A. <u>Coding</u>

Plastic Identification Coding:

i) Libraries downloading commands:

```
pip install numpy
pip install opencv-python
pip install tensorflow
pip install keras
```

ii) For Training:

```
import tensorflow as tf
import os, shutil
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras import optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_dir = 'D:/FYP THINGS/Images_FYP/training'
validation_dir = 'D:/FYP THINGS/Images_FYP/validation'

size=150
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(size,size,3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer=optimizers.RMSprop(lr=1e-4),metrics=['acc'])
# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
# This is the target directory
train_dir,
```

```
# All images will be resized to 150x150
target_size=(150, 150),
batch_size=20,
# Since we use binary_crossentropy loss, we need binary labels
class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
validation_dir,
target_size=(150, 150),
batch_size=20,
class_mode='binary')

history = model.fit_generator(train_generator,steps_per_epoch=int(1249/100),epochs=15,
                              validation_data=validation_generator,
validation_steps=int(451/50))
```

jupyter Training_FYP_Model Last Checkpoint: 10/17/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

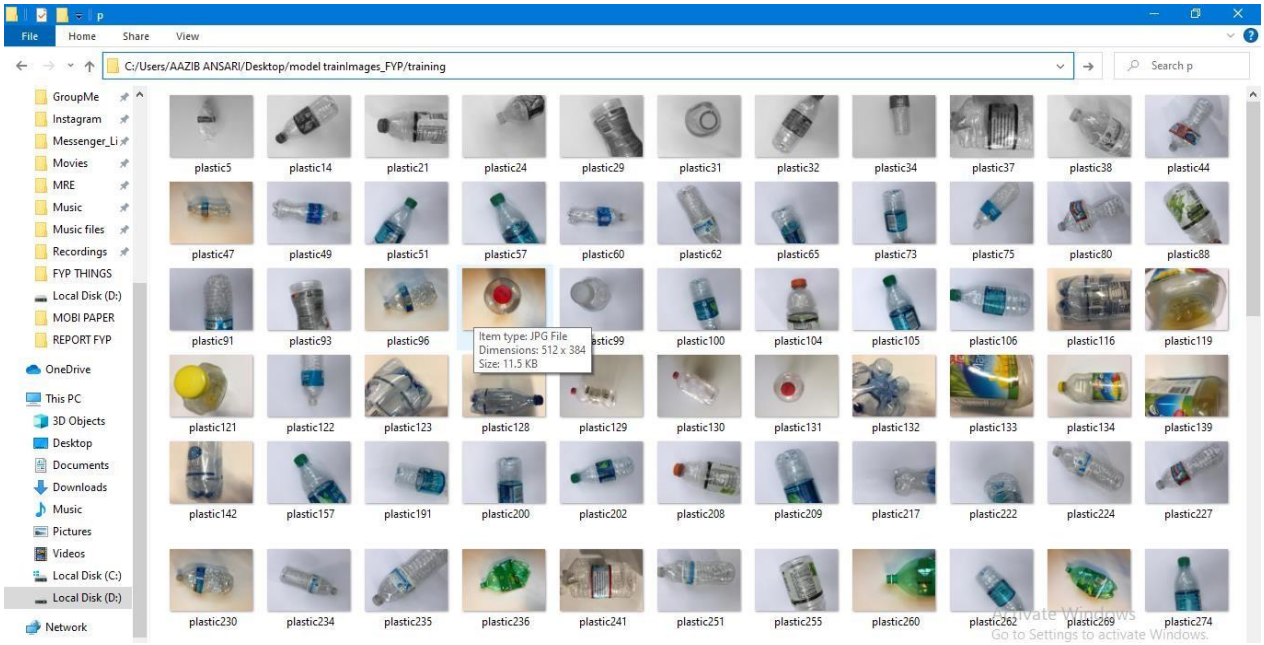
Run Code

```
Found 2999 images belonging to 2 classes.

In [6]: history = model.fit_generator(train_generator,steps_per_epoch=100,epochs=30,
        validation_data=validation_generator,
