

# **PySolderInspect DL: Aplikasi Deteksi Cacat Solder pada *Printed Circuit Board* (PCB)**

## **Berbasis *Deep Learning***

Buku Manual Penggunaan Aplikasi



### **Ditulis Oleh**

1. Eko Rudiawan Jamzuri
2. Charlie Rolando Adrian Tamba
3. Muhammad Syah Hari Arrasyid
4. Annisa Salsabilla
5. Dimas Rizky Saputra
6. Dani Muti Aziz
7. Syadiba Puan Shazinta
8. Fitria Salsabillah
9. Hasnira

**Politeknik Negeri Batam  
2024**

## Kata Pengantar

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku manual penggunaan aplikasi PySolderInspect DL ini dapat diselesaikan. Manual ini kami susun sebagai panduan bagi pengguna dalam mengoperasikan aplikasi PySolderInspect DL yang dirancang untuk mendeteksi cacat pada solder di *Printed Circuit Board* (PCB) secara otomatis dengan menggunakan teknologi *Deep Learning*.

Aplikasi ini dikembangkan dengan tujuan untuk membantu meningkatkan efisiensi dan akurasi dalam inspeksi visual pada proses produksi elektronik, terutama pada komponen PCB yang memiliki standar kualitas yang tinggi. Diharapkan, manual ini dapat membantu para pengguna memahami langkah-langkah instalasi, pengoperasian, serta konfigurasi aplikasi ini secara efektif. Setiap langkah dijelaskan secara rinci untuk memastikan aplikasi dapat dioperasikan dengan mudah dan optimal.

Ucapan terima kasih kami sampaikan kepada semua pihak yang telah berkontribusi dalam pengembangan aplikasi dan penyusunan buku ini. Kami berharap buku manual ini dapat bermanfaat dan menjadi panduan yang dapat diandalkan dalam proses inspeksi PCB.

Batam, 10 November 2024



Eko Rudiawan Jamzuri

(Tim Pengembang)

## **Daftar Isi**

Kata Pengantar .....	2
Daftar Isi .....	3
Daftar Gambar .....	4
1. Instalasi Aplikasi.....	5
1.1. Instalasi pylon Viewer.....	5
1.2. Instalasi Python dan Paket Pendukung .....	6
1.3. Instalasi Aplikasi .....	6
2. Pengoperasian Aplikasi.....	7
3. <i>Source Code</i> Aplikasi.....	11
3.1. Tautan <i>Source Code</i> Aplikasi .....	11
3.2. Skrip Aplikasi.....	11

## **Daftar Gambar**

Gambar 1. Tampilan aplikasi pylon Viewer.....	5
Gambar 2. Hasil tangkapan gambar pada aplikasi pylon Viewer.....	5
Gambar 3. Tampilan versi Python yang telah ter- <i>install</i> pada tablet PC. ....	6
Gambar 4. Koneksi kamera Basler ke USB. ....	7
Gambar 5. Koneksi Coral USB Accelerator ke tablet PC.....	7
Gambar 6. Tombol power pada tablet PC .....	7
Gambar 7. Tampilan ikon aplikasi PySolderInspect DL.....	8
Gambar 8. Penempatan sampel PCB pada alat inspeksi. ....	8
Gambar 9. Tampilan aplikasi PySolderInspect DL.....	9
Gambar 10. Pengaturan fokus dan pencahayaan pada lensa kamera. ....	9
Gambar 11. Hasil deteksi sampel PCB yang REJECT/FAIL.....	10
Gambar 12. Hasil deteksi sampel PCB yang GOOD/PASS.....	10

