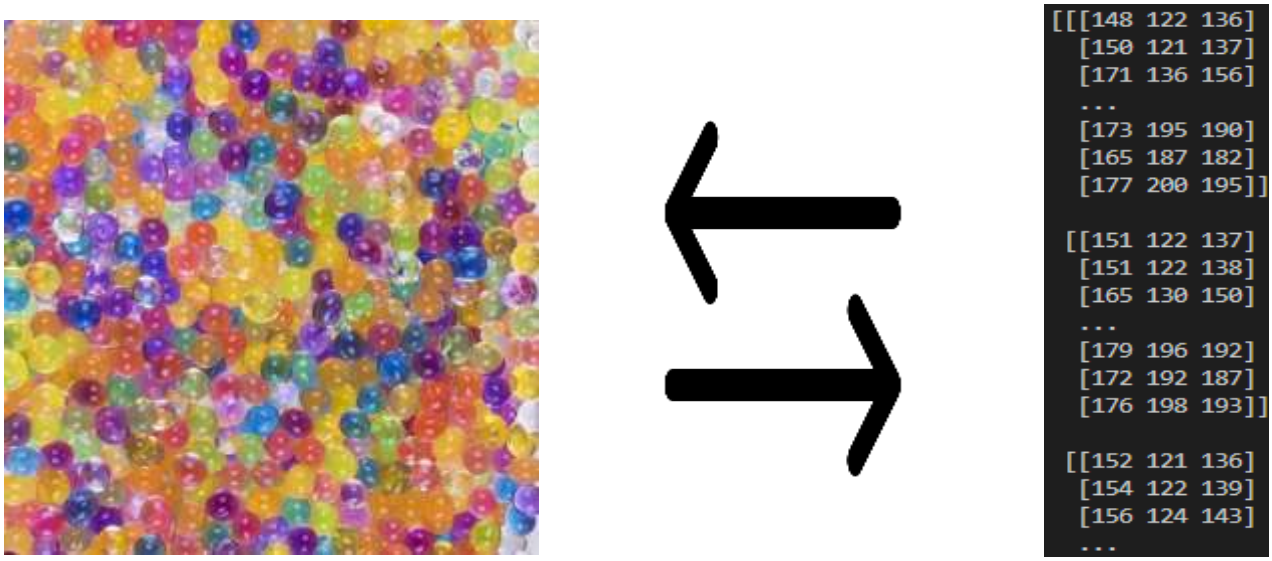


Figure 6: Representation of a coloured image as a 3-dimensional array

To write the code for estimating the relative positions of the colour bands, I referred to this - https://github.com/thegouger/ResistorScanner . If we try to process the complete frame and process it, that would consume a lot of resources, and the program would take a very long time to process the whole image. On top of that, the algorithm would also need to perform object detection to locate the resistor in the captured frame. To solve this problem, I decided to take only a small rectangular part of the Image and process it instead. The camera view has a red rectangle at the centre of the frame to guide the user in positioning the camera correctly to capture the resistor.

Figure 7: Example of a resistor being scanned using the rectangle

Processing of the cropped image

The data type of the captured image is Mat. The class Mat represents an n-dimensional dense numerical single-channel or multi-channel array [6]. Since we only need a small part of the image, it is cropped using the submat() function in the 'core' class of OpenCV. The processing of the image starts by applying the Bilateral Filter to remove the noise and better edge detection.

Bilateral Filtering - A bilateral filter is used for smoothening images and reducing noise while preserving edges [7]. Its formulation is simple: each pixel is replaced by a weighted average of its neighbours. This aspect is crucial because it makes it easy to acquire intuition about its behaviour, to adapt it to application-specific requirements, and to implement it. The Bilateral filter is defined as

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Where W_p is

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Original Image *After applying Bilateral Filter*

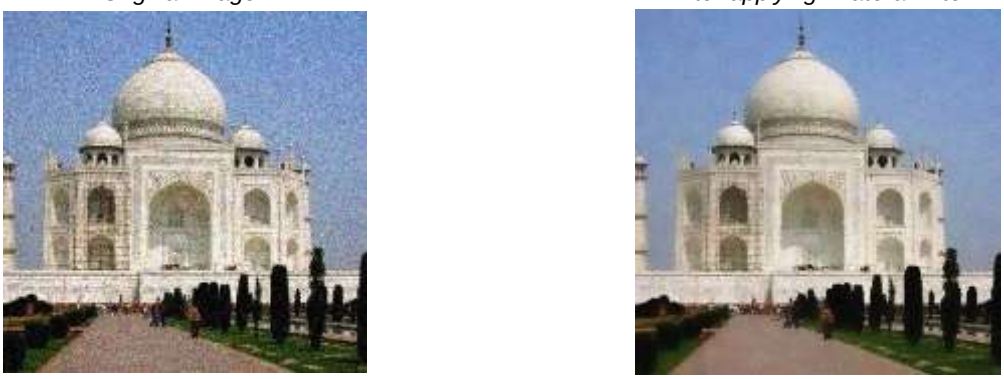


Figure 8: Example of Bilateral Filtering [7]

Bilateral filtering makes detecting the colour bands easier by reducing the noise.