        validation_steps=50)

Epoch 1/15
100/100 [==============================] - 939s 9s/step - loss: 0.6589 - acc: 0.5940 - val_loss: 0.4835 - val_ac
c: 0.6810
Epoch 2/15
100/100 [==============================] - 945s 9s/step - loss: 0.6340 - acc: 0.6304 - val_loss: 0.6973 - val_ac
c: 0.6740
Epoch 3/15
100/100 [==============================] - 941s 9s/step - loss: 0.6197 - acc: 0.6410 - val_loss: 0.6873 - val_ac
c: 0.5826
Epoch 4/15
100/100 [==============================] - 1051s 11s/step - loss: 0.6090 - acc: 0.6665 - val_loss: 0.5343 - val_
acc: 0.6460
Epoch 5/15
100/100 [==============================] - 898s 9s/step - loss: 0.5882 - acc: 0.6840 - val_loss: 0.6095 - val_ac
c: 0.6230
Epoch 6/15
100/100 [==============================] - 910s 9s/step - loss: 0.6023 - acc: 0.6650 - val_loss: 0.7596 - val_ac
c: 0.6266
Epoch 7/15
100/100 [==============================] - 987s 10s/step - loss: 0.5837 - acc: 0.6886 - val_loss: 0.4353 - val_a
cc: 0.6710
Epoch 8/15
100/100 [==============================] - 1007s 10s/step - loss: 0.5663 - acc: 0.6975 - val_loss: 1.0324 - val_
acc: 0.5500
Epoch 9/15
100/100 [==============================] - 934s 9s/step - loss: 0.5622 - acc: 0.6925 - val_loss: 0.5040 - val_ac
c: 0.5816
Epoch 10/15
100/100 [==============================] - 916s 9s/step - loss: 0.5707 - acc: 0.7015 - val_loss: 0.6489 - val_ac
c: 0.6090
Epoch 11/15
100/100 [==============================] - 926s 9s/step - loss: 0.5613 - acc: 0.7026 - val_loss: 0.6759 - val_ac
c: 0.5800
```