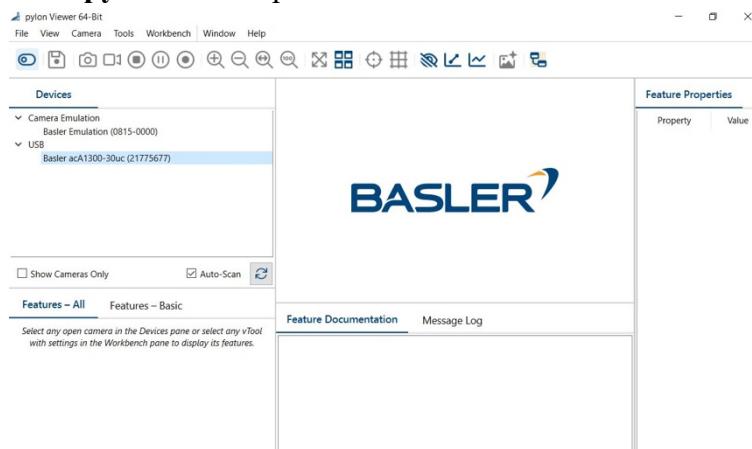
# 1. Instalasi Aplikasi

## 1.1. Instalasi pylon Viewer

Aplikasi **PySolderInspect DL** membutuhkan akses ke perangkat keras kamera Basler USB untuk menangkap gambar dan mendeteksi cacat solder pada tangkapan gambar. Akses ke perangkat keras kamera Basler USB ini membutuhkan instalasi perangkat lunak **pylon Viewer** yang dapat di unduh pada tautan berikut <https://www.baslerweb.com/en/software/pylon/>. Lakukan instalasi sesuai petunjuk pada tautan tersebut.

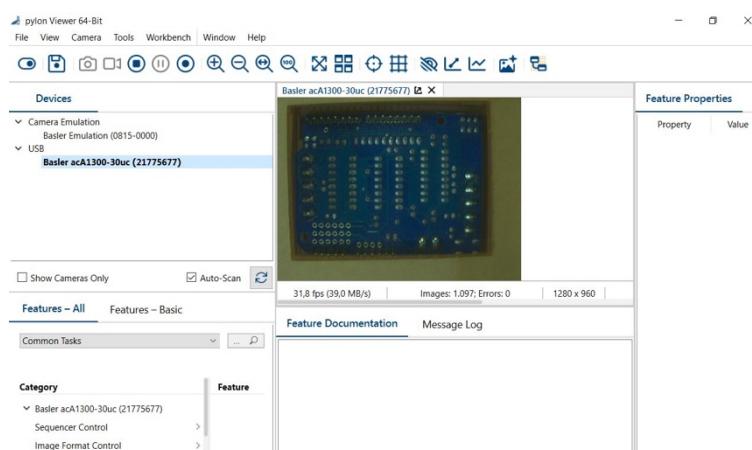
Setelah proses instalasi **pylon Viewer** selesai, selanjutnya lakukan verifikasi terhadap koneksi kamera dengan langkah-langkah sebagai berikut.

- 1) Buka perangkat lunak **pylon Viewer** pada Windows Start Menu.



Gambar 1. Tampilan aplikasi pylon Viewer.

- 2) Tekan tombol Open Camera pada **pylon Viewer** dan pastikan kamera menampilkan tangkapan gambar.



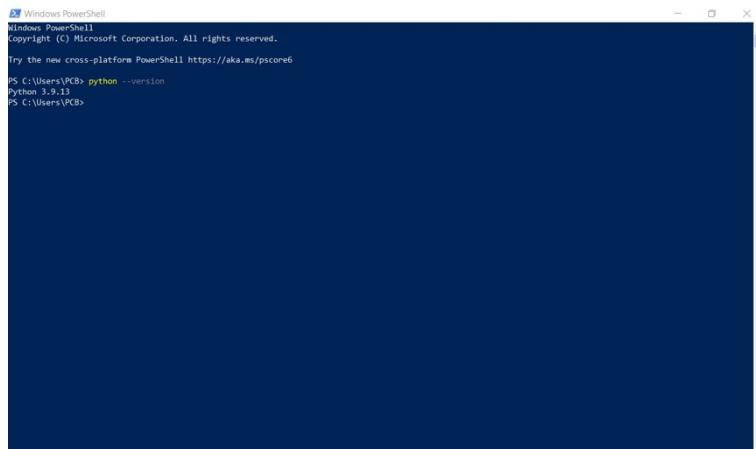
Gambar 2. Hasil tangkapan gambar pada aplikasi pylon Viewer.