After applying the bilateral filter, the image is converted from RGB colour space to HSV colour space. The remaining processing is done using HSV colour space because it is easy to define colour ranges in HSV. It is necessary to define ranges for colour because there are different shades of colour, e.g. you cannot expect two resistors to have the same shade of red or any other colour. Even if the shades are the same, the light received by the camera sensor entirely depends on the environment; even a slight difference in the light received by the sensor can change the RGB values of a pixel.

HSV colour space – It defines a type of colour space. It is similar to the modern RGB model. The HSV colour space has three components: hue, saturation and value. In HSV, the hue represents colour. In this model, the hue is an angle from 0 degrees to 360 degrees. Saturation indicates the range of grey in the colour space. It ranges from 0 to 100%. Value is the brightness of the colour and varies with colour saturation. It ranges from 0 to 100%. The colour space will be totally black when the value is '0'. With the increase in the value, the colour space brightness up and shows various colours.

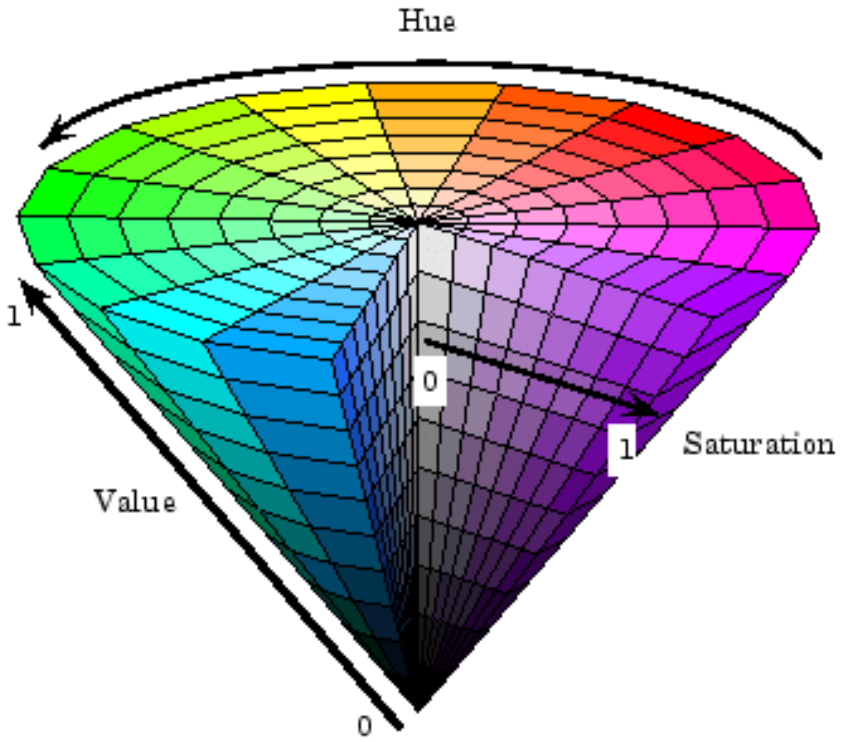


Figure 9: HSV colour space represented as a cone [8]

The colour ranges used in the app are given in Appendix B.

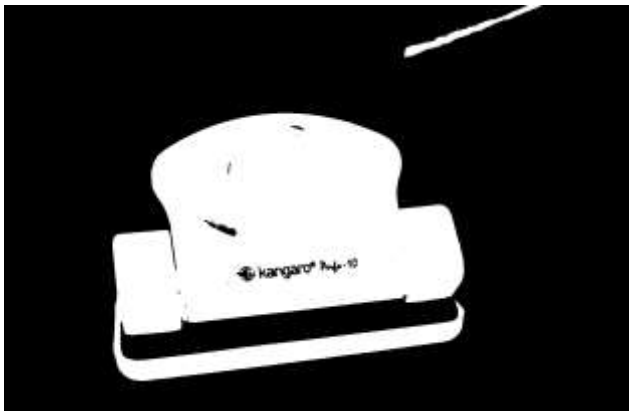
The next step is to create a mask for every colour using the range of every colour provided in HSV.

Mask - A mask is a binary image consisting of zero- and non-zero values. If a mask is applied to another binary or a grayscale image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged.