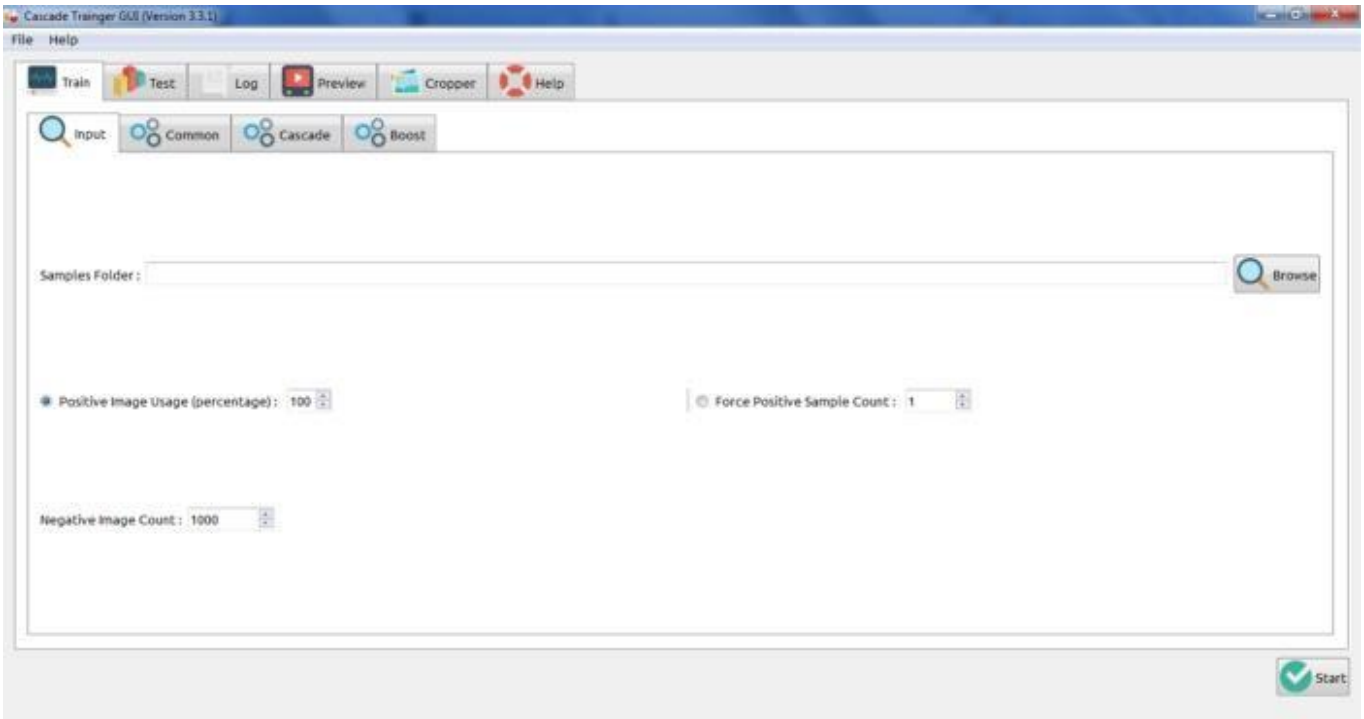
```
#Almost 90% Accuracy
# Model file save
model.save('FYP_Model1.h5')
```