- 3) Jika tangkapan gambar tidak tampil pada pylon Viewer, lakukan pengecekan terhadap koneksi kamera.

## 1.2. Instalasi Python dan Paket Pendukung

Aplikasi **PySolderInspect DL** membutuhkan interpreter **Python** untuk menjalankan skrip aplikasi, sehingga interpreter **Python** harus terpasang terlebih dahulu pada tablet PC yang berada pada alat inspeksi. Unduh **Python versi 3.9** dari tautan berikut ini <https://www.python.org/> dan lakukan proses instalasi pada tablet PC yang akan digunakan untuk melakukan proses inspeksi. Setelah instalasi selesai, lakukan verifikasi bahwa **Python** telah ter-*install* dengan cara sebagai berikut.

1. Buka **Command Prompt** (Windows) atau **Terminal** (Linux / MacOS).
2. Ketik perintah `python --version` di terminal dan pastikan terminal menampilkan versi **Python 3.9.xx**.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\PCB> python --version
Python 3.9.13
PS C:\Users\PCB>
```

Gambar 3. Tampilan versi Python yang telah ter-*install* pada tablet PC.

## 1.3. Instalasi Aplikasi

1. Unduh *source code* pada tautan berikut ini.  
<https://github.com/Barelang7/PySolderInspect-DL>
2. Lakukan ekstraksi zip file dari *source code* yang telah di unduh.
3. Buka terminal pada direktori *source code* dan lakukan instalasi paket PyCoral dengan perintah berikut.  
`python3 -m pip install --extra-index-url https://google-coral.github.io/py-repo/ pycoral~=2.0`
4. Lakukan instalasi paket pypylon, tk dan opencv-python dengan perintah berikut.  
`pip install pypylon tk opencv-python`

## 2. Pengoperasian Aplikasi

1. Hubungkan kabel konektor USB dari tablet PC ke USB pada kamera Basler.



Gambar 4. Koneksi kamera Basler ke USB.

2. Hubungkan perangkat **Coral USB Accelerator** ke USB pada tablet PC.



Gambar 5. Koneksi Coral USB Accelerator ke tablet PC.

3. Nyalakan tablet PC dengan menghubungkan adaptor ke konektor catu daya pada tablet PC, kemudian tekan tombol power di bagian samping tablet PC dan tunggu sampai tampilan Windows Desktop muncul.