I used the inRange() function, using the inRange() function, we can filter all the colours between the range of lower and upper boundaries. For example, to capture the red colour, we can filter out Hue in the range 160-180, and the values and saturation can be in the range 50-255. This will give us different variations of dark-bright and dull-pure red colour variations that are captured [9].

Original

Mask



Mask applied to the original

Figure 10: Example of masking a colour

The masked images are stored in a new variable and then used to find the contours in the next step.

The contours must be detected because we need to find the centroid of the colour bands to determine their coordinate on the X-axis. I used findContours() function of the Imgproc class to find the contours from the mask of each colour.

Contours - Contours are defined as the line joining all the points along the boundary of an image that has the same intensity. Contours come in handy in shape analysis, finding the size of the object of interest, and object detection. OpenCV has findContour() function that helps in extracting the contours from the image [10].

It works best on binary images. that's why I used mask instead of the original image for detecting contour

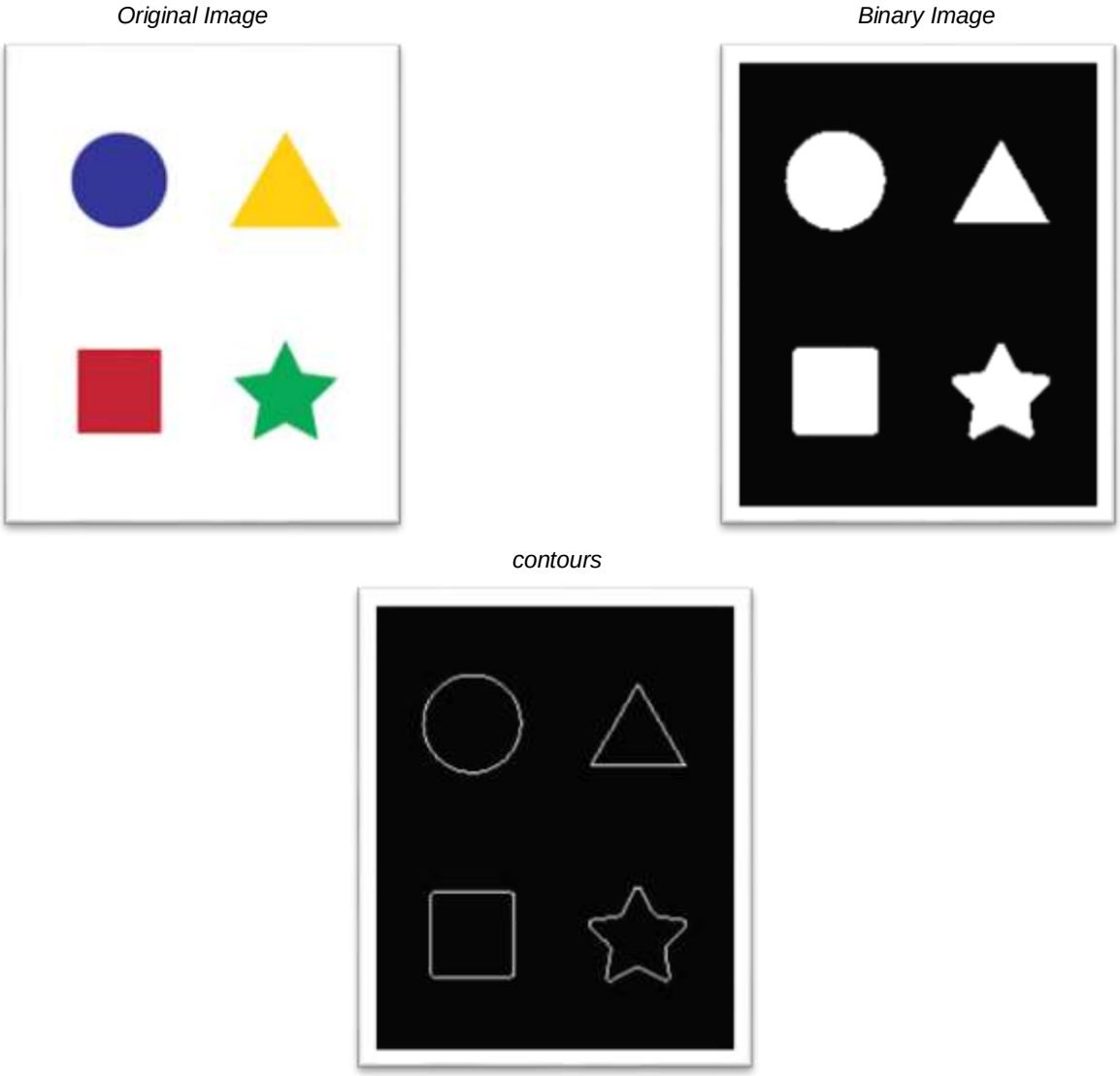


Figure 11: Drawing Contours from a coloured image [11]

After finding the contours, the coordinates of the centres are calculated using Moments. In OpenCV, moments are the average of an image's pixels intensities. The centroid of the contour is calculated using this formula, $C_x = m_{10} / m_{00}$.

