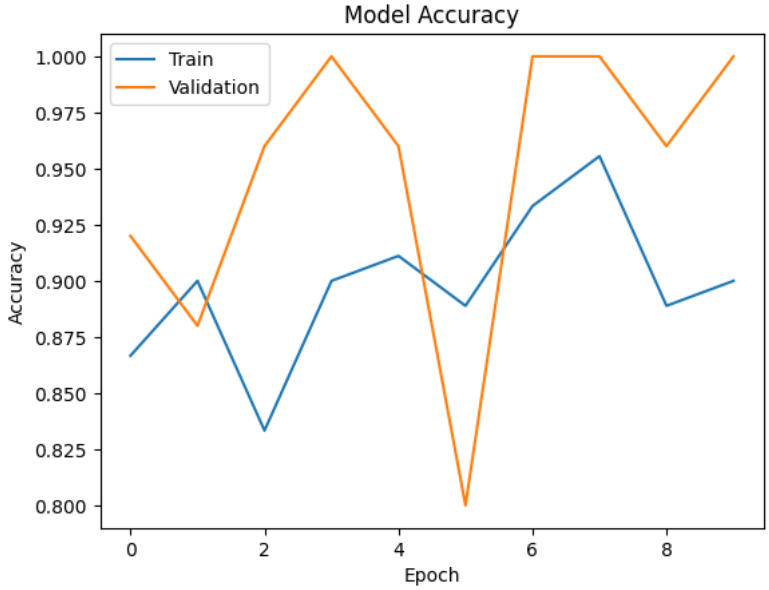
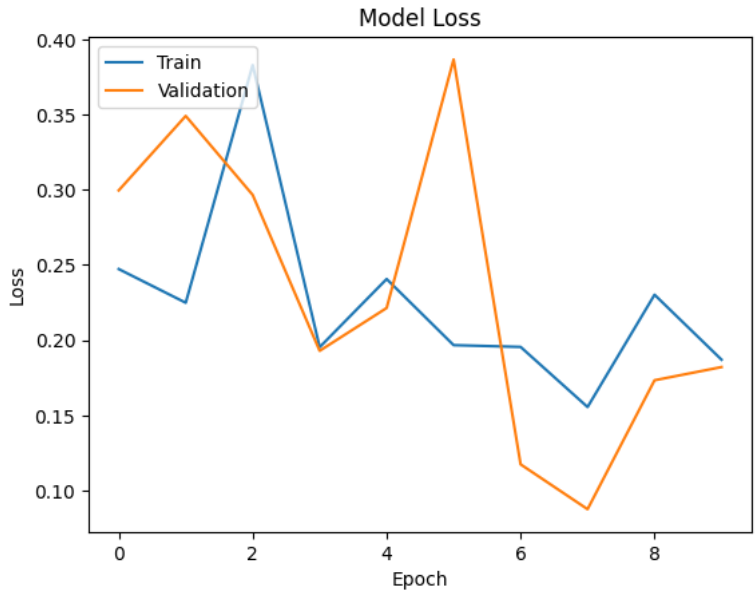
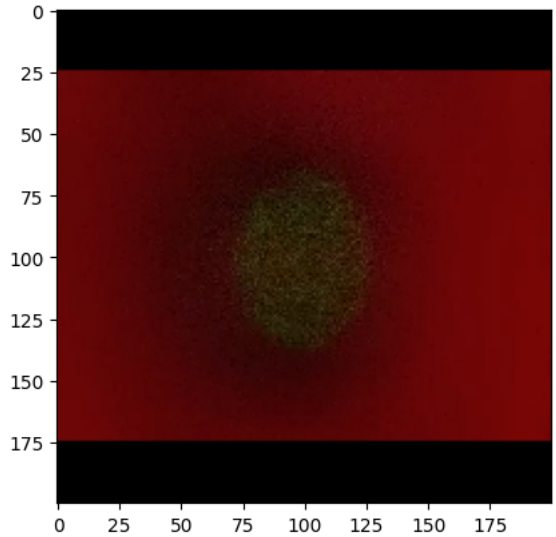
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os