Data:

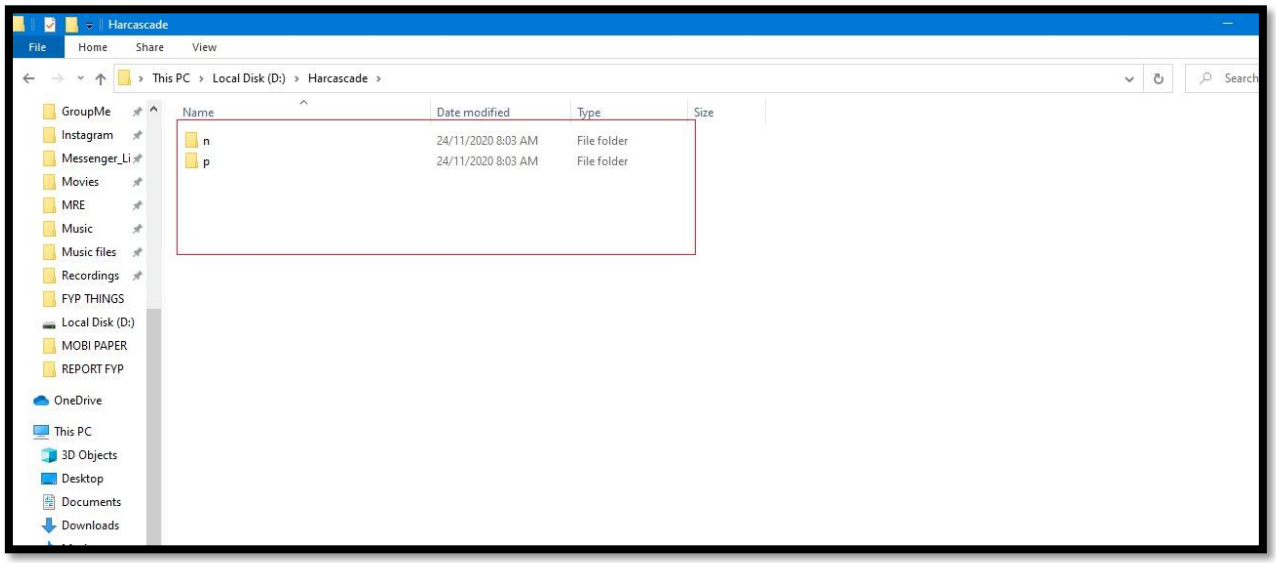


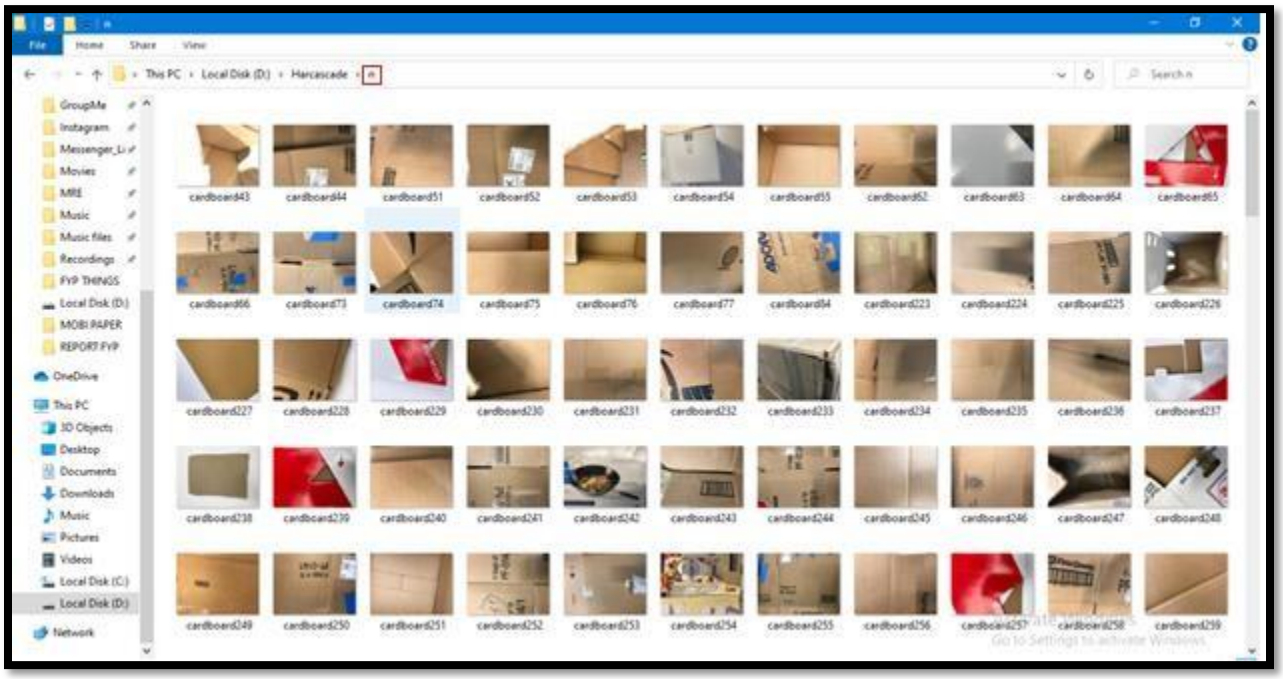
iii) To Train Haar Classifier from own dataset:

