

# 임베디드 보드 실습과 응용 프로그램

## Chapter 7.

인공지능 라이브러리 활용하기  
[모션 인식]

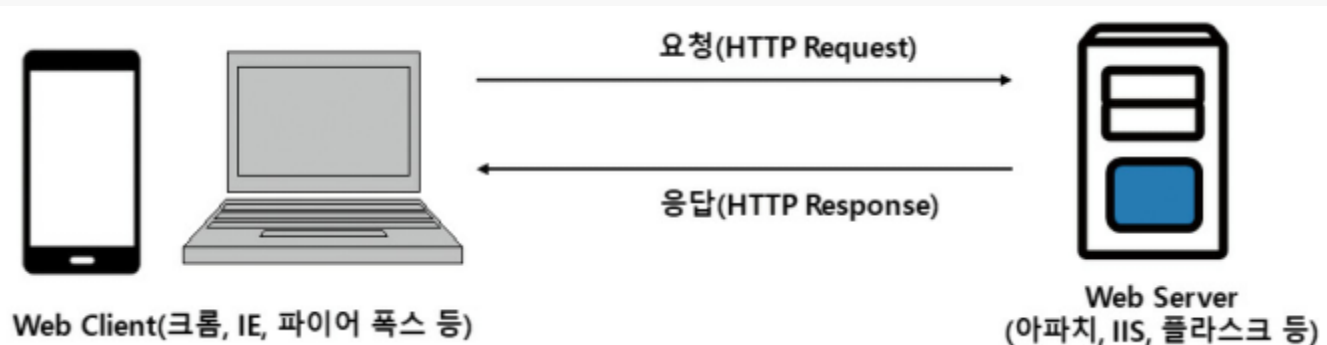
최영근

010-5898-3202

# 웹 서버

## • 웹 서버 역할

- web을 통해 서버 역할의 하드웨어에서 구동되는 서비스 프로그램
- 서버 컴퓨터의 메모리에 상주하면서 웹 브라우저 등 클라이언트 프로그램에서 요청한 HTTP 프로토콜에 대해 응답하는 프로그램



- 라즈베리 파이에서는 Flask, Django 등의 framework를 주로 사용해서 웹 서버 구현

# 웹 서버

- Flask

- 간단하게 웹 서버 기능을 구현하도록 해주는 micro web framework
- 용량이 작고 가벼우며 사용법도 간단

- <https://flask.palletsprojects.com/>

- 파이썬 버전 확인

- **python3 --version**

- flask 버전 확인해보고 설치되어 있다면 다음 페이지로

- **flask --version**

- 설치되어 있지 않다면 flask 설치

- **sudo apt-get install python3-flask**

```
pi@raspberrypi:~ $ python3 --version
Python 3.9.2
pi@raspberrypi:~ $ flask --version
Python 3.9.2
Flask 1.1.2
Werkzeug 1.0.1
```

# Flask 테스트

## • 테스트 프로그램

- 테스트 파일들을 저장할 디렉토리 생성

- pwd

```
pi@raspberrypi:~ $ pwd
```

```
/home/pi
```

- mkdir fla

```
pi@raspberrypi:~ $ mkdir fla
```

- 웹 브라우저에서 라즈베리 파이의 80번 포트([라즈베리 파이의 ip:80](#))에 접속하면 텍스트를 출력
- /home/pi/fla/test.py 파일로 저장

```
from flask import Flask
```


```
app = Flask(__name__)          # flask 객체 생성
```

```
@app.route('/')                # URL(/는 default 위치)로 요청이 오면  
def hello():                   # hello() 함수 실행  
    return 'Hello Flask'
```

```
if __name__ == '__main__':     # 80번 포트에 서비스 실행  
    app.run(debug=True, port=80, host='0.0.0.0')
```

# Flask 테스트

- 테스트 프로그램

- 테스트 파일이 저장된 디렉토리로 이동
- `cd fla`  `pi@raspberrypi:~ $ cd fla`
- 관리자 권한으로 파이썬 파일을 실행(이후 나오는 Flask를 이용한 파이썬 소스코드들은 모두 이런 방식으로 실행)
- `sudo python3 test.py`
- 웹 브라우저에 라즈베리 파이의 ip:80 을 입력하면 Hello Flask 라는 텍스트가 출력
- 라즈베리 파이의 IP는 아래 명령어를 사용하거나 테더링-연결된 기기에서 확인 가능
- `ifconfig`