Moments m_{ij} are calculated as:

$$mij = \sum_{x,y}(\text{array}(x, y) * x^j * y^i)$$

The x-coordinate of the centroids is saved in an array along with the index of the colour then the actual value of the resistance is calculated and displayed on the screen.

For version control, I used git. I made a private GitHub repository. For all the other git-related work, I used the vcs functionality present inside Android Studio itself. Android Studio provides a feature that can make a repository, commit any changes, push them, etc., without writing a single line of code.

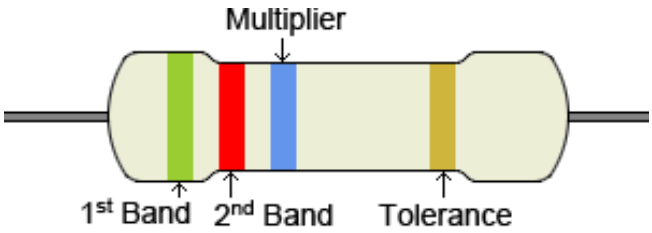
Conclusion and Recommendations

The final outcome of the project was the Resistor Scanner app, this app can be used by anyone while handling resistors. I learned a lot of things while working on this project including but not limited to Android Studio and OpenCV for android app development. My PS mentor, Mr. Pramod Tanwar wanted to add another module to the app but due to the time constraint, I could not make that. If given the chance, I would like to work on the app even after the PS ends. He has also asked me to work on a web version of the scanner app. Other members of my group were working on AR-Based Circuit design which combined with the scanner could become a really useful tool.

Appendix – A

For a 4 Band resistor first and the second band gives the first two significant digits, the third band gives the multiplier and the fourth band gives the tolerance.

Table 1: Table to convert colour code to resistance [12]



Color	1st, 2nd Band Significant Figures	Multiplier	Tolerance
Black	0	× 1	
Brown	1	× 10	±1% (F)
Red	2	× 100	±2% (G)
Orange	3	× 1K	±0.05% (W)
Yellow	4	× 10K	±0.02% (P)
Green	5	× 100K	±0.5% (D)
Blue	6	× 1M	±0.25% (C)
Violet	7	× 10M	±0.1% (B)
Grey	8	× 100M	±0.01% (L)
White	9	× 1G	
Gold		× 0.1	±5% (J)
Silver		× 0.01	±10% (K)

Formula – *((First significant digit*10) + (second significant digit)) *Multiplier*

Appendix – B

Table 2: Ranges used for different colours

Colour	Range of colour in HSV	
	Lower Limit	**Upper Limit**
Black	(0, 0, 0)	(180, 250, 50)
Brown	(0, 31, 41)	(25, 250, 99)
Red1	(0, 65, 60)	(8, 100, 100)
Red2	(158, 65, 50)	(180, 250, 150)
Orange	(7, 150, 150)	(18, 250, 250)
Yellow	(25, 130, 100)	(34, 250, 160)
Green	(35, 60, 60)	(75, 250, 150)
Blue	(80, 50, 50)	(106, 250, 150)
Purple	(130, 60, 50)	(165, 250, 150)
Gray	(0, 0, 50)	(180, 50, 80)
White	(0, 0, 90)	(180, 15, 140)

References

[1] What is Computer Vision?

[2] OpenCV logo

[3] https://www.geeksforgeeks.org/opencv-overview/

[4] https://commons.wikimedia.org/wiki/File:Logo_Blender.svg

[5] Android Studio

[6] https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html#details

[7] https://www.geeksforgeeks.org/python-bilateral-filtering/

[8] https://www.researchgate.net/figure/Illustration-of-the-HSV-Color-Space-B-Color-Feature-Extraction-Color-feature-is-extracted_fig1_321126312

[9] https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/

[10] https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/#:~:text=Contours%20are%20defined%20as%20the,the%20contours%20from%20the%20image.

[11] https://www.geeksforgeeks.org/python-opencv-find-center-of-contour/

[12] https://www.calculator.net/resistor-calculator.html

Glossary

AR – Artificial Reality.

VR – Virtual Reality.

OpenCV – Open-source Computer vision library.

VCS – Version control system. In software engineering, version control is a class of systems responsible for managing changes to computer programs.

RGB – Red, Blue and Green. A colour space.

HSV – Hue, Saturation, Value. Another colour space similar to RGB.

IDE - An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.