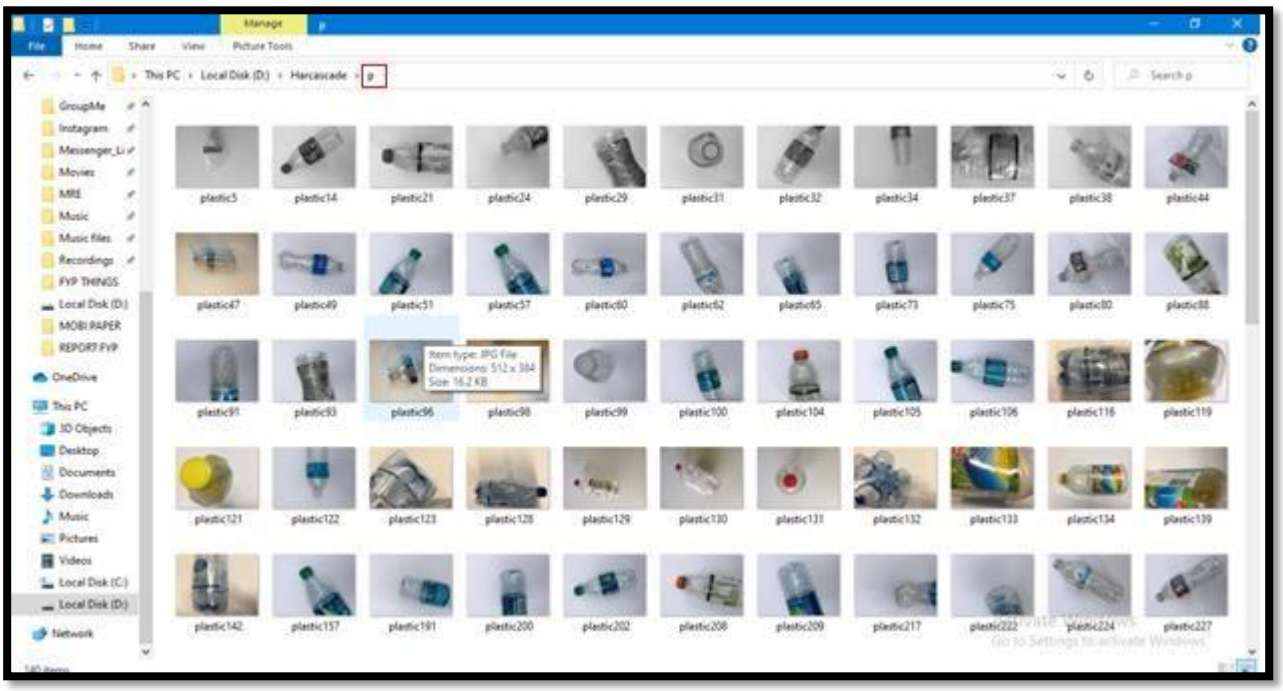
a) First download Cascade Trainer GUI



b) Create folder where create sub-folder names “n” for negative pictures and “p” for positive images.


Harcascade
File
Home
Share
View
This PC › Local Disk (D:) › Harcascade ›
Search
GroupMe
Instagram
Messenger_Li
Movies
MRE
Music
Music files
Recordings
FYP THINGS
Local Disk (D:)
MOBI PAPER
REPORT FYP
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Name
Date modified
Type
Size
n
24/11/2020 8:03 AM
File folder
p
24/11/2020 8:03 AM
File folder


File
Home
Share
View
This PC › Local Disk (D:) › Harcascade ›
Search n
GroupMe
Instagram
Messenger_Li
Movies
MRE
Music
Music files
Recordings
FYP THINGS
Local Disk (D:)
MOBI PAPER
REPORT FYP
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Network
cardboard43
cardboard44
cardboard51
cardboard52
cardboard53
cardboard54
cardboard55
cardboard62
cardboard63
cardboard64
cardboard65
cardboard66
cardboard73
cardboard74
cardboard75
cardboard76
cardboard77
cardboard84
cardboard223
cardboard224
cardboard225
cardboard226
cardboard227
cardboard228
cardboard229
cardboard230
cardboard231
cardboard232
cardboard233
cardboard234
cardboard235
cardboard236
cardboard237
cardboard238
cardboard239
cardboard240
cardboard241
cardboard242
cardboard243
cardboard244
cardboard245
cardboard246
cardboard247
cardboard248
cardboard249
cardboard250
cardboard251
cardboard252
cardboard253
cardboard254
cardboard255
cardboard256
cardboard257
cardboard258
cardboard259



c) Open the cascade trainer GUI paste the previously copied path into **sample folder** location also count the no of negative images you have and put it in negative image count.

d) After training we have .xml file which we will use in over last code where we detect Plastic.

