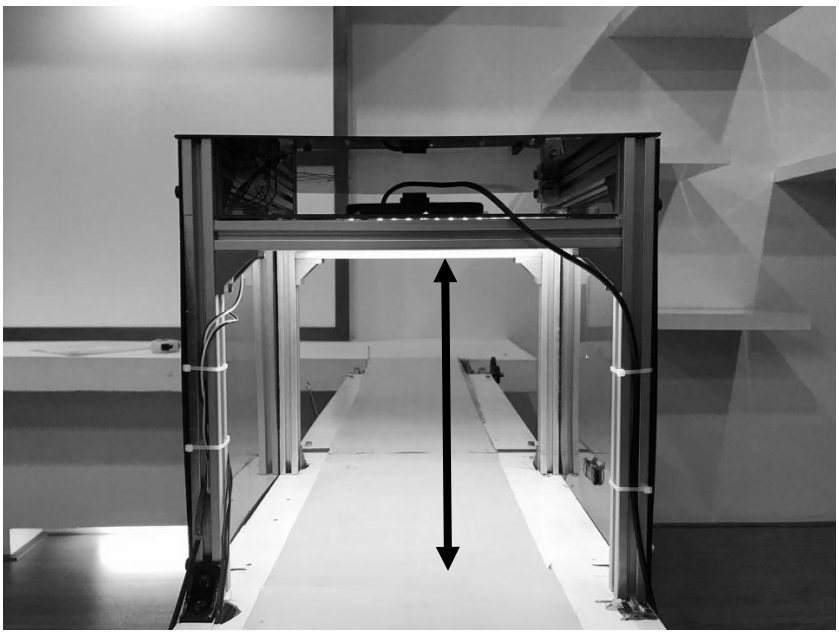
ภาคผนวก

ภาคผนวก ก

อุปกรณ์ระบบคัดแยกสับปะรดด้วยสีผิวของเปลือก

ภาพที่ ก.1 เครื่องคัดแยกสับปะรดด้วยสีผิวของเปลือก

ภาพที่ ก.2 หน้าจอแสดงผลของระบบคัดแยกสับปะรดด้วยสีผิวของเปลือก



ภาพที่ ก.3 การติดตั้งกล้องในระยะความห่างที่เหมาะสม

ภาพที่ ก.4 รูปภาพที่ได้รับผ่านกล้องของระบบคัดแยกสับปะรดด้วยสีของเปลือก

ภาคผนวก ข

โค้ดโปรแกรมระบบคัดแยกระดับของสับปะรดด้วยสีผิวของเปลือก เทรนโมเดล

```
#********************************** Import Libraly ******************************#
import tensorflow as tf
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Conv2D,MaxPooling2D,Dense,GlobalAveragePooling2D,Dropout,Flatten
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
#******************************** Dataset to Path*******************************#
dataset_path = "./โฟล์เดอร์สับปะรด5ระดับ/"
data_gen = ImageDataGenerator(rescale=1./255,
                              rotation_range=40,
                              width_shift_range=0.2,
                              height_shift_range=0.2,
                              shear_range=0.2,
                              zoom_range=0.2,
                              vertical_flip=True,
                              horizontal_flip=True,
                              fill_mode='nearest',
                              validation_split=0.2)
train_generator = data_gen.flow_from_directory(dataset_path ,
                                               subset='training',
                                               target_size=(128,128),
                                               batch_size=32,
                                               color_mode="rgb",
                                               shuffle=True,
                                               class_mode="categorical")
valid_generator = data_gen.flow_from_directory(dataset_path ,
                                               subset='validation',
                                               target_size=(128,128),
                                               batch_size=32,
                                               color_mode="rgb",
```

```
                                    shuffle=True,
                                    class_mode="categorical")
#************************************** Build Model*****************************#
model = Sequential()
# C1 Convolutional Layer
model.add(Conv2D(20, kernel_size=(3, 3),  activation='relu', input_shape=(128,128,3)))
# S2 Pooling Layer
model.add(MaxPooling2D(pool_size=(2, 2)))
# C3 Convolutional Layer
model.add(Conv2D(40, kernel_size=(3, 3),  activation='relu'))
# S4 Pooling Layer
model.add(MaxPooling2D(pool_size=(2, 2)))
#Flatten the CNN output so that we can connect it with fully connected layers
model.add(Flatten())
# FC6 Fully Connected Layer
model.add(Dense(100, activation='relu',name="layer1"))
model.add(Dropout(0.5))
model.add(Dense(50, activation='relu',name="layer2"))
model.add(Dropout(0.2))
#Output Layer with softmax activation
model.add(Dense(5, activation='softmax'))

model.compile(loss="categorical_crossentropy", optimizer='adam' ,metrics=["accuracy"])
model.summary()

#***************************Train Model and Save*******************************#
history = model.fit(train_generator,validation_data=valid_generator,epochs=500,verbose=1)
_,acc=model.evaluate_generator(valid_generator,verbose=1)
print('>%.3f'%(acc*100.0))
model.save('ชื่อโฟล์เดอร์\ชื่อโมเดล.model')


#************************************ convert model*****************************#
```

```
converter = tf.lite.TFLiteConverter.from_saved_model('ที่อยู่เซฟโมเดล')

tflite_model = converter.convert()

open("ที่อยู่เซฟโมเดลที่คอนเวิสแล้ว", "wb").write(tflite_model)
```

ภาคผนวก ค

โค้ดโปรแกรมระบบคัดแยกระดับของสับปะรดด้วยสีผิวของเปลือกตัวเครื่อง

```
import time #OLED
import Adafruit_GPIO.SPI as SPI#OLED
import Adafruit_SSD1306 #OLED
from PIL import Image#OLED
from PIL import ImageDraw#OLED
from PIL import ImageFont#OLED
# Raspberry Pi pin configuration:
RST = 24
# Note the following are only used with SPI:
DC = 23
SPI_PORT = 0
SPI_DEVICE = 0
# 128x32 display with hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)
# Initialize library.
disp.begin()
# Clear display.
disp.clear()
disp.display()
#begin
image = Image.open('/home/pi/cnnpine/wellcome.png').convert('1')
disp.image(image)
disp.display()


import tensorflow as tf #Tensorflow
import matplotlib.pyplot as plt #image
import numpy as np #image to array
import cv2 #video and image
import RPi.GPIO as GPIO#gpio

GPIO.setmode(GPIO.BCM)
GPIO.setup(14,GPIO.IN)#ส้มไฟ แดงสัญญาน น้ำตาลกราว
```

```
en = 12
moter = 16

GPIO.setup(moter,GPIO.OUT)
GPIO.setup(en,GPIO.OUT)

GPIO.output(moter,GPIO.LOW)
GPIO.PWM(en,150)

#begin
image = Image.open('/home/pi/cnnpine/loaddone.png').convert('1')
disp.image(image)
disp.display()



cam = cv2.VideoCapture(0)
cv2.namedWindow("pineapple")
# Load TFLite model and allocate tensors.
print("Load TFLite model and allocate tensors")
interpreter =
tf.lite.Interpreter(model_path='/home/pi/cnnpine/converted_modelpine5cDataGen5_10_63
-1.tflite')
interpreter.allocate_tensors()
# Get input and output tensors.
print("Get input and output tensors.")
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Test the model on input data.
print("Test the model on input data.")
input_shape = input_details[0]['shape']
print(input_shape)

#begin
image = Image.open('/home/pi/cnnpine/cameramodel.png').convert('1')
disp.image(image)
```

```
disp.display()
while True:
    ret, frame = cam.read()
    if not ret:
        print("failed to grab frame")
        #displaytext
        width = disp.width
        height = disp.height
        image = Image.new('1', (width, height))
        draw = ImageDraw.Draw(image)
        draw.rectangle((0,0,width,height), outline=0, fill=0)
        padding = -2
        top = padding
        bottom = height-padding
        x = 0
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()
        draw.text((x, top),    "        Error :         " ,  font=font, fill=255)
        draw.text((x, top+8), " failed to grab frame ", font=font, fill=255)
        draw.text((x, top+16),"Pleses restart machin ",  font=font, fill=255)
        draw.text((x, top+25),"   Pineapple project  ",  font=font, fill=255)
        # Display image.
        disp.image(image)
        disp.display()
        break
    cv2.imshow("pineapple", frame)

    s = False
    #begin
    image = Image.open('/home/pi/cnnpine/machinworking.png').convert('1')
    disp.image(image)
    disp.display()
    GPIO.output(moter,GPIO.HIGH)
    GPIO.output(en,GPIO.HIGH)
    GPIO.PWM(en,100)
```

```
k = cv2.waitKey(1)
if k%256 == 27:
    # ESC pressed
    print("Escape hit, closing...")
    #displaytext
    width = disp.width
    height = disp.height
    image = Image.new('1', (width, height))
    draw = ImageDraw.Draw(image)
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    padding = -2
    top = padding
    bottom = height-padding
    x = 0
    draw = ImageDraw.Draw(image)
    font = ImageFont.load_default()
    draw.text((x, top),    "       Cotrol Admin :        " ,  font=font, fill=255)
    draw.text((x, top+8), " Escape hit, closing... ", font=font, fill=255)
    draw.text((x, top+16),"Pleses restart machin ",  font=font, fill=255)
    draw.text((x, top+25),"   Pineapple project  ",  font=font, fill=255)
    # Display image.
    disp.image(image)
    disp.display()
    break
elif (k%256 == 32 or (GPIO.input(14) == False)):
    if s == False:
        s = True
        # SPACE pressed
        #begin
        image = Image.open('/home/pi/cnnpine/detect.png').convert('1')
        disp.image(image)
        disp.display()
        GPIO.output(en,GPIO.LOW)
        img_name = "/home/pi/cnnpine/pineapple.jpg"
        cv2.imwrite(img_name,frame)
```

```
print("{} written!".format(img_name))
# Load and transform image
print("Load and transform image.")
image_a = plt.imread('/home/pi/cnnpine/pineapple.jpg')
image_a = cv2.resize(image_a,(128,128))
image_a = np.asarray(image_a)/255
image_a = np.reshape(image_a,(1,128,128,3))
# Use same image as Keras model
print("Use same image as Keras model")
input_data = np.array(image_a, dtype=np.float32)
interpreter.set_tensor(input_details[0]['index'], input_data)

interpreter.invoke()

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.
print("Anser")
tf_results = interpreter.get_tensor(output_details[0]['index'])
output_data = np.array(tf_results)
CATEGORIES = ["level1", "level2","level3","level4","level5"]
np.nanmax(output_data[0])
print(output_data[0])
result = np.where(output_data[0] == np.amax(output_data[0]))
print(result)
print(CATEGORIES[int(result[0])])
if CATEGORIES[int(result[0])] == "level1":
        print("level1") #mediam
        #begin
        image = Image.open('/home/pi/cnnpine/lv1.png').convert('1')
        disp.image(image)
        disp.display()
        time.sleep(3)
elif CATEGORIES[int(result[0])] == "level2":
        print("level2") #mediam
        #begin
        image = Image.open('/home/pi/cnnpine/lv2.png').convert('1')
```

```
                disp.image(image)
                disp.display()
                time.sleep(3)
            elif CATEGORIES[int(result[0])] == "level3":
                print("level3") #mediam
                #begin
                image = Image.open('/home/pi/cnnpine/lv3.png').convert('1')
                disp.image(image)
                disp.display()
                time.sleep(3)
            elif CATEGORIES[int(result[0])] == "level4":
                print("level4") #mediam
                 #begin
                image = Image.open('/home/pi/cnnpine/lv4.png').convert('1')
                disp.image(image)
                disp.display()
                time.sleep(3)
            elif CATEGORIES[int(result[0])] == "level5":
                print("level5") #mediam
                #begin
                image = Image.open('/home/pi/cnnpine/lv5.png').convert('1')
                disp.image(image)
                disp.display()
                time.sleep(3)
            print("Done")
            #begin
            image = Image.open('/home/pi/cnnpine/done.png').convert('1')
            disp.image(image)
            disp.display()
        else :
            s = False


cam.release()
GPIO.output(moter,GPIO.LOW)
GPIO.output(en,GPIO.LOW)
```

```
disp.clear()
cv2.destroyAllWindows()
```

ภาคผนวก ง

โค้ดโปรแกรมทดลอง 10 fod cross validation

```
import tensorflow as tf
import os
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import  Conv2D,MaxPooling2D,Dense,GlobalAveragePooling2D,MaxPool2D,Dropout,Flat
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
%matplotlib inline

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import backend as K
```

```
image_dir = "./Dataset10flod/"
train_data = pd.read_csv('./dataset10-11-63/pineapple100set.csv')
Y = train_data[['label']]
```

```
data_gen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=30,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    shear_range=0.2,
                                    zoom_range=0.2,
                                    vertical_flip=True,
                                    horizontal_flip=True,
                                    fill_mode='nearest'
                              #,validation_split=0.2
                               )
```

```
# Define per-fold score containers
acc_per_fold = []
loss_per_fold = []
# Define the K-fold Cross Validator
kfold = KFold(n_splits=10 ,shuffle = True)
```

```
# K-fold Cross Validation model evaluation
fold_no = 1
for train_index,test_index in kfold.split(Y):
    training_data = train_data.iloc[train_index]
    validation_data = train_data.iloc[test_index]


    train_generator = data_gen.flow_from_dataframe(training_data, directory= image_dir,
                                                x_col="filename",
                                                y_col="label",
                                                target_size=(128,128),
                                                      color_mode="rgb",
                                                      shuffle=True,
                                                      class_mode="categorical")
    valid_generator = data_gen.flow_from_dataframe(validation_data, directory= image_dir,
                                                x_col="filename",
                                                y_col="label",
                                                target_size=(128,128),
                                                      color_mode="rgb",
                                                      shuffle=True,
                                                      class_mode="categorical")
    print("fold ",fold_no)
    print( training_data,"have ",len(training_data))
    print( validation_data,"have ",len(validation_data))
    print("------------------------------------------------------")
    fold_no += 1
```

```
# K-fold Cross Validation model evaluation
fold_no = 1
for train_index,test_index in kfold.split(Y):
    training_data = train_data.iloc[train_index]
    validation_data = train_data.iloc[test_index]


    train_generator = data_gen.flow_from_dataframe(training_data, directory= image_dir,
                                                   x_col="filename",
                                                   y_col="label",
                                                   target_size=(128,128),
                                                         color_mode="rgb",
                                                         shuffle=True,
                                                         class_mode="categorical")
    valid_generator = data_gen.flow_from_dataframe(validation_data, directory= image_dir,
                                                   x_col="filename",
                                                   y_col="label",
                                                   target_size=(128,128),
                                                         color_mode="rgb",
                                                         shuffle=True,
                                                         class_mode="categorical")
    # initialize the model
    print("[INFO] compiling model...")
        #Instantiate an empty model
    model = Sequential()
        # C1 Convolutional Layer
    model.add(Conv2D(20, kernel_size=(5, 5), strides=(1, 1), activation='relu', input_shape=(128,128,3), pad
        # S2 Pooling Layer
    model.add(MaxPool2D(pool_size=(2, 2), strides=(1, 1), padding='valid'))
        # C3 Convolutional Layer
    model.add(Conv2D(40, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='valid'))
        # S4 Pooling Layer
    model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
        #Flatten the CNN output so that we can connect it with fully connected layers
    model.add(Flatten())
        # FC6 Fully Connected Layer
    model.add(Dense(100, activation='relu'))
    model.add(Dense(50, activation='relu'))
        #Output Layer with softmax activation
    model.add(Dense(5, activation='softmax'))
    opt = Adam(lr=1e-3, decay=1e-3 / 5)
    model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=["accuracy"])
        # Generate a print
    print('------------------------------------------------------------------------')
    print(f'Training for fold {fold_no} ...')
          # Fit data to model
    history = model.fit(train_generator,validation_data=valid_generator,epochs=50,verbose=1)

        # Generate generalization metrics
    scores = model.evaluate(valid_generator,verbose=1)

    print(f'Score for fold {fold_no}: {model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of {
    acc_per_fold.append(scores[1] * 100)
    loss_per_fold.append(scores[0])
    K.clear_session()

        # Increase fold number
    fold_no = fold_no + 1
```

```
# == Provide average scores ==
print('------------------------------------------------------------------------')
print('Score per fold')
for i in range(0, len(acc_per_fold)):
    print('------------------------------------------------------------------------')
    print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy: {acc_per_fold[i]}%')
print('------------------------------------------------------------------------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(acc_per_fold)} (+- {np.std(acc_per_fold)})')
print(f'> Loss: {np.mean(loss_per_fold)}')
print('------------------------------------------------------------------------')
```