Gambar 6. Tombol power pada tablet PC

4. Klik ikon `PCB Inspection` untuk menjalankan aplikasi dan tunggu hingga tampilan aplikasi muncul pada layar tablet PC.



Gambar 7. Tampilan ikon aplikasi PySolderInspect DL.

5. Letakkan sampel PCB yang ingin dideteksi pada area penempatan PCB.



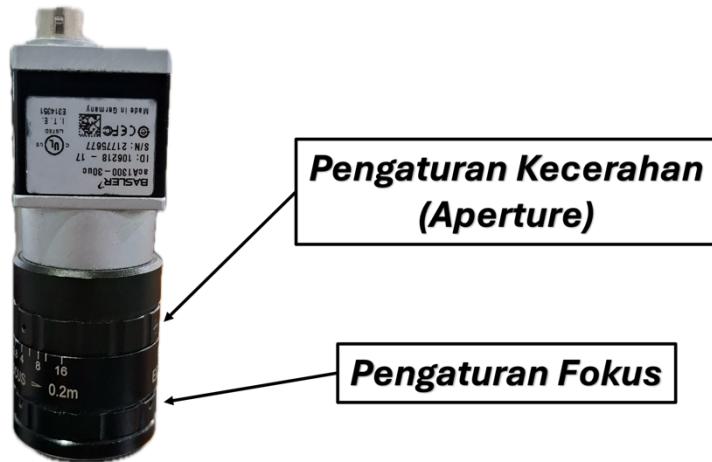
Gambar 8. Penempatan sampel PCB pada alat inspeksi.

6. Lakukan pengambilan sampel gambar PCB dengan cara menekan tombol `Capture Image` pada aplikasi dan pastikan gambar sampel PCB tampil pada aplikasi.



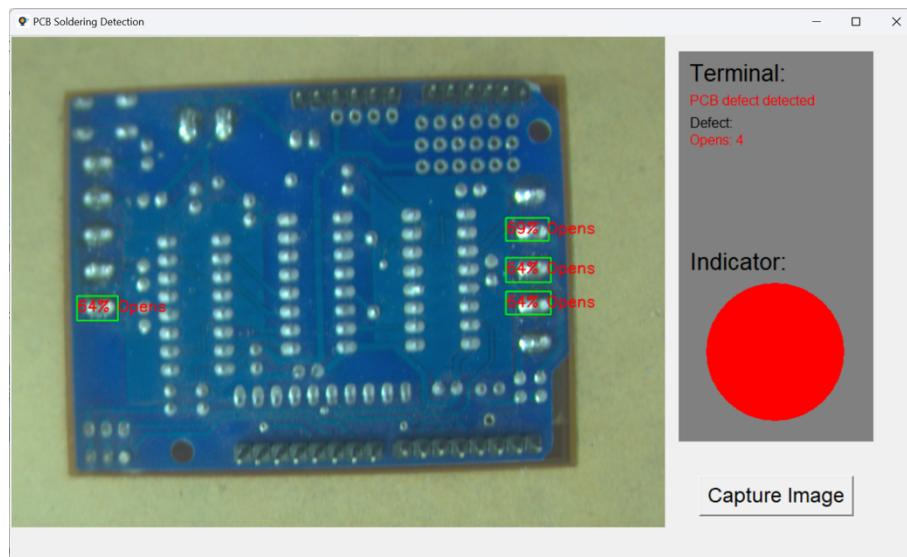
Gambar 9. Tampilan aplikasi PySolderInspect DL.

7. Lakukan pengaturan terhadap lensa kamera dengan cara memutar lensa jika hasil gambar kurang fokus atau terlalu gelap.

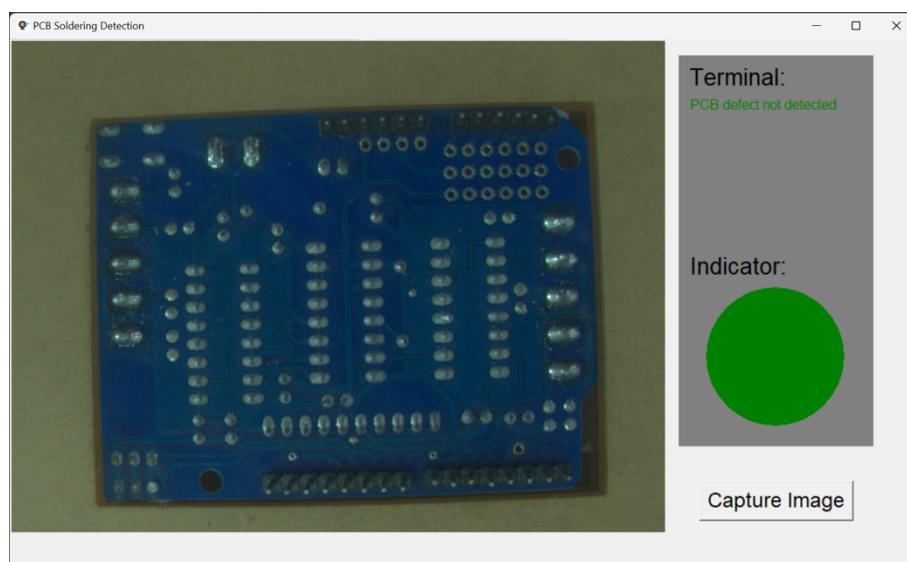


Gambar 10. Pengaturan fokus dan pencahayaan pada lensa kamera.

8. Jika pengaturan telah sesuai, tekan kembali tombol 'Capture Image'. Aplikasi akan memulai proses pendekripsi pada bagian hasil solder pada bagian solder pad PCB. Jika terdapat cacat pada hasil solder, suara alarm akan berbunyi dan indikator berwarna merah akan muncul untuk menandakan kondisi PCB REJECT/FAIL. Sementara itu, jika tidak ada cacat pada hasil solder, indikator berwarna hijau akan muncul untuk menandakan kondisi PCB GOOD/PASS.



Gambar 11. Hasil deteksi sampel PCB yang REJECT/FAIL.



Gambar 12. Hasil deteksi sampel PCB yang GOOD/PASS.

9. Ulangi proses inspeksi untuk beberapa sampel PCB sesuai kebutuhan.
10. Jika proses inspeksi telah selesai, tutup aplikasi dengan cara menekan tombol silang.
11. Matikan tablet PC dengan menekan tombol power.

### 3. Source Code Aplikasi

#### 3.1. Tautan Source Code Aplikasi

Aplikasi **PySolderInspect DL** dapat diunduh pada tautan berikut ini <https://github.com/Barelang7/PySolderInspect-DL>.

#### 3.2. Skrip Aplikasi

Skrip aplikasi **PySolderInspect DL** adalah sebagai berikut.