iv) Deploy Training Dataset(Code):

```
import sys
import PyQt5
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import pymysql
from PyQt5 import QtGui, QtWidgets, QtCore
import numpy as np
import cv2
from gpiozero import Servo
from time import sleep
m=180
mr=m*(8.9/100)
f=m*(0.078/100)
wax=m*(0.12/100)
con=((m-mr)/(m))*100
amount=m/18
class Window(QDialog):
```

```
def __init__(self):
    super().__init__()

    self.title = "Plastic Detector"
    self.top = 600
    self.left = 600
    self.width = 800
    self.height = 600


    self.InitWindow()


def InitWindow(self):


    self.setWindowTitle(self.title)
    self.setGeometry(self.top, self.left, self.width, self.height)

    oImage = QImage("plastic2.jpg")
    sImage = oImage.scaled(QSize(780,430))                    # resize Image to widgets size
    palette = QPalette()
    palette.setBrush(QPalette.Window, QBrush(sImage))
    self.setPalette(palette)
    vbox = QVBoxLayout()

    self.button1 = QPushButton("Start Detection", self)
    self.button1.setStyleSheet('background:blue')
    self.button1.setFont(QtGui.QFont("Sanserif", 15))
    vbox.addWidget(self.button1)
    self.button1.clicked.connect(self.Detection)
    self.setLayout(vbox)
    self.show()

def Detection(self):

    self.Plastic_cascade=cv2.CascadeClassifier('Plastic_cascaded.xml')
    self.cap = cv2.VideoCapture(0)
    self.cap.set(3,640) #width=640
    self.cap.set(4,480) #height=480

    if self.cap.isOpened():
        _,self.frame = self.cap.read()
        self.cap.release() #releasing camera immediately after capturing picture
        if _ and self.frame is not None:
            cv2.imwrite('img1.jpg', self.frame)
    self.img=cv2.imread('img1.jpg')
```

```
        self.gray = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
        self.plastic=self.Plastic_cascade.detectMultiScale(self.gray,1.1,3)
        for (x, y, w, h) in self.plastic:
            cv2.rectangle(self.img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(self.img,'Plastic Bottle',(x,y),font,1,(255,0,0), 1, cv2.LINE_AA)


        if self.Plastic_cascade:
            cv2.imshow('(1) img', self.img)
            self.cams = Window1()
            self.cams.show()
            self.close()

        else:
            cv2.imshow('(0) img', self.img)
            self.cams = Window2()
            self.cams.show()
            self.close()

class Window1(QDialog):
    def __init__(self):
        super().__init__()

        self.title = "Plastic Identification"
        self.top = 500
        self.left = 500
        self.width = 800
        self.height = 400


        self.InitUi()

    def InitUi(self):


        self.setWindowTitle(self.title)
        self.setGeometry(self.top, self.left, self.width, self.height)

        oImage = QImage("plastic2.jpg")
        sImage = oImage.scaled(QSize(780,430))                # resize Image to widgets size
        palette = QPalette()
        palette.setBrush(QPalette.Window, QBrush(sImage))
        self.setPalette(palette)
        vbox = QVBoxLayout()
        self.Detect = QLabel("Following Element Was:")
        vbox.addWidget(self.Detect)
```

```
        #if cnn = 1:
        self.DetectBox = QLabel("This is Plastic Bottle(PET)")
        self.DetectBox.setAlignment(Qt.AlignCenter)
        self.DetectBox.setStyleSheet('background:green')
        self.DetectBox.setFont(QtGui.QFont("Sanserif", 20))
        #upon hardware coding, it will be change to read only
        vbox.addWidget(self.DetectBox)
        self.button1 = QPushButton("Calculate Weight", self)
        self.button1.setStyleSheet('background:blue')
        self.button1.setFont(QtGui.QFont("Sanserif", 15))
        vbox.addWidget(self.button1)
        self.button1.clicked.connect(self.Weight)
        self.setLayout(vbox)
        self.show()
    def Weight(self):
        self.cams = Window3()
        self.cams.show()
        self.close()
class Window2(QDialog):
    def __init__(self):
        super().__init__()

        self.title = "Plastic Identification"
        self.top = 600
        self.left = 600
        self.width = 800
        self.height = 600


        self.InitUi()

    def InitUi(self):


        self.setWindowTitle(self.title)
        self.setGeometry(self.top, self.left, self.width, self.height)