# Flask 테스트

- 테스트 프로그램

```
pi@raspberrypi:~/fla $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether dc:a6:32:90:16:c6 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 652 bytes 1836208 (1.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 652 bytes 1836208 (1.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.180.33 netmask 255.255.255.0 broadcast 192.168.180.255
    inet6 fe80::c73:2d2b:688a:29ee prefixlen 64 scopeid 0x20<link>
    ether dc:a6:32:90:16:c7 txqueuelen 1000 (Ethernet)
    RX packets 12264 bytes 14786865 (14.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8529 bytes 3656243 (3.4 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

# Flask 웹 페이지 추가

- **@app.route() 함수**

- 여러 개의 서비스 페이지를 추가할 수 있음
- /home/pi/fla/sub.py 파일로 저장

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello Flask'

@app.route('/sub1')                                # /sub1로 요청이 오면 sub1 함수 실행
def sub1():
    return 'SUB1 Page'

@app.route('/sub2')                                # /sub2로 요청이 오면 sub2 함수 실행
def sub2():
    return 'SUB2 Page'

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

# Flask 웹 페이지 추가

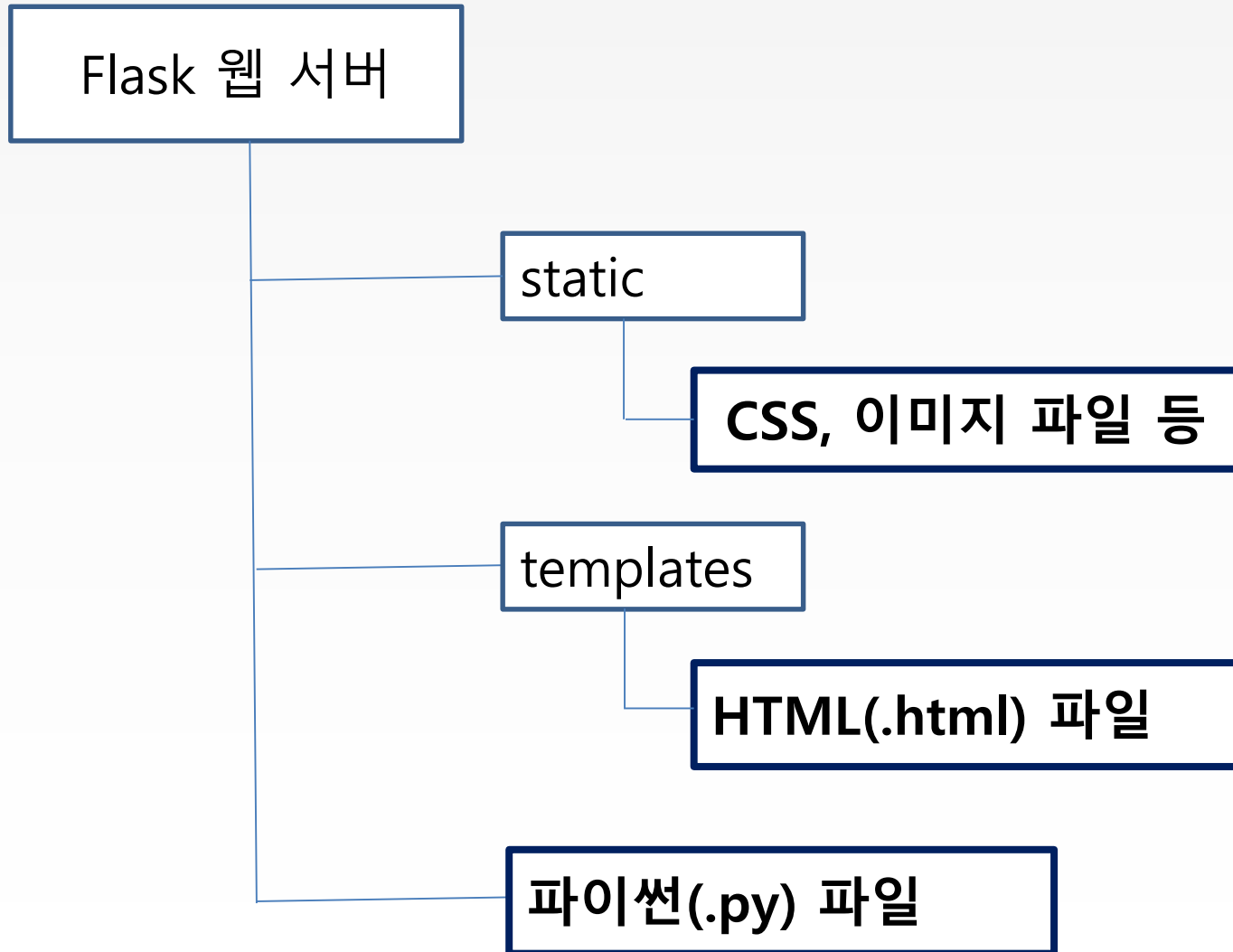
- **@app.route()** 함수

- 웹 브라우저에 아래 주소를 입력
- 라즈베리 파이의 ip/sub1
- 라즈베리 파이의 ip/sub2
- cd fla
- sudo python3 sub.py



# HTML 출력

- Flask 웹 서버 구조



# HTML 출력

- 디렉토리 생성

- HTML 파일과 이미지 등을 Flask에서 서비스할 디렉토리에 삽입한 후 `render_templates()` 함수를 이용해서 텍스트와 이미지를 보여주는 코드를 작성
- 아래 명령어를 입력해서 디렉토리 생성
  - `pwd`
  - `cd fla`
  - `mkdir templates`
  - `mkdir static`

# HTML 출력

- **templates, static 폴더에 파일 삽입**

- 구글드라이브 Ch.7 폴더의 f\_test.html 파일과 f\_test.png 파일을 아래 디렉토리에 넣기

|               |                     |                    |
|---------------|---------------------|--------------------|
| /home/pi/fla/ | <b>test_html.py</b> |                    |
|               | templates/          | <b>f_test.html</b> |
|               | static/             | <b>f_test.png</b>  |

- HTML 파일

```
<html>
<head>
<title>Flask HTML test page</title>
</head>
<body>
<center>
<br>
<strong>Flask HTML Test</strong>
<br>

</center>
</body>
</html>
```

# HTML 출력

## • 파이썬 코드 작성

- 다음과 같이 HTML 파일의 내용을 실행할 코드를 작성한 후
- 웹 브라우저에 **라즈베리 파이의 ip** 를 입력하면 이미지 파일이 출력

```
from flask import Flask, render_template    # render_template 패키지 사용
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('f_test.html')    # HTML 파일 내용을 실행

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

# get, post

## • get 방식 파라미터

- JSP, PHP, ASP 등과 같이 동적(dynamic, 상황에 따라 다른 내용을 출력)인 서비스를 위해 사용
- 웹 클라이언트에서 넘겨주는 파라미터를 Flask에서 받아 동적인 출력을 실행
- get, post 2가지 방식이 있음
- 웹 브라우저의 URL에 파라미터를 포함하여 작성하거나
- HTML에서 입력 태그에서 Submit 방식을 get으로 지정해서 요청

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route('/method', methods=['GET']) # method를 get으로 지정
def method():
    if request.method == 'GET':
        id = request.args["id"] # URL의 id 값
        password = request.args.get("password") # URL의 password 값
        return "Data(id : {}, pwd : {})".format(id, password)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

# get, post

- **get 방식 파라미터**

- 웹 서버의 URL 다음에 ? 를 삽입한 후 name=value 형식으로 작성
- 파라미터를 여러 개 보내려면 & 로 구분해서 입력하면 됨

- 웹 브라우저에

라즈베리 파이의 IP 주소/method?id=asdf&password=1234

입력

# get, post

- **get 방식 파라미터**

- 앞 페이지의 소스 코드는 매번 URL에 파라미터를 입력해야 하는 단점이 있음
- HTML 파일을 작성해서 get 방식으로 요청
- 구글 드라이브에 첨부한 method\_get.html 파일을 templates 디렉토리에 저장

```
<html>  
<head>  
  <title>GET method Test</title>  
</head>  
  
<body>  
  <h2>ID:{{ id }}, PASSWORD:{{ password }}</h2>    <!-- 전달된 값을 받아 출력 -->  
  <form method="get" action="/method_get_act">      <!-- submit 버튼을 눌렀을 때 get 방식으로 어떤 함수를 호출할지 결정 -->  
    <label id="Label1">id</label>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
    <input name="id" type="text" />  
    <label id="Label1"><br />password </label>  
    <input name="password" type="text" /><br /><br />  
    <input name="Submit1" type="submit" value="submit" />  
  </form>  
</body>  
</html>
```

