Q5

In [1]:

```
import numpy as np
import os
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import keras
import tensorflow as tf
from random import randint
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras import layers
from keras import models
```

In [2]:

```
reverselookup = dict()
lookup = dict()
count = 0

for i in os.listdir(r'C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 08\leapGestRe
    if not i.startswith('.'):
        gesture_name = i[3:]
        lookup[gesture_name] = count
        reverselookup[count] = gesture_name
        count = count + 1

lookup
```

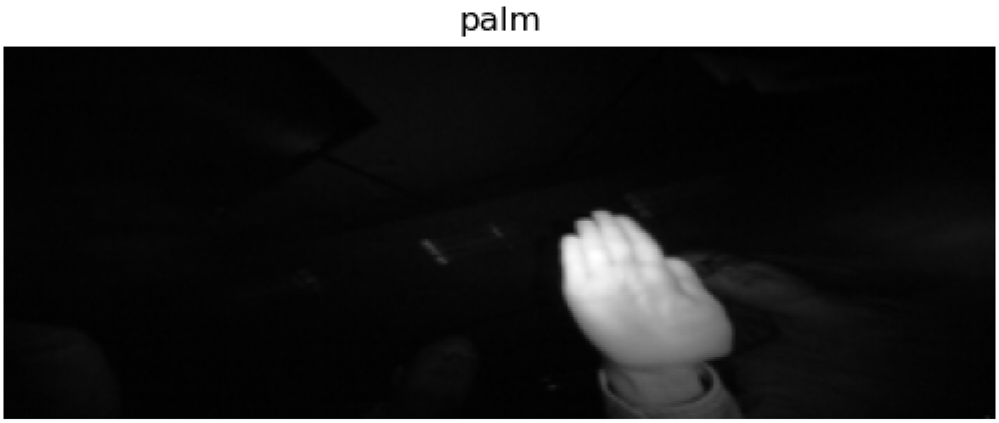
Out[2]:

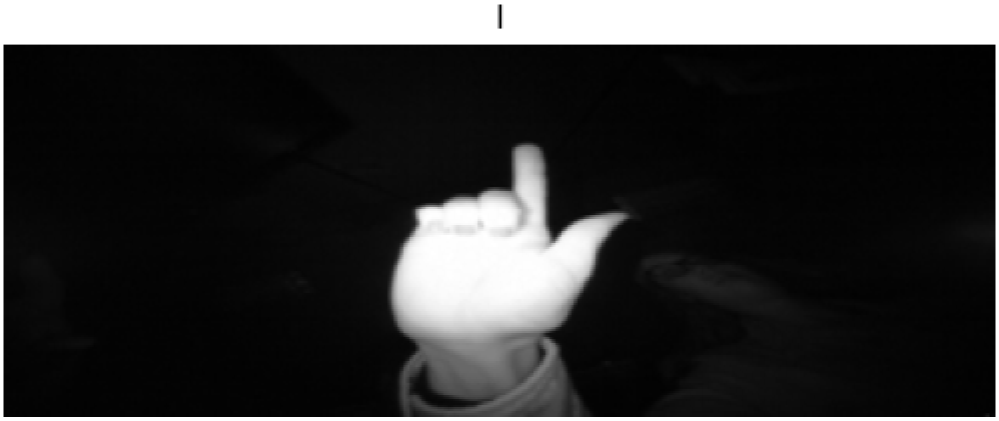
```
{'palm': 0,
 'l': 1,
 'fist': 2,
 'fist_moved': 3,
 'thumb': 4,
 'index': 5,
 'ok': 6,
 'palm_moved': 7,
 'c': 8,
 'down': 9}
```

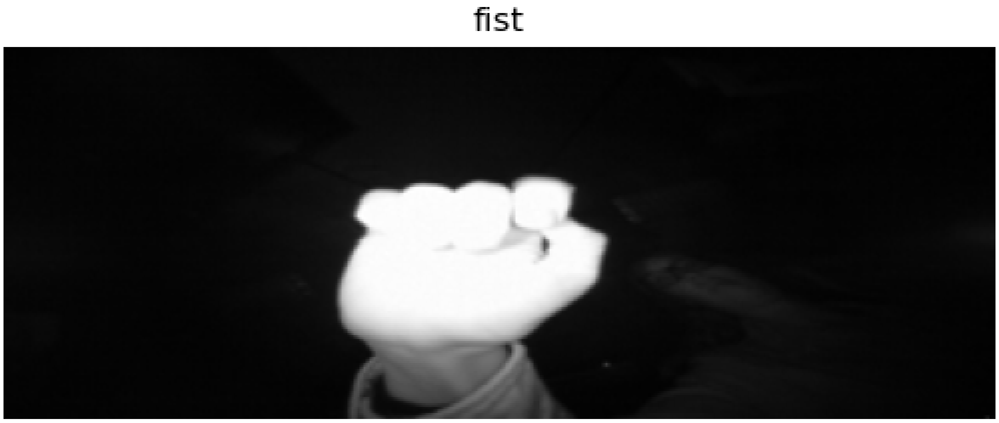
In [3]:

```
x_data = []
y_data = []
datacount = 0

for i in range(0, 10):
    for j in os.listdir(r'C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 08\leapGes

        if not j.startswith('.'):
            count = 0
            for k in os.listdir(r'C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 08

                img = Image.open(r'C:\Users\ABC\Desktop\BAI\BAI-S7\CV Lab\Lab 0
                img = img.resize((320, 120))
                arr = np.array(img)
                x_data.append(arr)
                count = count + 1
            gesture_name = j[3:]
            y_values = np.full((count, 1), lookup[gesture_name])
            y_data.append(y_values)
            datacount = datacount + count

x_data = np.array(x_data, dtype = 'float32')
y_data = np.array(y_data)
y_data = y_data.reshape(datacount, 1)
```

In [4]:
```
for i in range(0, 10):
    plt.imshow(x_data[i*200 , :, :], cmap="gray")
    plt.title(reverselookup[y_data[i*200 ,0]])
    plt.axis('off')
    plt.show()
```







fist_moved

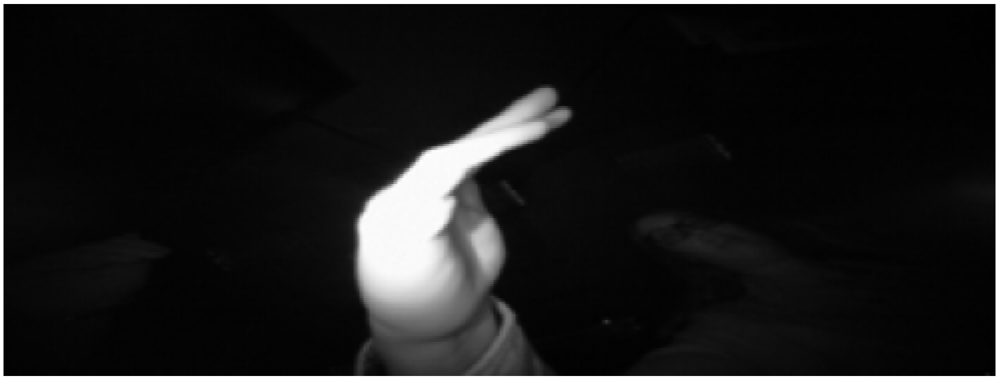
thumb

index

ok

palm_moved

c



down

In [5]:
```
y_data = to_categorical(y_data)
```

In [6]:
```
x_data = x_data.reshape((datacount, 120, 320, 1))
x_data /= 255
```

In [7]:
```
x_train, x_test_val, y_train, y_test_val = train_test_split(x_data, y_data, tes
x_validate, x_test, y_validate, y_test = train_test_split(x_test_val, y_test_va
```

In [8]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), strides = (2, 2), activation = 'relu', inpu
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(64, (3, 3), activation = 'relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation = 'relu'))
model.add(layers.Dense(10, activation = 'softmax'))
```

In [9]:

```
model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics
model.fit(x_train, y_train, epochs = 10, batch_size = 32, verbose = 1, validati
```

```
Epoch 1/10
500/500 [==============================] - 219s 404ms/step - loss: 0.2158 - a
ccuracy: 0.9314 - val_loss: 0.0026 - val_accuracy: 1.0000
Epoch 2/10
500/500 [==============================] - 176s 351ms/step - loss: 0.0121 - a
ccuracy: 0.9976 - val_loss: 0.0065 - val_accuracy: 0.9990
Epoch 3/10
500/500 [==============================] - 172s 345ms/step - loss: 0.0056 - a
ccuracy: 0.9990 - val_loss: 0.0035 - val_accuracy: 0.9990
Epoch 4/10
500/500 [==============================] - 176s 353ms/step - loss: 0.0022 - a
ccuracy: 0.9993 - val_loss: 6.2701e-05 - val_accuracy: 1.0000
Epoch 5/10
500/500 [==============================] - 190s 380ms/step - loss: 2.4270e-06
- accuracy: 1.0000 - val_loss: 5.6018e-06 - val_accuracy: 1.0000
Epoch 6/10
500/500 [==============================] - 204s 408ms/step - loss: 5.2095e-07
- accuracy: 1.0000 - val_loss: 5.8764e-06 - val_accuracy: 1.0000
Epoch 7/10
500/500 [==============================] - 194s 388ms/step - loss: 3.7250e-07
- accuracy: 1.0000 - val_loss: 5.2844e-06 - val_accuracy: 1.0000
Epoch 8/10
500/500 [==============================] - 191s 382ms/step - loss: 2.9396e-07
- accuracy: 1.0000 - val_loss: 4.9739e-06 - val_accuracy: 1.0000
Epoch 9/10
500/500 [==============================] - 200s 401ms/step - loss: 2.4395e-07
- accuracy: 1.0000 - val_loss: 4.7480e-06 - val_accuracy: 1.0000
Epoch 10/10
500/500 [==============================] - 175s 350ms/step - loss: 2.0940e-07
- accuracy: 1.0000 - val_loss: 4.4782e-06 - val_accuracy: 1.0000
```

Out[9]:

```
<keras.src.callbacks.History at 0x25fff21e090>
```

In [10]:

```
[loss, acc] = model.evaluate(x_test,y_test,verbose = 1)
print("Accuracy:" + str(acc))
```

```
63/63 [==============================] - 6s 97ms/step - loss: 9.3993e-06 - ac
curacy: 1.0000
Accuracy:1.0
```

In [11]:

```python
model.save_weights('hand_gesture_model_weights.h5')
model.save("hand_gesture_model.h5")
```

```
C:\Users\ABC\anaconda3\Lib\site-packages\keras\src\engine\training.py:3079: U
serWarning: You are saving your model as an HDF5 file via `model.save()`. Thi
s file format is considered legacy. We recommend using instead the native Ker
as format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [13]:

```
import cv2
import mediapipe as mp

image = cv2.imread('handgest.jpg', cv2.IMREAD_GRAYSCALE)
height, width = image.shape

mp_hands = mp.solutions.hands
hands = mp_hands.Hands()

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

results = hands.process(image_rgb)

if results.multi_hand_landmarks:
    for landmarks in results.multi_hand_landmarks:
        for point in landmarks.landmark:
            x, y = int(point.x * width), int(point.y * height)
            cv2.circle(image, (x, y), 5, (0, 255, 0), -1)

img = cv2.resize(image, (320, 120))
arr = np.array(img)
arr = tf.reshape(arr, (-1, 120, 320, 1))

prediction = model.predict(arr)
print(prediction)

plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

```
1/1 [==============================] - 1s 713ms/step
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
```