```
1. from tkinter import *
2. import tkinter
3. import cv2
4. from PIL import Image, ImageTk
5. from pypylon import pylon
6. import winsound
7. import time
8. import threading
9. import argparse
10. import os
11.
12. from pycoral.adapters.common import input_size
13. from pycoral.adapters.detect import get_objects
14. from pycoral.utils.dataset import read_label_file
15. from pycoral.utils.edgetpu import make_interpreter
16. from pycoral.utils.edgetpu import run_inference
17.
18. threshold_ = 0.61
19.
20. def message(message):
21.     # Update teks pada kotak pesan
22.     notif.config(text=message)
23.
24. def toggle_text():
25.     label_c = tkinter.Label(app, text="Indicator:", font=("Helvetica", 20), fg="black",
highlightthickness=0, bg="grey")
26.     label_c.place(x = 790, y = 245)
27.     indicator_frame.place(x=812, y=289)
28.     canvas.place(x = 2000, y = 0)
29.     # Set notification label text
30.     message("Success Capturing!")
31.     notif.place(x=842, y=489)
32.     # After 2500 milliseconds (2.5 seconds), reset notification label
33.     app.after(2500, lambda: notif.place(x=2000, y=65))
34.
35.     global camera
36.     global args
37.     global interpreter
38.     global inference_size
39.     grab_result = camera.RetrieveResult(5000, pylon.TimeoutHandling_ThrowException)
40.
41.     label_c1["text"] = "PCB defect not detected"
42.     label_c1.configure(fg="green")
43.
44.     # Create circle
45.     indicator_canvas.create_oval(0, 0, 160, 160, fill="green", outline='green')
46.
47.     label_c2.place(x = 2000, y = 80)
48.     label_c3.place(x = 2000, y = 80)
49.     label_c4.place(x = 2000, y = 80)
50.     label_c5.place(x = 2000, y = 80)
51.
52.     if grab_result.GrabSucceeded():
53.         # Play a sound when image is captured
54.         winsound.PlaySound("success_sound.wav", winsound.SND_ASYNC)
55.         # Convert the image to an OpenCV format (BGR)
```

```

56.     frame = grab_result.Array
57.     frame = cv2.cvtColor(frame, cv2.COLOR_BAYER_RG2RGB)
58.     frame = cv2.flip(frame, 0)
59.     frame = cv2.flip(frame, 1)
60.     cv2_im = frame
61.     cv2_im_rgb = cv2_im
62.     cv2_im_rgb = cv2.resize(cv2_im_rgb, inference_size)
63.     run_inference(interpreter, cv2_im_rgb.tobytes())
64.     objs = get_objects(interpreter, args.threshold)[:args.top_k]
65.     cv2_im = append_objs_to_img(cv2_im, inference_size, objs, labels)
66.     # Edited #
67.     width = int(cv2_im.shape[1] * scale_percent / 100)
68.     height = int(cv2_im.shape[0] * scale_percent / 100)
69.     dim = (width, height)
70.     # resize image
71.     cv2_im = cv2.resize(cv2_im, dim, interpolation=cv2.INTER_AREA)
72.     # Capture the latest frame and transform to image
73.     captured_image = Image.fromarray(cv2_im)
74.     # Convert captured image to photoimage
75.     photo_image = ImageTk.PhotoImage(image=captured_image)
76.     # position
77.     label_widget.place(x = 0, y = 0)
78.     # Displaying photoimage in the label
79.     label_widget.photo_image = photo_image
80.     # Configure image in the label
81.     label_widget.configure(image=photo_image)
82.
83. def write_terminal(x, y, z):
84.     # Create circle
85.     indicator_canvas.create_oval(0, 0, 160, 160, fill="red", outline='red')
86.     # Play a sound when image is captured
87.     winsound.PlaySound("defect_found.wav", winsound.SND_ASYNC)
88.     label_c1["text"] = "PCB defect detected"
89.     label_c1.configure(fg="red")
90.     label_c2["text"] = "Defect:"
91.     label_c2.configure(fg="black")
92.     label_c2.place(x = 790, y = 90)
93.     label_c3["text"] = x
94.     label_c3.configure(fg="red")
95.     label_c3.place(x = 790, y = 110)
96.     label_c4["text"] = y
97.     label_c4.configure(fg="red")
98.     label_c4.place(x = 790, y = 130)
99.     label_c5["text"] = z
100.    label_c5.configure(fg="red")
101.    label_c5.place(x = 790, y = 150)
102.
103. def append_objs_to_img(cv2_im, inference_size, objs, labels):
104.     height, width, channels = cv2_im.shape
105.     scale_x, scale_y = width / inference_size[0], height / inference_size[1]
106.     label_ =
107.     label_counter = [0, 0, 0]
108.
109.     for obj in objs:
110.         bbox = obj.bbox.scale(scale_x, scale_y)
111.         x0, y0 = int(bbox.xmin), int(bbox.ymin)
112.         x1, y1 = int(bbox xmax), int(bbox ymax)
113.         percent = int(100 * obj.score)
114.         label = '{}% {}'.format(percent, labels.get(obj.id, obj.id))
115.         label_ = label_ + label + '\n'
116.         if labels.get(obj.id, obj.id) == 'Opens':
117.             label_counter[0] = label_counter[0] + 1
118.         elif labels.get(obj.id, obj.id) == 'Short circuit':
119.             label_counter[1] = label_counter[1] + 1
120.         elif labels.get(obj.id, obj.id) == 'Toomuch':
121.             label_counter[2] = label_counter[2] + 1
122.         cv2_im = cv2.rectangle(cv2_im, (x0, y0), (x1, y1), (0, 255, 0), 2)
123.         cv2_im = cv2.putText(cv2_im, label, (x0, y0+30), cv2.FONT_HERSHEY_SIMPLEX, 1.0,
124.                             (255, 0, 0), 2)
124.         label_a_ = label_b_ = label_c_ =

```

```

125.     detect_status = 0
126.
127.     if label_counter[0] > 0:
128.         label_a_ = str("Opens: " + str(label_counter[0])))
129.         detect_status = 1
130.     if label_counter[1] > 0:
131.         label_b_ = str("Short circuit: " + str(label_counter[1])))
132.         detect_status = 1
133.     if label_counter[2] > 0:
134.         label_c_ = str("Toomuch: " + str(label_counter[2])))
135.         detect_status = 1
136.     if detect_status != 0:
137.         write_terminal(label_a_, label_b_, label_c_)
138.     return cv2_im
139.
140. # Create a GUI app
141. app = tkinter.Tk()
142. app.iconbitmap("logo.ico")
143. app.title("PCB Soldering Detection")
144. # percent of original size
145. scale_percent = 59.6
146. # save data terminal
147. terminal_temp = ''
148. # Create an instant camera object
149. camera = pylon.InstantCamera(pylon.TlFactory.GetInstance().CreateFirstDevice())
150. # Open the camera
151. camera.Open()
152. # ExposureTime # Bisa buat atur kecerahan
153. camera.ExposureTime.SetValue(50000) # 50000
154. # Start grabbing frames
155. camera.StartGrabbing(pylon.GrabStrategy_LatestImageOnly)
156. # Bind the app with Escape keyboard to
157. # quit app whenever pressed
158. app.bind('<Escape>', lambda e: app.quit())
159. app.geometry("300x300")
160. # Create a label and display it on app
161. label_widget = Label(app)
162. label_widget.pack()
163. canvas = tkinter.Canvas(app, width=761, height=572, bg="black")
164. canvas.place(x = 0, y = 0)
165. canvas2 = tkinter.Canvas(app, width=227, height=455, bg="grey")
166. canvas2.place(x = 778, y = 17)
167. # Create circle as indicator
168. # Make frame
169. indicator_frame = tkinter.Frame(app)
170. indicator_frame.place(x=2000, y=70)
171. # Make canvas
172. indicator_canvas = tkinter.Canvas(indicator_frame, bg='grey', width=162, height=162,
highlightthickness=0)
173. indicator_canvas.pack()
174. # Make notification
175. notif = tkinter.Label(app, text="", font=("Arial", 8), fg="green")
176. notif.place(x=2000, y=65)
177. # Label and Button Camera
178. label_a = tkinter.Label(app, text="Camera is close", font=("Helvetica", 24), fg="red",
highlightthickness=0)
179. label_a.place(x = 2000, y = 10)
180. label_b = tkinter.Label(app, text="Terminal:", font=("Helvetica", 20), fg="black",
highlightthickness=0, bg="grey")
181. label_b.place(x = 790, y = 25)
182. label_c1 = tkinter.Label(app, text="", font=("Helvetica", 12), fg="black",
highlightthickness=0, bg="grey")
183. label_c1.place(x = 790, y = 64)
184. label_c2 = tkinter.Label(app, text="", font=("Helvetica", 12), fg="black",
highlightthickness=0, bg="grey")
185. label_c2.place(x = 2000, y = 46)
186. label_c3 = tkinter.Label(app, text="", font=("Helvetica", 12), fg="black",
highlightthickness=0, bg="grey")
187. label_c3.place(x = 2000, y = 46)

```

```
188. label_c4 = tkinter.Label(app, text="", font=("Helvetica", 12), fg="black",
highlightthickness=0, bg="grey")
189. label_c4.place(x = 2000, y = 46)
190. label_c5 = tkinter.Label(app, text="", font=("Helvetica", 12), fg="black",
highlightthickness=0, bg="grey")
191. label_c5.place(x = 2000, y = 46)
192. button_1 = tkinter.Button(app, text="Capture Image", font=("Helvetica", 18) ,
command=toggle_text) #, width=20, height=5)
193. button_1.place(x = 804, y = 514)
194.
195. default_model_dir = ''
196. default_model = 'pcb_edgetpu_4Agus23.tflite'
197. default_labels = 'labels.txt'
198. parser = argparse.ArgumentParser()
199. parser.add_argument('--model', help='.tflite model path',
default=os.path.join(default_model_dir,default_model))
200. parser.add_argument('--labels', help='label file path',
default=os.path.join(default_model_dir, default_labels))
203. parser.add_argument('--top_k', type=int, default=30,
help='number of categories with highest score to display')
205. parser.add_argument('--camera_idx', type=int, help='Index of which video source to use. ',
default = camera)
206. parser.add_argument('--threshold', type=float, default=threshold_,
help='classifier score threshold')
208. args = parser.parse_args()
209.
210. interpreter = make_interpreter(args.model)
211. interpreter.allocate_tensors()
212. labels = read_label_file(args.labels)
213. inference_size = input_size(interpreter)
214.
215. if __name__ == '__main__':
216.     app.mainloop()
217.
```