# get, post

- get 방식 파라미터

- 아래와 같이 소스 코드를 작성

```
from flask import Flask, request, render_template
```

```
app = Flask(__name__)
```

```
@app.route('/method_get', methods=['GET'])
```

```
def method_get():
```

```
    return render_template('method_get.html')
```

# HTML의 <form method="get" action="/method\_get\_act"> 태그에 의해 호출

```
@app.route('/method_get_act', methods=['GET'])
```

```
def method_get_act():
```

```
    if request.method == 'GET':
```

```
        id = request.args["id"]
```

# HTML에서 요청한 id 읽기

```
        password = request.args.get("password")
```

# HTML에서 요청한 password 읽기

# 읽은 id, password 파라미터를 리턴

```
        return render_template('method_get.html', id=id, password=password)
```

```
if __name__ == '__main__':
```

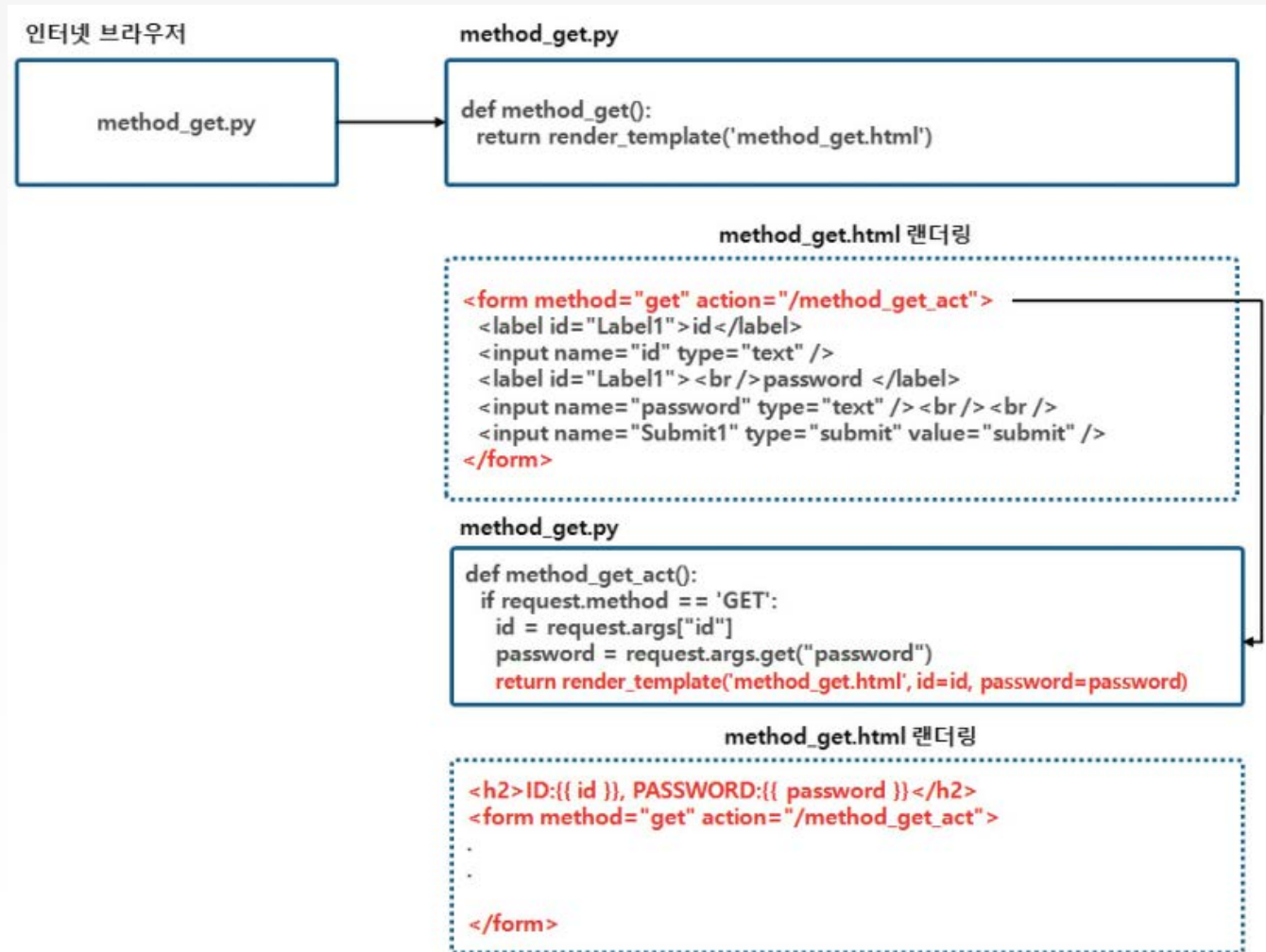
```
    app.run(debug=True, port=80, host='0.0.0.0')
```



# get, post

- get 방식 파라미터

- 다음과 같은 순서로 웹 브라우저와 Flask 서버 간에 호출됨



# get, post

- post 방식

- URL에 파라미터를 통해 요청하지 않음
- 첨부한 method\_post.html 파일을 templates 디렉토리에 저장
- `method="post"` 로 지정

```
<html>  
  <head>  
    <title>POST method Test</title>  
  </head>  
  
  <body>  
    <h2>ID: {{ id }}, PASSWORD: {{ password }}</h2>  
    <form method="post" action="/method_post_act"> <!-- submit 버튼을 눌렀을 때 post 방식으로 어떤 함수를 호출할지 결정 -->  
      <label id="Label1">id</label>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
      <input name="id" type="text" />  
      <label id="Label1"><br />password </label>  
      <input name="password" type="text" /><br /><br />  
      <input name="Submit1" type="submit" value="submit" />  
    </form>  
  </body>  
</html>
```

# get, post

## • post 방식

- p.15 get 방식 소스 코드와의 차이점은?
- ID, 패스워드 입력 후 Submit 버튼을 눌렀을 때 웹 브라우저에 표시되는 주소가 get 방식을 사용할 때와 어떻게 다른지 확인해 보시오.

```
from flask import Flask, request, render_template
```

```
app = Flask(__name__)
```

```
@app.route('/method_post', methods=['GET', 'POST'])
```

```
def method_post():
```

```
    return render_template('method_post.html')
```

```
@app.route('/method_post_act', methods=['GET', 'POST'])
```

```
def method_post_act():
```

```
    if request.method == 'POST':
```

```
        id = request.form["id"] # request.form에 의해 id,password 읽음
```

```
        password = request.form["password"]
```

```
        return render_template('method_post.html', id=id, password=password)
```