# Mount Google Drive to access the dataset
from google.colab import drive
drive.mount('/content/drive')

# Define a function for green color detection
def detect_green_color(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_green = np.array([60, 50, 50])  # Adjust the values as needed
    upper_green = np.array([80, 255, 255])  # Adjust the values as needed
    green_mask = cv2.inRange(hsv_img, lower_green, upper_green)
    contours, _ = cv2.findContours(green_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if len(contours) > 0:
        return True
    else:
        return False

# Define the image data generators
train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)

# Load the training and validation datasets
train_dataset = train.flow_from_directory('/content/drive/MyDrive/Computer Vision/Base Data/Training',
                                          target_size=(200, 200),
                                          batch_size=3,
                                          class_mode='binary')

validation_dataset = validation.flow_from_directory('/content/drive/MyDrive/Computer Vision/Base Data/Testing',
                                                    target_size=(200, 200),
                                                    batch_size=3,
                                                    class_mode='binary')

# Create the CNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['accuracy'])

# Train the model
history = model.fit(train_dataset,
                    validation_data=validation_dataset,
                    steps_per_epoch=30,
                    epochs=10)

# Plot the accuracy and loss curves during training
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
# Specify the directory path containing the LED images
dir_path = '/content/drive/MyDrive/Data/intensity5'
```

```
    for i in os.listdir(dir_path):
        img = image.load_img(os.path.join(dir_path, i), target_size=(200, 200))
        plt.imshow(img)
        plt.show()

        # Preprocess the image for LED classification
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0

        # Predict the class using the trained model
        prediction = model.predict(img_array)
        if prediction[0] >= 0.5:
            led_label = "Led On"
        else:
            led_label = "Led Off"

        # Perform green color detection
        green_detected = detect_green_color(np.array(img))
        if green_detected:
            print("Green color detected!")
            # Perform further processing on the green region, e.g., find ROI, etc.

        print(led_label)
```

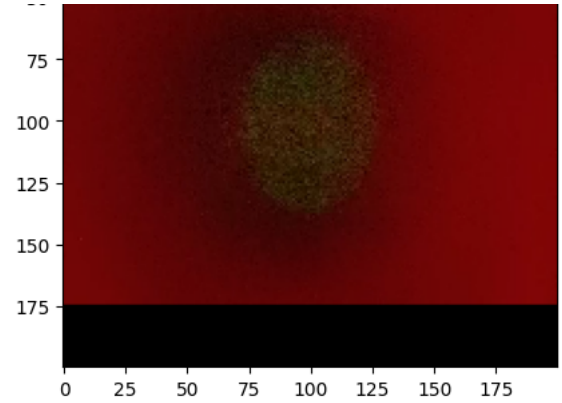
```
30/30 [==============================] - 18s 581ms/step - loss: 0.1556 - accuracy: 0.9556 - val_loss: 0.0876 - val_accuracy: 1.0
Epoch 9/10
30/30 [==============================] - 17s 577ms/step - loss: 0.2302 - accuracy: 0.8889 - val_loss: 0.1734 - val_accuracy: 0.9
Epoch 10/10
30/30 [==============================] - 18s 606ms/step - loss: 0.1871 - accuracy: 0.9000 - val_loss: 0.1821 - val_accuracy: 1.0
```



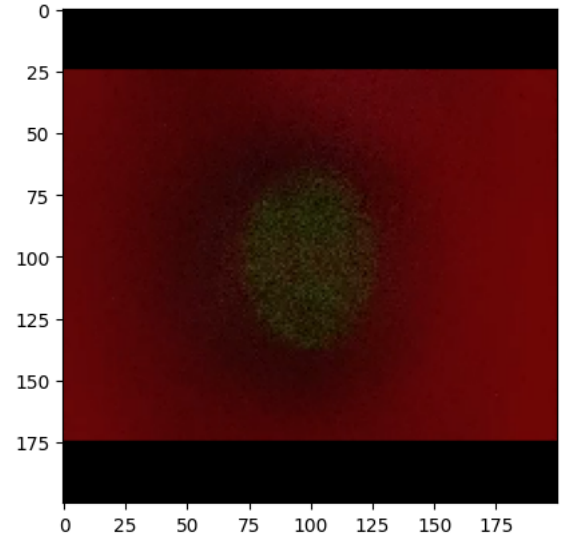




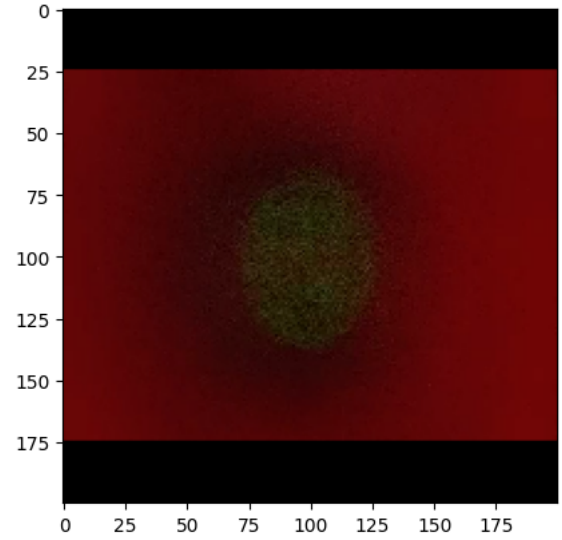

```
1/1 [==============================] - 0s 154ms/step
Led On
```



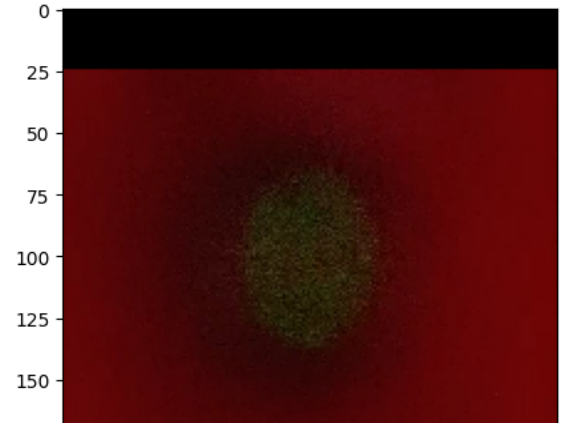

```
1/1 [==============================] - 0s 38ms/step
Led On
```

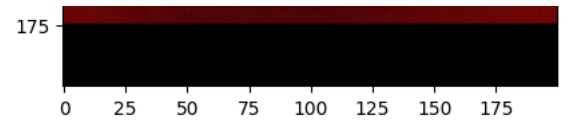



```
1/1 [==============================] - 0s 38ms/step
Led On
```

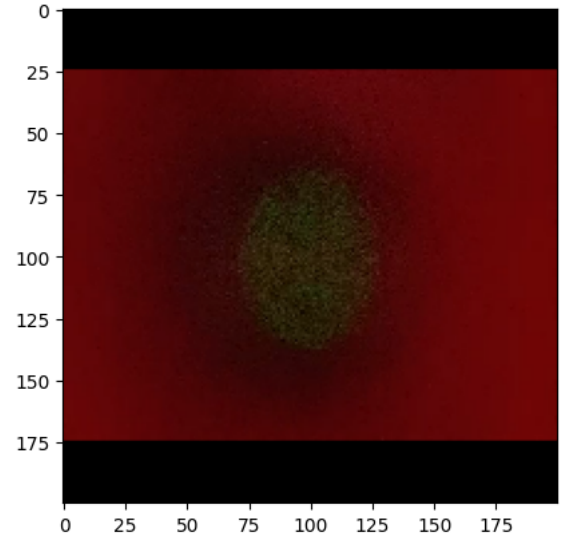



```
1/1 [==============================] - 0s 36ms/step
Led On
```

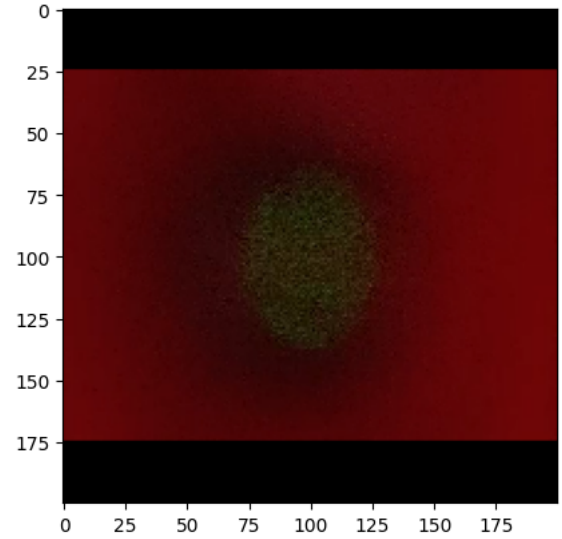




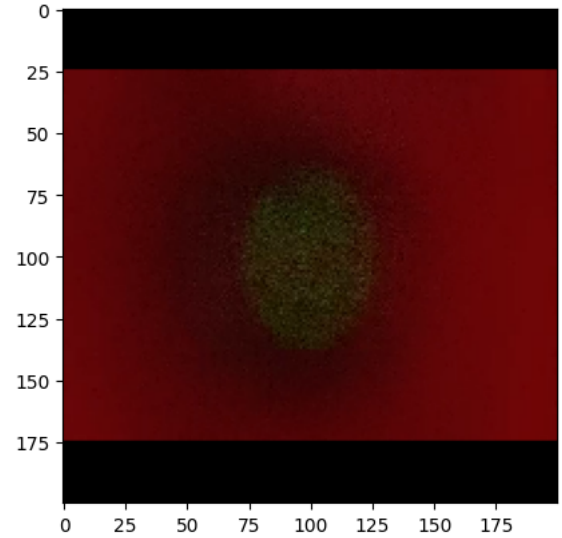

```
1/1 [==============================] - 0s 36ms/step
Led On
```



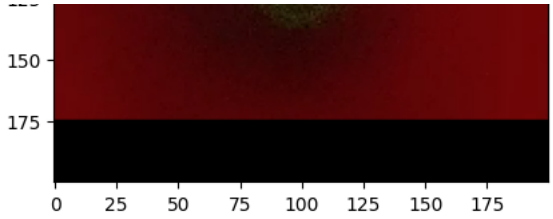

```
1/1 [==============================] - 0s 35ms/step
Led On
```



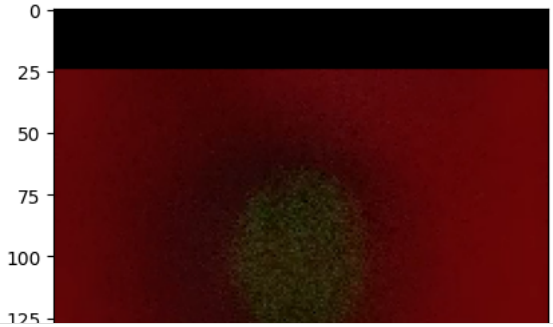

```
1/1 [==============================] - 0s 38ms/step
Led On
```




```
1/1 [==============================] - 0s 36ms/step
Led On
```




```
1/1 [==============================] - 0s 36ms/step
Led On
```



ducts - Cancel contracts here

✓ 4s completed at 11:28