        oImage = QImage("plastic2.jpg")
        sImage = oImage.scaled(QSize(780,430))                # resize Image to widgets size
        palette = QPalette()
        palette.setBrush(QPalette.Window, QBrush(sImage))
        self.setPalette(palette)
        vbox = QVBoxLayout()
        self.Detect = QLabel("Following Element Was:")
        vbox.addWidget(self.Detect)
        #if cnn = 1:
        self.DetectBox = QLabel("This is not Plastic Bottle(PET)")
        self.DetectBox.setAlignment(Qt.AlignCenter)
```

```
        self.DetectBox.setStyleSheet('background:red')
        self.DetectBox.setFont(QtGui.QFont("Sanserif", 20))
        #upon hardware coding, it will be change to read only
        vbox.addWidget(self.DetectBox)
        self.button1 = QPushButton("Go To Main Menu", self)
        self.button1.setStyleSheet('background:green')
        self.button1.setFont(QtGui.QFont("Sanserif", 15))
        vbox.addWidget(self.button1)
        self.button1.clicked.connect(self.Withdraw1)
        self.setLayout(vbox)
        self.show()
    def Withdraw1(self):
        self.cams = Window()
        self.cams.show()
        self.close()
class Window3(QDialog):
    def __init__(self):
        super().__init__()

        self.title = "Weight of Bottles"
        self.top = 500
        self.left = 500
        self.width = 800
        self.height = 400


        self.InitUi1()

    def InitUi1(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.top, self.left, self.width, self.height)

        oImage = QImage("plastic2.jpg")
        sImage = oImage.scaled(QSize(780,430))                # resize Image to widgets size
        palette = QPalette()
        palette.setBrush(QPalette.Window, QBrush(sImage))
        self.setPalette(palette)
        vbox = QVBoxLayout()

        self.Weight = QLabel("Mass of Plastic is:")
        vbox.addWidget(self.Weight)


        #pi coding to obtain reading from weight sensor:
        self.WeightBox = QLineEdit(str(m)+ "gram")
        self.WeightBox.setStyleSheet('background:white')
        self.WeightBox.setFont(QtGui.QFont("Sanserif", 16))
```

```
        #upon hardware coding, it will be change to read only
        vbox.addWidget(self.WeightBox)

        self.button2 = QPushButton("Withdraw", self)
        self.button2.setStyleSheet('background:blue')
        self.button2.setFont(QtGui.QFont("Sanserif", 15))
        vbox.addWidget(self.button2)
        self.button2.clicked.connect(self.Withdraw)

        self.setLayout(vbox)

        self.show()

    def Withdraw(self):
        servo = 17
        myCorecction=0.45
        maxPW=(2.0+myCorecction)/1000
        minPW=(1.0-myCorecction)/1000
        myservo=Servo(servo,min_pulse_width=minPW,max_pulse_width=maxPW)

        myservo.mid()
        sleep(2)
        myservo.max()
        sleep(3)
        myservo.mid()
        sleep(3)

        self.cams = Window4()
        self.cams.show()
        self.close()
class Window4(QDialog):
    def __init__(self):
        super().__init__()

        self.title = "Withdraw Details"
        self.top = 400
        self.left = 400
        self.width = 800
        self.height = 400


        self.InitUi2()

    def InitUi2(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.top, self.left, self.width, self.height)
```

```
oImage = QImage("plastic2.jpg")
sImage = oImage.scaled(QSize(780,430))                # resize Image to widgets size
palette = QPalette()
palette.setBrush(QPalette.Window, QBrush(sImage))
self.setPalette(palette)
vbox = QVBoxLayout()
self.Weight = QLabel("Mass of Plastic is:")
vbox.addWidget(self.Weight)

#pi coding to obtain reading from weight sensor:
self.WeightBox = QLineEdit(str(m)+ "gram")
self.WeightBox.setStyleSheet('background:white')
self.WeightBox.setFont(QtGui.QFont("Sanserif", 16))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.WeightBox)
self.Money = QLabel("Amount from Mass:")
vbox.addWidget(self.Money)
#formula to calculate money:
self.MoneyBox = QLineEdit("Rs."+str(amount))
self.MoneyBox.setStyleSheet('background:white')
self.MoneyBox.setFont(QtGui.QFont("Sanserif", 15))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.MoneyBox)
self.Keyboard = QLabel("Please Open virtual Keyboard from Top Right side of Taskbar")