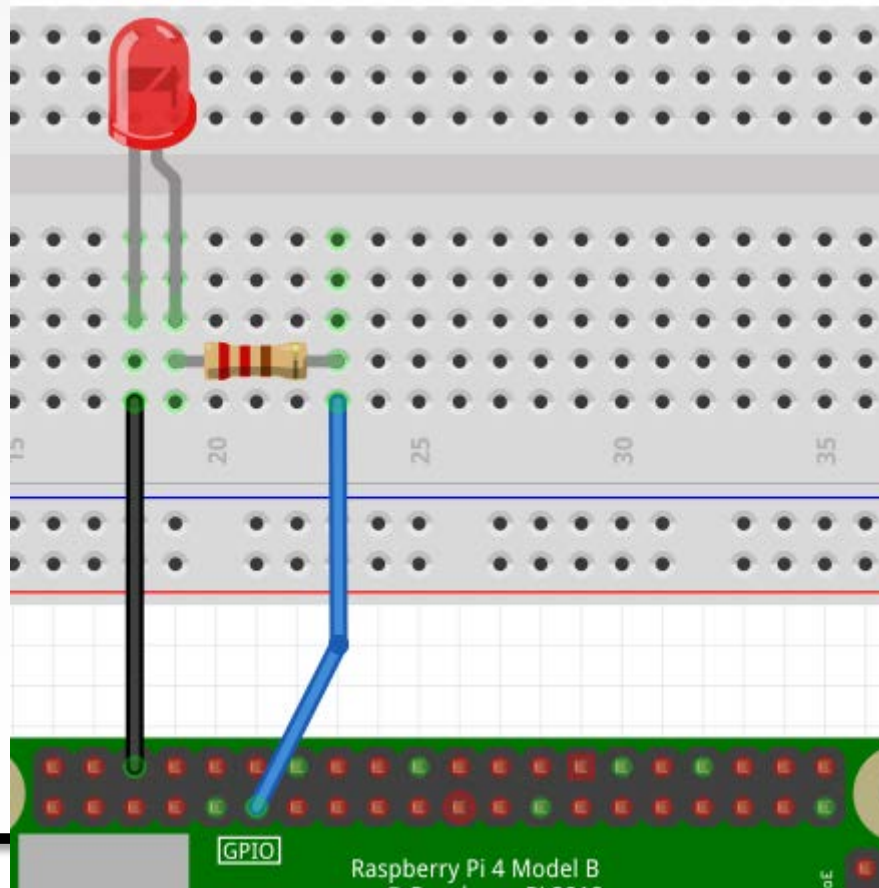
```
if __name__ == '__main__':
```

```
    app.run(debug=True, port=80, host='0.0.0.0')
```

# 웹 GPIO 제어

- LED On/Off 제어

- BCM 17 번 핀 – 220옴 저항 – LED - GND



# 웹 GPIO 제어

- LED On/Off 제어

- 첨부한 led\_control.html 파일을 templates 디렉토리에 저장

```
<html>
<head>
  <title>WEB LED Control</title>
  <style type="text/css">
    .auto-style1 {
      text-align: center;
    }
    .auto-style3 {
      background-color: #008000;
    }
    .auto-style6 {
      border-style: solid;
      border-color: #000000;
      text-align: center;
      color: #FFFFFF;
      background-color: #FF9900;
    }
  </style>
</head>
<body>
  <center>
    <strong><br>LED CONTROL<br><br><br></strong>
    <table style="width: 50%">
      <tr>
        <td class="auto-style6" style="height: 81; width: 30%"><strong>
          <!-- led_control_act 함수를 호출해서 파라미터가 1이면 LED ON -->
          <a href="led_control_act?led=1">LED ON</a></strong></td>
        <td class="auto-style1" style="height: 81; width: 50%">&nbsp;</td>
        <td class="auto-style6" style="height: 81; width: 30%"><strong>
          <!-- led_control_act 함수를 호출해서 파라미터가 2이면 LED OFF -->
          <a href="led_control_act?led=2">LED OFF</a>
        </strong></td>
      </tr>
    </table>
    <!-- 리턴된 ret 값에 의해 현재 LED 상태를 표시 -->
    <strong><br>LED is {{ ret }}</strong>
  </center>
</body>
</html>
```

# 웹 GPIO 제어

## • LED On/Off 제어

- 아래와 같이 소스 코드를 작성

```
from flask import Flask, request, render_template
import RPi.GPIO as gpio

LED = 17
gpio.setmode(gpio.BCM)
gpio.setup(LED, gpio.OUT)

app = Flask(__name__)

@app.route('/led_control')
def led_control():
    return render_template('led_control.html')

@app.route('/led_control_act', methods=['GET'])
def led_control_act():
    if request.method == 'GET':
        status = '' # 웹 페이지에 출력할 문자열
        led = request.args["led"]
        if led == '1': # HTML에서 요청한 led 값이 1이면 LED on, status를 ON으로
            gpio.output(LED, True)
            status = 'ON'
        else: # HTML에서 요청한 led 값이 1이 아니면 LED on, status를 OFF로
            gpio.output(LED, False)
            status = 'OFF'
        return render_template('led_control.html', ret=status)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

# 웹 GPIO 제어

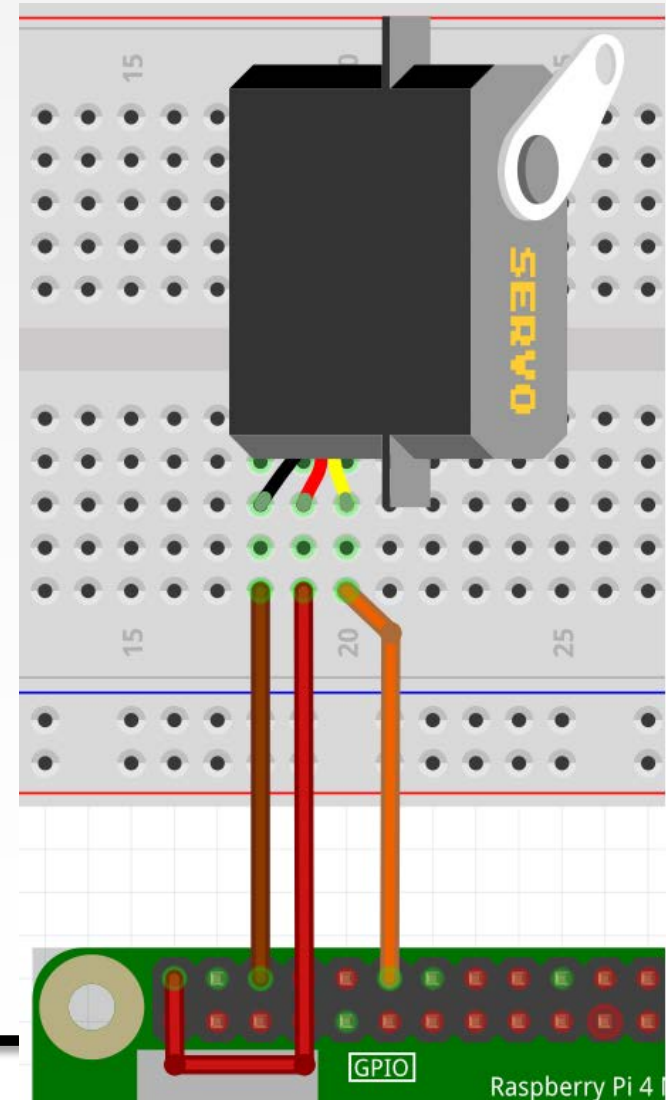
- LED On/Off 제어

- /home/pi/fla/led\_control.py 파일로 저장
- 관리자 권한으로 파이썬 파일을 실행
- **sudo python3 led\_control.py**
- 웹 브라우저에  
라즈베리 파이의 IP 주소/led\_control  
입력

# 웹 GPIO 제어

- 서보 모터 제어

- BCM 18 번 핀 – SG90의 PWM 핀 연결





# 웹 GPIO 제어

## • 서보 모터 제어

- 첨부한 sg90\_control.html 파일을 templates 디렉토리에 저장

```
<html>
<head>
<title>Servo Motor CONTROL</title>
<style type="text/css">
.auto-style1 {
    text-align: center;
}
.auto-style6 {
    border-style: solid;
    border-color: #000000;
    text-align: center;
    color: #FFFFFF;
    background-color: #FF9900;
}
</style>
</head>
<body>
<center>
<strong><br>SG-90 CONTROL<br><br><br></strong>
<table style="width: 50%">
<tr>
<td class="auto-style6" style="height: 81; width: 30%">
<a href="sg90_control_act?servo=L">&lt;&lt;</a></td>
<td class="auto-style1" style="height: 81; width: 50" align="center"></td>
<td class="auto-style6" style="height: 81; width: 30%;">
<a href="sg90_control_act?servo=R">&gt;&gt;</a></td>
</tr>
</table>
<strong><br>Degree is {{ degree }}</strong>
</center>
</body>
</html>
```

# 웹 GPIO 제어

## • 서보 모터 제어

- 아래와 같이 소스 코드를 작성 (1/2)

```
from flask import Flask, request, render_template
import RPi.GPIO as gpio
import time

servoPin = 18
SERVO_MAX_DUTY = 12
SERVO_MIN_DUTY = 3
cur_pos = 90      # 초기 상태는 90도 회전 상태

gpio.setmode(gpio.BCM)
gpio.setup(servoPin, gpio.OUT)

servo = gpio.PWM(servoPin, 50)
servo.start(0)

app = Flask(__name__)

def servo_control(degree, delay):
    if degree > 180: # 180도를 초과하지 못하도록
        degree = 180
    elif degree < 0: # 0도 미만이 되지 않도록
        degree = 0

    duty = SERVO_MIN_DUTY+(degree*(SERVO_MAX_DUTY-SERVO_MIN_DUTY)/180.0)
    print("Degree: {} to {}(Duty)".format(degree, duty))
    servo.ChangeDutyCycle(duty)
    time.sleep(delay)
    servo.ChangeDutyCycle(0)    # 다음 페이지에 계속
```

# 웹 GPIO 제어

## • 서보 모터 제어

- 아래와 같이 소스 코드를 작성 (2/2)

```
@app.route('/sg90_control')
def sg90_control():
    cur_pos = 90
    servo_control(cur_pos, 0.1)
    return render_template('sg90_control.html')

@app.route('/sg90_control_act', methods=['GET'])
def sg90_control_act():
    if request.method == 'GET':
        global cur_pos
        degree = ''
        servo = request.args["servo"]
        # request.args["servo"]로 요청한 인자가 L 이면 왼쪽으로 10도 회전
        if servo == 'L':
            cur_pos = cur_pos - 10
            if cur_pos < 0:
                cur_pos = 0
        # request.args["servo"]로 요청한 인자가 R 이면 오른쪽으로 10도 회전
        else:
            cur_pos = cur_pos + 10
            if cur_pos > 180:
                cur_pos = 180

        servo_control(cur_pos, 0.1)
        return render_template('sg90_control.html', degree=cur_pos)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

# 웹 GPIO 제어

- 서보 모터 제어

- /home/pi/fla/sg90\_control.py 파일로 저장
- 관리자 권한으로 파이썬 파일을 실행
- **sudo python3 sg90\_control.py**
- 웹 브라우저에  
라즈베리 파이의 IP 주소/sg90\_control  
입력

# OpenCV 모션 인식

## • 모션 인식

- 아래와 같이 소스 코드를 작성 (1/2)

```
import cv2
import numpy as np

threshold_move = 50      # 달라진 픽셀 값 기준치 설정. cv2.threshold() 함수에서 사용
diff_compare = 10        # 달라진 픽셀 갯수 기준치 설정
                        # cv2.countNonZero() 함수에 의해 달라진 픽셀 갯수 카운트

cap = cv2.VideoCapture(0, cv2.CAP_V4L)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)      # 영상의 폭을 320 으로 설정
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)     # 영상의 높이를 240 으로 설정

ret, img_first = cap.read()      # 1번째 프레임 읽기
ret, img_second = cap.read()     # 2번째 프레임 읽기

while True:
    ret, img_third = cap.read()   # 3번째 프레임 읽기
    scr = img_third.copy()        # 화면에 다른 점 표시할 이미지 백업

    # 그레이 스케일로 변경
    img_first_gray = cv2.cvtColor(img_first, cv2.COLOR_BGR2GRAY)
    img_second_gray = cv2.cvtColor(img_second, cv2.COLOR_BGR2GRAY)
    img_third_gray = cv2.cvtColor(img_third, cv2.COLOR_BGR2GRAY)

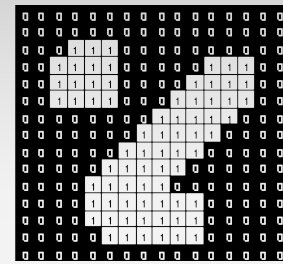
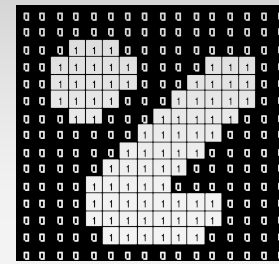
    # 이미지간의 차이점 계산
    diff_1 = cv2.absdiff(img_first_gray, img_second_gray)
    diff_2 = cv2.absdiff(img_second_gray, img_third_gray)

    # Threshold 적용해서 binarization
    ret, diff_1_thres = cv2.threshold(diff_1, threshold_move, 255, cv2.THRESH_BINARY)
    ret, diff_2_thres = cv2.threshold(diff_2, threshold_move, 255, cv2.THRESH_BINARY)
    # 다음 페이지에 계속
```

# OpenCV 모션 인식

## • 모션 인식

- 아래와 같이 소스 코드를 작성 (2/2)



```
# 1번째 영상-2번째 영상, 2번째 영상-3번째 영상 차이점
diff = cv2.bitwise_and(diff_1_thres, diff_2_thres)

# opening 연산으로 노이즈 제거 -- 위의 diff 변수를 주석 처리한 후
# k = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
# diff = cv2.morphologyEx(diff, cv2.MORPH_OPEN, k)

# 차이가 발생한 픽셀 갯수 판단 후 사각형 그리기
diff_cnt = cv2.countNonZero(diff)
if diff_cnt > diff_compare:
    nzero = np.nonzero(diff) # diff에서 0이 아닌 픽셀의 좌표(index) 얻기
    # min(nzero[1], nzero[0]) : 행, 열 중 가장 값이 작은 포인트
    # max(nzero[1], nzero[0]) : 행, 열 중 가장 값이 큰 포인트
    cv2.rectangle(scr, (min(nzero[1]), min(nzero[0])), #
                  (max(nzero[1]), max(nzero[0])), (0, 255, 0), 1)
    cv2.putText(scr, "Motion Detected", (10, 10), #
               cv2.FONT_HERSHEY_DUPLEX, 0.3, (0, 255, 0))

# 아래 2줄을 주석 해제하면 컬러 스케일 영상과 threshold 영상을 통합해서 출력
# stacked = np.hstack((scr, cv2.cvtColor(diff, cv2.COLOR_GRAY2BGR)))
# cv2.imshow('motion sensor', stacked)
cv2.imshow('scr', scr)

# 다음 비교를 위해 영상 저장
img_first = img_second
img_second = img_third

if cv2.waitKey(1) & 0xFF == 27:
    break

cap.release()
cv2.destroyAllWindows()
```

# OpenCV 모션 인식

- 모션 인식 스트리밍

- 첨부한 cv\_motion.html 파일을 templates 디렉토리에 저장

```
<html>
<head>
  <title>OpenCV motion detect streaming</title>
</head>
<body>
  <center>
    <h1>OpenCV motion detect streaming</h1>
    
  </div>
</body>
</html>
```

# OpenCV 모션 인식

## • 모션 인식 스트리밍

- 모션 인식 소스 코드를 아래와 같이 수정 (1/2)

```
from flask import Flask, render_template, Response
import cv2
import numpy as np

app = Flask(__name__)

threshold_move = 50
diff_compare = 10
img_first = None    # 추가
img_second = None   # 추가
img_third = None    # 추가

cap = cv2.VideoCapture(0, cv2.CAP_V4L)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

ret, img_first = cap.read()
ret, img_second = cap.read()

def gen_frames():
    while True:
        global img_first    # 추가
        global img_second   # 추가
        global img_third    # 추가
        global threshold_move # 추가
        global diff_compare  # 추가

        ret, img_third = cap.read()
        scr = img_third.copy()

        img_first_gray = cv2.cvtColor(img_first, cv2.COLOR_BGR2GRAY)
        img_second_gray = cv2.cvtColor(img_second, cv2.COLOR_BGR2GRAY)
        img_third_gray = cv2.cvtColor(img_third, cv2.COLOR_BGR2GRAY)

        diff_1 = cv2.absdiff(img_first_gray, img_second_gray)
        diff_2 = cv2.absdiff(img_second_gray, img_third_gray)
```



# OpenCV 모션 인식

## • 모션 인식 스트리밍

- 모션 인식 소스 코드를 아래와 같이 수정(2/2)

```
ret, diff_1_thres = cv2.threshold(diff_1, threshold_move, 255, cv2.THRESH_BINARY)
ret, diff_2_thres = cv2.threshold(diff_2, threshold_move, 255, cv2.THRESH_BINARY)

diff = cv2.bitwise_and(diff_1_thres, diff_2_thres)

diff_cnt = cv2.countNonZero(diff)
if diff_cnt > diff_compare:
    nzero = np.nonzero(diff)
    cv2.rectangle(scr, (min(nzero[1]), min(nzero[0])), #
                  (max(nzero[1]), max(nzero[0])), (0, 255, 0), 1)
    cv2.putText(scr, "Motion Detected", (10, 10), #
               cv2.FONT_HERSHEY_DUPLEX, 0.3, (0, 255, 0))

# cv2.imshow() 함수를 삭제

img_first = img_second
img_second = img_third

# 이 라인부터 아래 내용을 모두 추가
ret, buffer = cv2.imencode('.jpg', scr) # 동영상을 이미지 파일로 변환
frame = buffer.tobytes()                # 이미지 파일을 바이트 형식으로 리턴

# frame을 하나씩 실시간으로 전송
yield (b'--frame\r\n'
       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/cv_motion')
def cv_motion():
    return render_template('cv_motion.html')

@app.route('/video_feed')
def video_feed(): # 동영상 출력 실행
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    # 지금까지의 실습과 달리 debug 값을 True로 설정하면 오류가 발생할 수 있음
    app.run(debug=False, port=80, host='0.0.0.0')
```

# OpenCV 모션 인식

## • 카카오 개발자 센터

- <https://developers.kakao.com/> 에서 카카오 계정으로 회원 가입/로그인

- 내 애플리케이션에서

내 애플리케이션

제품

문서

도구

포럼

- 애플리케이션 추가



애플리케이션 추가하기

- 앱 아이콘, 앱 이름과  
사업자명을  
임의로 작성

### 애플리케이션 추가하기

앱 아이콘

이미지  
업로드

파일 선택

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

앱 이름

내 애플리케이션 이름

사업자명

사업자 정보와 동일한 이름

- 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.



서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영정책을 위반하지 않는 앱입니다.

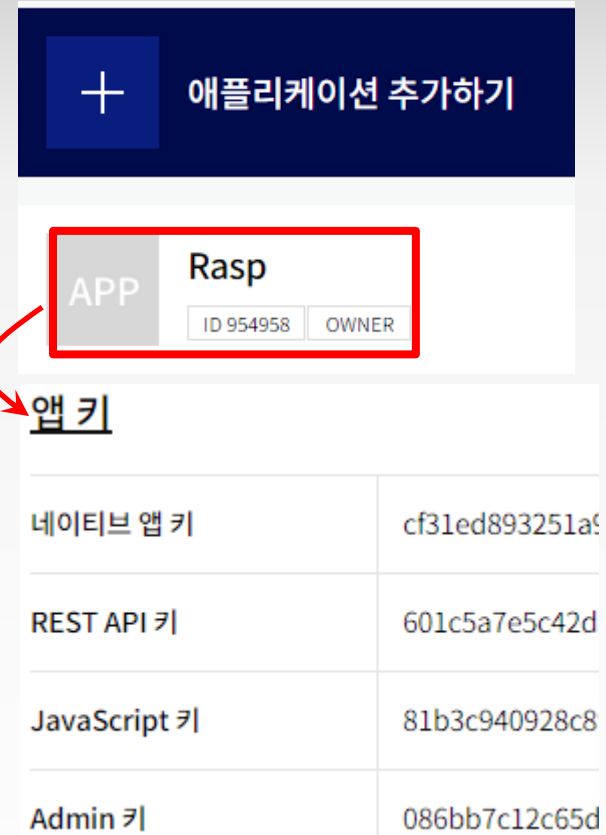
취소

저장

# OpenCV 모션 인식

## • 카카오 개발자 센터

- 추가된 앱을 클릭하면 앱 키를 확인할 수 있음



앱 키

|              |                |
|--------------|----------------|
| 네이티브 앱 키     | cf31ed893251a9 |
| REST API 키   | 601c5a7e5c42d  |
| JavaScript 키 | 81b3c940928c8  |
| Admin 키      | 086bb7c12c65d  |

- 왼쪽의 메뉴에서 카카오 로그인 클릭 후 활성화 상태 버튼을 클릭해서 ON



제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

카카오 로그인 OFF

활성화 설정

상태 OFF

카카오 로그인 ON

활성화 설정

상태 ON

# OpenCV 모션 인식

## • 카카오 개발자 센터

- 카카오 로그인 - 동의 항목  
맨 아래 접근권한에서 "카카오톡 메시지 전송" 설정

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

### 접근권한

| 항목 이름         | ID            | 상태                         |
|---------------|---------------|----------------------------|
| 카카오톡 스토리 글 목록 | story_read    | ● 사용 안함 <a href="#">설정</a> |
| 카카오톡 스토리 글 작성 | story_publish | ● 사용 안함 <a href="#">설정</a> |
| 카카오톡 메시지 전송   | talk_message  | ● 사용 안함 <a href="#">설정</a> |

# OpenCV 모션 인식

- 카카오 개발자 센터
  - 다음과 같이 설정

## 동의 항목 설정

### 항목

카카오톡 메시지 전송 / talk\_message

### 동의 단계

☒ 선택 동의

사용자가 동의하지 않아도 카카오 로그인을 완료할 수 있습니다.

☐ 이용 중 동의

카카오 로그인 시 동의를 받지 않고, 항목이 필요한 시점에 동의를 받습니다.

☐ 사용 안함

사용자에게 동의를 요청하지 않습니다.

### 동의 목적 [필수]

대학교 강의에서 라즈베리 파이와 Flask를 이용해 모션 인식 알림 기능 구현

개발자 앱 동의 항목 관리 화면내에 입력하는 사실이 실제 서비스 내용과 다를 경우 API 서비스의 거부 사유가 될 수 있습니다.

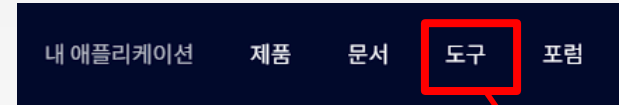
취소

저장

# OpenCV 모션 인식

## • 카카오 개발자 센터

- 도구에서 REST API 테스트 도구 바로가기 클릭



### REST API 테스트

API를 직접 호출하고 결과를 확인할 수 있는 테스트 도구입니다.

- 왼쪽의 메뉴에서 메시지 클릭 후  
나에게 기본 템플릿으로 메시지 보내기  
선택

#### 메시지

나에게 기본 템플릿으로 메시지 보내기

나에게 사용자 정의 템플릿으로 메시지 보내기

나에게 스크랩 메시지 보내기

친구에게 기본 템플릿으로 메시지 보내기

친구에게 사용자 정의 템플릿으로 메시지 보내기

친구에게 스크랩 메시지 보내기

도구 바로가기 →

# OpenCV 모션 인식

## • 카카오 개발자 센터

- 인증 앱을 새로 추가한 앱으로 변경하고 토큰 발급 클릭 - talk\_message가 선택되었는지 확인하고 클릭 - 전체 동의하고 계속하기 클릭

The screenshot shows the Kakao Developer Center interface with the following elements:

- 인증 (Authentication)** section:
  - 인증 앱 (Auth App):** Shows an app named "Rasp" with ID 954958 and role OWNER. A red box highlights the app icon, with a blue callout box saying "클릭해서 앱 선택" (Click to select app).
  - 인증 방식 (Auth Method):** "Access Token" is selected.
  - 엑세스 토큰 (Access Token):** A text box says "API 호출을 위해 액세스 토큰을 발급하세요." (Generate an access token for API calls). A red box highlights the "토큰 발급" (Generate Token) button.
- 권한 (Permissions):** A list of permissions where "talk\_message" is checked with a blue checkmark.
- Buttons:** "취소" (Cancel) and "확인" (Confirm) buttons are at the bottom left. A red box highlights the "확인" button.
- 동의 (Consent):** A box with a yellow checkmark and the text "전체 동의하기" (Grant all permissions). Below it, a note states: "전체동의를 선택목적에 대한 동의를 포함하고 있으며, 선택목적에 대한 동의를 거부해도 서비스 이용이 가능합니다." (Granting all permissions includes consent for the selected purpose, and service use is possible even if consent for the selected purpose is refused).

Red arrows indicate the flow: from the "토큰 발급" button to the "확인" button, and from the "확인" button to the "전체 동의하기" box.

# OpenCV 모션 인식

- 카카오 개발자 센터

- 액세스 토큰을 확인할 수 있음

|        |  |
|--------|--|
| 인증     |  |
| 인증 앱   | <div><div>APP</div><div>Rasp</div><div>ID 954958</div><div>OWNER</div></div>         |
| 인증 방식  | <input checked="" type="radio"/> Access Token  |
| 액세스 토큰 | <div>3wwQjOu9IFvoki4AyNgoCB4xfskHGMVvRMaPQUZ9CinI2QAAAYn5Vs5A</div> <div>토큰 발급</div> |

- <https://developers.kakao.com/tool/rest-api/open/post/v2-api-talk-memo-default-send>



# OpenCV 모션 인식

## • 카카오 개발자 센터

- 나에게 카카오톡 메시지 보내기 테스트

```
import json      # JavaScript Object Notation 모듈. 자바스크립트와 다른 언어 간에 데이터를 주고받을 때 사용
import requests

url = "https://kapi.kakao.com/v2/api/talk/memo/default/send"

headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "Authorization": "Bearer " + ₩
    "3wwQjOu9IFvoki4AyNgoCB4xfskHGMvVRMaPQUZ9CinI2QAAAYn5Vs5A" # 발급된 토큰
}

# 객체
data = {
    "template_object" : json.dumps      # json.dumps() : 파이썬 객체를 JSON 형식으로 직렬화(문자열 변환)하는 함수
    ({
        "object_type" : "text",
        "text" : "Motion Detected",    # 전송될 카카오톡 메시지
        "link" : {}
    })
}

response = requests.post(url, headers=headers, data=data)
print(response.status_code)
if response.json().get('result_code') == 0:
    print('Message send succeeded.')
else:
    print('Message send failed. : ' + str(response.json()))
```

# OpenCV 모션 인식

- 모션 인식이 되면 카카오톡 메시지 전송

- p.32, p.33 모션 인식 스트리밍 소스 코드와 합치기(1/3)

```
from flask import Flask, render_template, Response
import cv2
import numpy as np
import json
import requests

url = "https://kapi.kakao.com/v2/api/talk/memo/default/send"

headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "Authorization": "Bearer " + "hlzLriWR06hu84jnRMtSATopF2UGTGKWPLD4ww09dRsAAAF3n4ISRg"
}

data = {
    "template_object" : json.dumps(
        ({
            "object_type" : "text",
            "text" : "Motion Detected",
            "link" : {},
        })
    )
}

app = Flask(__name__)

threshold_move = 50
diff_compare = 10
img_first = None
img_second = None
img_third = None
```

# OpenCV 모션 인식

## • 모션 인식이 되면 카카오톡 메시지 전송

- p.32, p.33 모션 인식 스트리밍 소스 코드와 합치기(2/3)

```
cap = cv2.VideoCapture(0, cv2.CAP_V4L)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)

ret, img_first = cap.read()
ret, img_second = cap.read()

def gen_frames():
    while True:
        global img_first
        global img_second
        global img_third
        global threshold_move
        global diff_compare

        ret, img_third = cap.read()
        scr = img_third.copy()

        img_first_gray = cv2.cvtColor(img_first, cv2.COLOR_BGR2GRAY)
        img_second_gray = cv2.cvtColor(img_second, cv2.COLOR_BGR2GRAY)
        img_third_gray = cv2.cvtColor(img_third, cv2.COLOR_BGR2GRAY)

        diff_1 = cv2.absdiff(img_first_gray, img_second_gray)
        diff_2 = cv2.absdiff(img_second_gray, img_third_gray)

        ret, diff_1_thres = cv2.threshold(diff_1, threshold_move, 255, cv2.THRESH_BINARY)
        ret, diff_2_thres = cv2.threshold(diff_2, threshold_move, 255, cv2.THRESH_BINARY)

        diff = cv2.bitwise_and(diff_1_thres, diff_2_thres)

        diff_cnt = cv2.countNonZero(diff)
        if diff_cnt > diff_compare:
            nzero = np.nonzero(diff)
            cv2.rectangle(scr, (min(nzero[1]), min(nzero[0])), #
                          (max(nzero[1]), max(nzero[0])), (0, 255, 0), 1)
            cv2.putText(scr, "Motion Detected", (10, 10), #
                       cv2.FONT_HERSHEY_DUPLEX, 0.3, (0, 255, 0))
```

# OpenCV 모션 인식

- 모션 인식이 되면 카카오톡 메시지 전송

- p.32, p.33 모션 인식 스트리밍 소스 코드와 합치기(3/3)

```
# 카카오톡 메세지 보내기
response = requests.post(url, headers=headers, data=data)

img_first = img_second
img_second = img_third

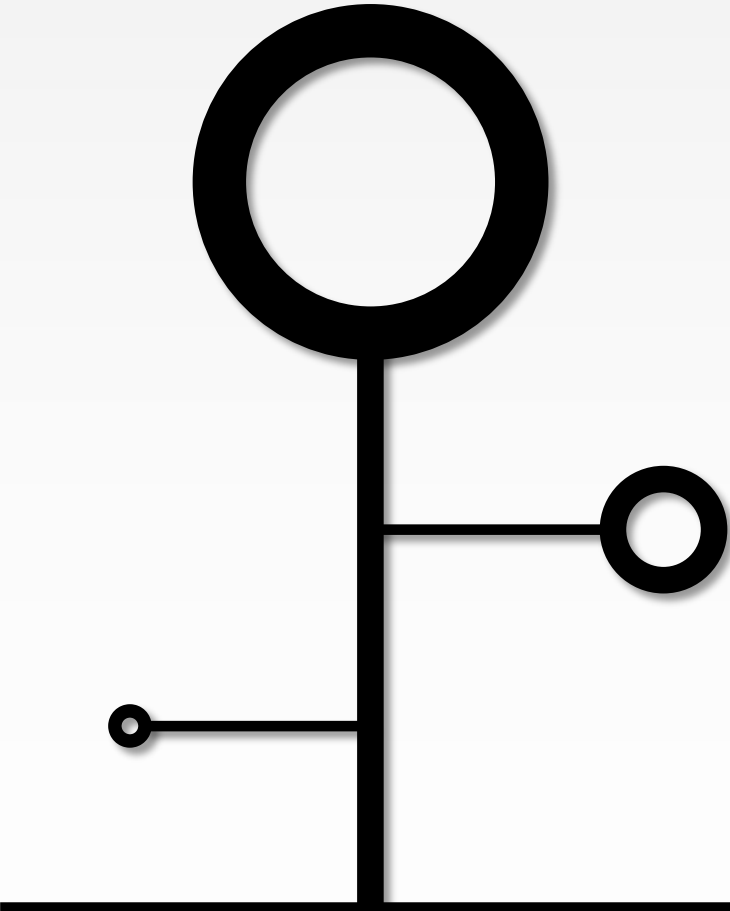
ret, buffer = cv2.imencode('.jpg', scr)
frame = buffer.tobytes()

yield (b'--frame\r\n'
       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/cv_motion')
def cv_motion():
    return render_template('cv_motion.html')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(debug=False, port=80, host='0.0.0.0')
```



THANK YOU