self.Keyboard.setFont(QtGui.QFont("Sanserif", 16))
vbox.addWidget(self.Keyboard)

self.Withdraw = QLabel("To Withdraw, Please Fill the Following:")
vbox.addWidget(self.Withdraw)

self.Name = QLineEdit(self)
self.Name.setPlaceholderText('Please Enter Your Name')
self.Name.setStyleSheet('background:white')
self.Name.setFont(QtGui.QFont("Sanserif", 15))

vbox.addWidget(self.Name)

self.Email = QLineEdit(self)
self.Email.setPlaceholderText('Please Enter Your Email')
self.Email.setFont(QtGui.QFont("Sanserif", 15))
self.Email.setStyleSheet('background:white')

vbox.addWidget(self.Email)

self.Phone = QLineEdit(self)
self.Phone.setPlaceholderText('Please Enter Your Easypaisa Number')
self.Phone.setFont(QtGui.QFont("Sanserif", 15))
```

```
        self.Phone.setStyleSheet('background:white')

        vbox.addWidget(self.Phone)

        self.button = QPushButton("Insert Data", self)
        self.button.setStyleSheet('background:blue')

        self.button.setFont(QtGui.QFont("Sanserif", 15))
        vbox.addWidget(self.button)
        self.button.clicked.connect(self.InsertData)

        self.setLayout(vbox)

        self.show()

    def InsertData(self):
        con =
pymysql.connect(host="localhost",user="HHUA",password="1234",database="exampledb")

        with con.cursor() as cursor:
            cursor.execute("INSERT INTO WithdrawDetailofPlasticFYP(WEIGHT, AMOUNT, NAME,
EMAIL, EASYPAISANUMBER)" "VALUES('%s','%s','%s','%s','%s')"
%(".join(self.WeightBox.text()),".join(self.MoneyBox.text()),".join(self.Name.text()),".join(self.Email
.text()),".join(self.Phone.text())))
            con.commit()
            con.close()
            QMessageBox.about(self,'Connection', 'Data Inserted Successfully')
            self.close()
        self.cams = Window5()
        self.cams.show()
        self.close()

class Window5(QDialog):
    def __init__(self):
        super().__init__()

        self.title = "Pyrolysis to Plastic Details"
        self.top = 400
        self.left = 400
        self.width = 800
        self.height = 400


        self.InitUi3()

    def InitUi3(self):
        self.setWindowTitle(self.title)
```

```
self.setGeometry(self.top, self.left, self.width, self.height)

oImage = QImage("plastic2.jpg")
sImage = oImage.scaled(QSize(780,430))                # resize Image to widgets size
palette = QPalette()
palette.setBrush(QPalette.Window, QBrush(sImage))
self.setPalette(palette)
vbox = QVBoxLayout()
self.Weight = QLabel("Conversion of Plastic:")
vbox.addWidget(self.Weight)

#pi coding to obtain reading from weight sensor:
self.WeightBox = QLineEdit(str(con)+"%")
self.WeightBox.setStyleSheet('background:white')
self.WeightBox.setFont(QtGui.QFont("Sanserif", 16))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.WeightBox)
self.Money = QLabel("Mass of fuel:")
vbox.addWidget(self.Money)
#formula to calculate money:
self.MoneyBox = QLineEdit(str(f)+ "gram")
self.MoneyBox.setStyleSheet('background:white')
self.MoneyBox.setFont(QtGui.QFont("Sanserif", 15))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.MoneyBox)
self.Money1 = QLabel("Mass of Wax:")
vbox.addWidget(self.Money1)
#formula to calculate money:
self.Money1Box = QLineEdit(str(wax)+ "gram")
self.Money1Box.setStyleSheet('background:white')
self.Money1Box.setFont(QtGui.QFont("Sanserif", 15))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.Money1Box)
self.Money2 = QLabel("Mass of Residue and other condensable gases")
vbox.addWidget(self.Money2)
#formula to calculate money:
self.Money2Box = QLineEdit(str(mr)+ "gram")
self.Money2Box.setStyleSheet('background:white')
self.Money2Box.setFont(QtGui.QFont("Sanserif", 15))
#upon hardware coding, it will be change to read only
vbox.addWidget(self.Money2Box)
self.button2 = QPushButton("Goto Main Menu", self)
self.button2.setStyleSheet('background:blue')
self.button2.setFont(QtGui.QFont("Sanserif", 15))
vbox.addWidget(self.button2)
self.button2.clicked.connect(self.gotoMainMenu)
```

```
        self.setLayout(vbox)

        self.show()

    def gotoMainMenu(self):
        self.cams = Window()
        self.cams.show()
        self.close()

if __name__ == '__main__':
    app=QApplication(sys.argv)
    ex=Window()
    sys.exit(app.exec_())
cv2.waitKey(0)&0xFF
```

PLC Ladder Diagram: