스마트 모빌리티 프로그래밍

Ch 16. 파이썬 기반 Web Server 구현, 웹서비스 기반 원격제어

영남대학교 정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- Raspberry Pi에 WiFi AP 구성
- 파이썬 기반의 web server 구현
- Web service 기반의 원격 제어 기능 구조
- Web service 기반의 원격 스마트 조명제어 기능 구현

Raspberry Pi에 WiFi AP 구성

Raspberry Pi로 WiFi AP 만들기

◆ Raspberry Pi에 hostapd 기반 WiFi AP 구성

- 참고자료: https://limjunho.github.io/2020/08/25/Raspberry-Pi-AP%EB%A7%8C%EB%93%A4%EA%B8%B0.html

```
$ sudo apt update
$ sudo apt upgrade -y

$ sudo apt-get install hostapd -y
$ sudo apt-get install dnsmasq -y
$ sudo apt-get install iptables -y

$ sudo systemctl stop hostapd
$ sudo systemctl stop dnsmasq
```

hostapd 기반 WiFi AP 구성 (1)

◆ /etc/dhcpcd.conf 파일의 맨 아래에 다음 내용을 추가

```
interface wlan0
static ip_address=192.168.0.1/24
nohook wpa_supplicant
```

◆ /etc/hostapd/hostapd.conf 파일을 생성하고, 다음 내용을 포함

```
interface=wlan0
driver=nl80211
ssid=RPi_AP
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa_=2
wpa_passphrase=rpiap1234
wpa_key_mgmt.=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

hostapd 기반 WiFi AP 구성 (2)

◆ /etc/default/hostapd 파일 수정

- # DAEMON_CONF 의 주석 # 을 지우고 아래 문구 작성

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

◆ hostapd 동작 확인 (/etc/hostapd 폴더에서 수행)

```
# hostapd -d hostapd.conf
```

◆ Ctrl+C를 사용하여 hostapd 데몬 종료

hostapd 기반 WiFi AP 구성 (3)

◆ /etc/dnsmasq.conf에 아래 내용을 추가

- $ cp /etc/dnsmasq.conf /etc/dnsmasq.conf.orig => 백업본 생성
- /etc/dnsmasq.conf 파일 맨 끝에 다음 내용을 추가

```
interface=wlan0
listen-address=192.168.0.1
bind-interfaces
domain-needed
server=165.229.11.5
bogus-priv
dhcp-range=192.168.0.10, 192.168.0.100, 12h
```

◆ /etc/sysctl.conf 파일 수정

- # net.ipv4.ip_forward=1 에서 주석 표시인 # 를 삭제
  - net.ipv4.ip_forward=1

hostapd 기반 WiFi AP 구성 (4)

- **Port forwarding 설정 (Raspberry Pi에서 직접 실행)**

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables –A FORWARD –i eth0 –o wlan0 –m state
    --state RELATED,ESTABLISHED –j ACCEPT
# iptables –A FORWARD –i wlan0 –o eth0 –j ACCEPT
```

- **/etc/rc.local파일의 exit 0 줄 위에 다음 내용 입력**

```
iptables-restore < /etc/iptables.ipv4.nat
exit 0
```

- **Raspberry Pi 재부팅**

```
# reboot
```

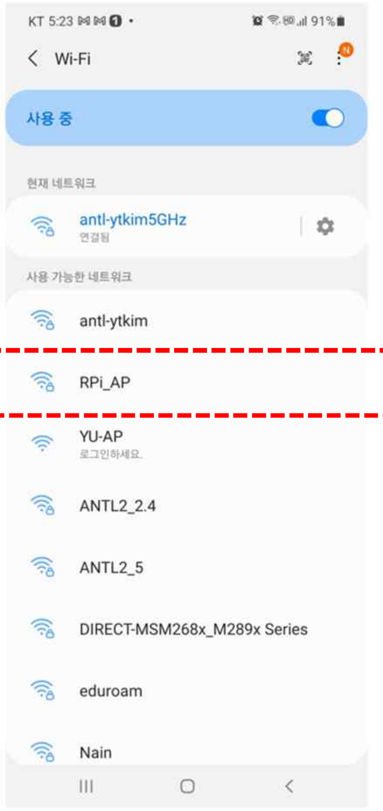
- **hostapd와 dnsmasq를 재 가동**

```
# service hostapd start
# service dnsmasq start
```

hostapd 기반 WiFi AP 구성 (5)

◆ 휴대폰에서 Wi-Fi AP 검색 및 연결확인

- ssid=RPi_AP



KT 5:23 91%

Wi-Fi

사용 중

현재 네트워크

antl-ytkim5GHz
연결됨

사용 가능한 네트워크

antl-ytkim

RPi_AP

YU-AP
로그인하세요.

ANTL2_2.4

ANTL2_5

DIRECT-MSM268x_M289x Series

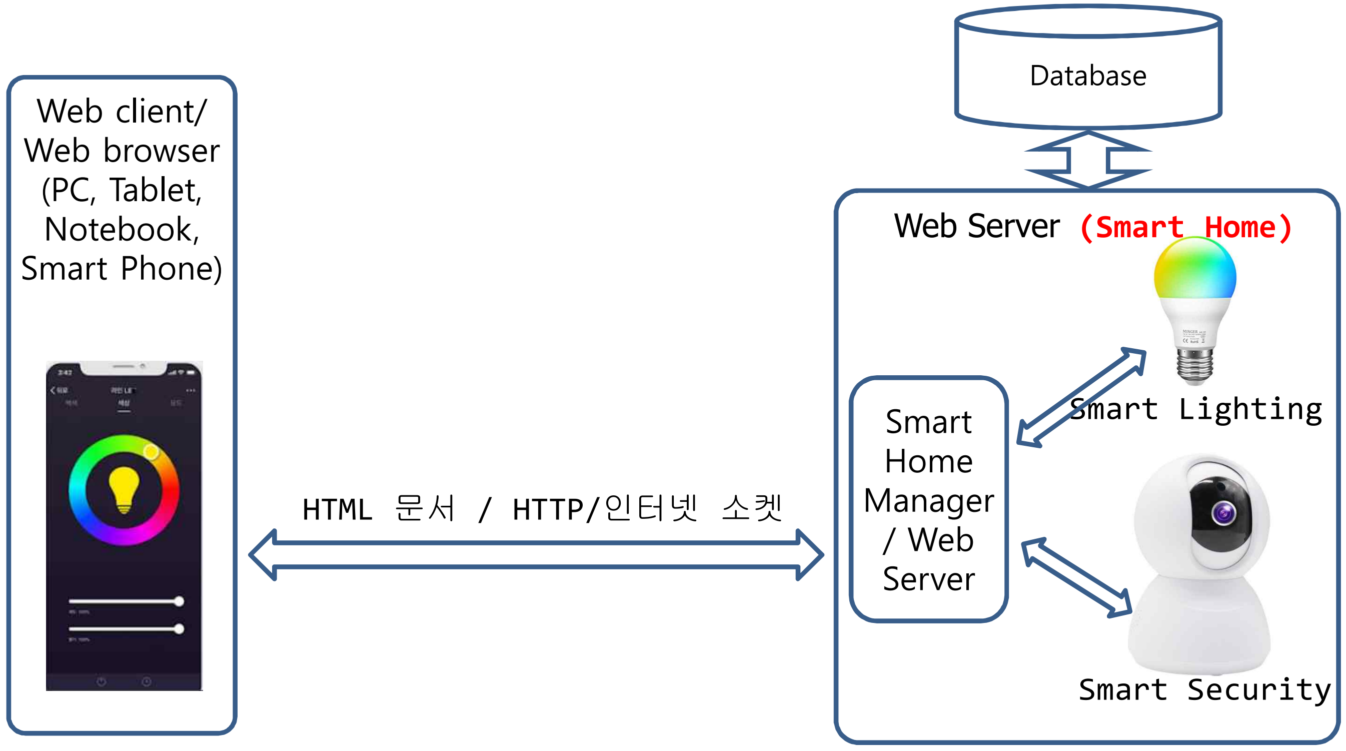
eduroam

Nain

파이썬 기반의 Web Server 구현

Web Service 구조

◆ Web Service 구조 - Web server, Web client, HTML

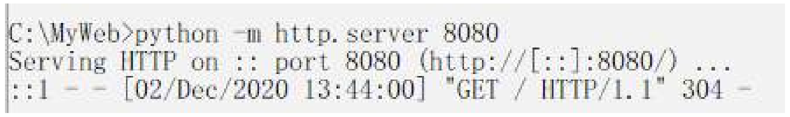


Web Server의 실행, Web Client 실행

◆ C:\MyWeb 디렉토리에 index.html 준비

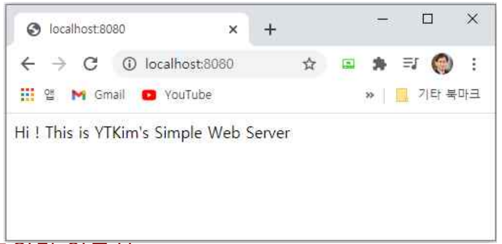
```
<html>
<body>
Hi !
This is YTKim's Simple Web Server
</body>
</html>
```

◆ Command 창에서 다음과 같이 http.server 실행



```
C:\MyWeb>python -m http.server 8080
Serving HTTP on :: port 8080 (http://[::]:8080/) ...
::1 - - [02/Dec/2020 13:44:00] "GET / HTTP/1.1" 304 -
```

◆ Client 실행: Web browser에서 http://localhost:8080



Hi ! This is YTKim's Simple Web Server

Python 환경에서의 Web Server 구현

◆ Flask 모듈

- https://flask.palletsprojects.com/en/2.0.x/
- Python 프로그램으로 Web application 개발을 할 수 있는 가벼운 micro web framework
- 파이썬 Flask 사용법 (기초) : https://hleecaster.com/flask-introduction/
- Raspberry Pi OS (Raspbian)에 포함되어 있음

◆ Bottle 모듈

- 파이썬 기본 라이브러리로 제공되며, 하나의 파일로 제공
- https://bottlepy.org/docs/dev/

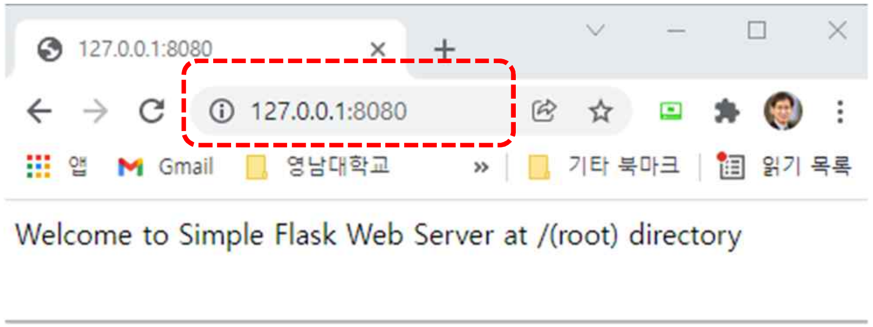
Python Flask 모듈 기반의 Web Server 구현

◆ SimpleWebServer_Flask.py

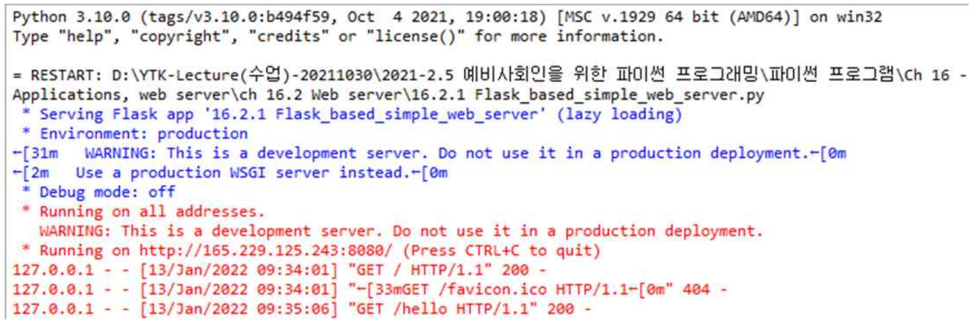
```
# Simple Web Server with Flask

from flask import Flask

# use current module name (__name__) in Flask constructor
simple_web = Flask(__name__)

# route() function of Flask defines the URL path of the invoked request
@simple_web.route('/')
def simpleWebServer_Flask():
    return "Welcome to Simple Flask Web Server at /(win) directory"

@simple_web.route('/hello')
def simpleWebServer_Hello():
    return "You accessed Simple Flask Web Server at /hello directory"

# run() method of Flask class executes the web application.
# if host='0.0.0.0' defined, it allows external access to the Web server
# port defines the port number for the web service request

if __name__ == '__main__':
    simple_web.run(host='0.0.0.0', port=8080, debug=True)
```

실행결과

◆ http://127.0.0.1:8080

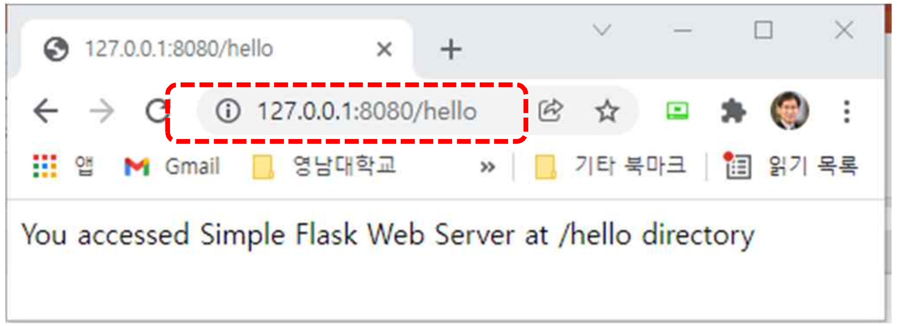


Welcome to Simple Flask Web Server at /(root) directory



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: D:\YTK-Lecture(수업)-20211030\2021-2.5 예비사회인을 위한 파이썬 프로그래밍\파이썬 프로그램\Ch 16 -
Applications, web server\ch 16.2 Web server\16.2.1 Flask_based_simple_web_server.py
 * Serving Flask app '16.2.1 Flask_based_simple_web_server' (lazy loading)
 * Environment: production
←[31m   WARNING: This is a development server. Do not use it in a production deployment.←[0m
←[2m   Use a production WSGI server instead.←[0m
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://165.229.125.243:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [13/Jan/2022 09:34:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [13/Jan/2022 09:34:01] "←[33mGET /favicon.ico HTTP/1.1←[0m" 404 -
127.0.0.1 - - [13/Jan/2022 09:35:06] "GET /hello HTTP/1.1" 200 -
```

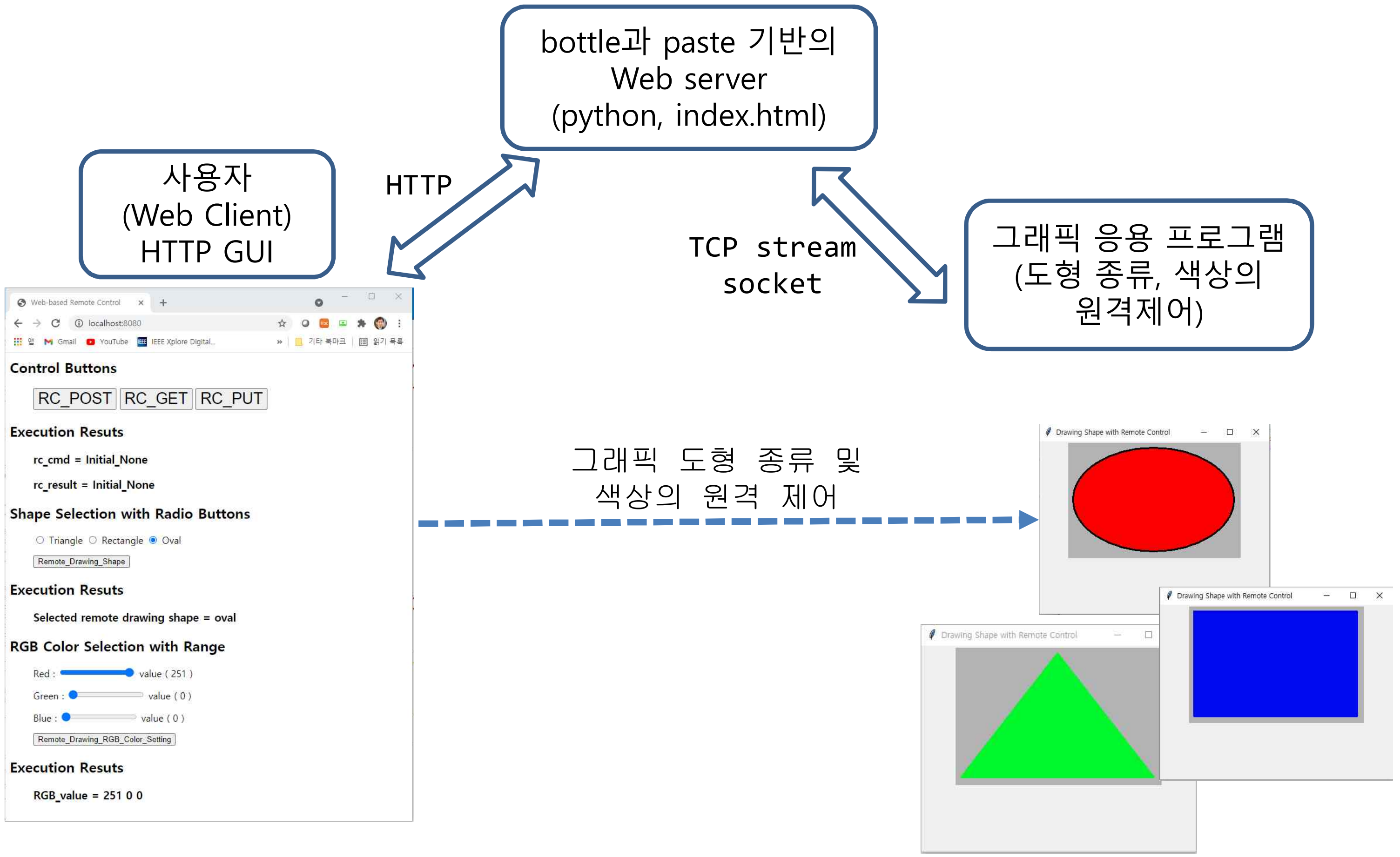
◆ http://127.0.0.1:8080/hello



You accessed Simple Flask Web Server at /hello directory

웹서버를 통한 그래픽 원격 제어

◆ 구성도


bottle과 paste 기반의 Web server (python, index.html)
사용자 (Web Client) HTTP GUI
HTTP
TCP stream socket
그래픽 응용 프로그램 (도형 종류, 색상의 원격제어)
그래픽 도형 종류 및 색상의 원격 제어
Web-based Remote Control
localhost:8080
Drawing Shape with Remote Control

Control Buttons

RC_POST RC_GET RC_PUT

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

Triangle Rectangle [✓ Oval]

Remote_Drawing_Shape

Execution Resuts

Selected remote drawing shape = oval

RGB Color Selection with Range

Red : value (251)

Green : value (0)

Blue : value (0)

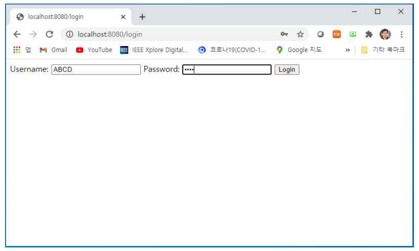
Remote_Drawing_RGB_Color_Setting

Execution Resuts

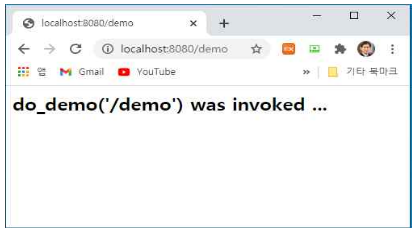
RGB_value = 251 0 0

Simple Python-based Web Server with bottle and paste

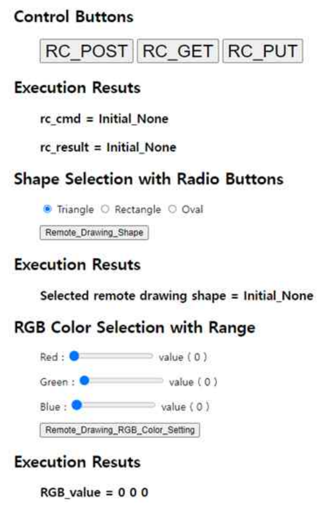
```
# web_remote_control_server.py - Simple Web Application with Python (1)
# for web server, bottle module is used
# for multi-threading, paste module is used
# So, bottle and paste modules must be installed before execution !!

import socket
from bottle import route, run, get, post, response, static_file, request

RC_drawing_port = 8088
Web_host_port = 8080

hostname = socket.gethostname()
hostAddr = socket.gethostbyname(hostname)
servSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servSock.bind((hostAddr, RC_drawing_port ))
print("Web server ({}) is waiting a client to connect ....".format(hostAddr))
servSock.listen(1)
sock_conn, cliAddr = servSock.accept()
print("Web Server is connected to the RC_Drawing client ({})...".format(cliAddr))
```



Username: ABCD Password: ••••



do_demo('/demo') was invoked ...



Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

[✓ Triangle] Rectangle Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

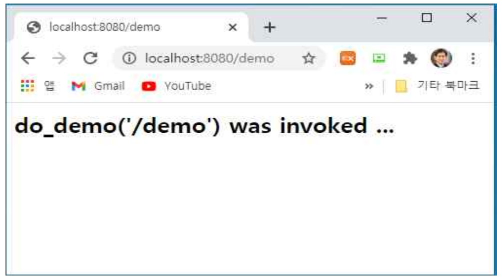
Green : value (0)

Blue : value (0)

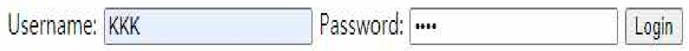
Execution Resuts

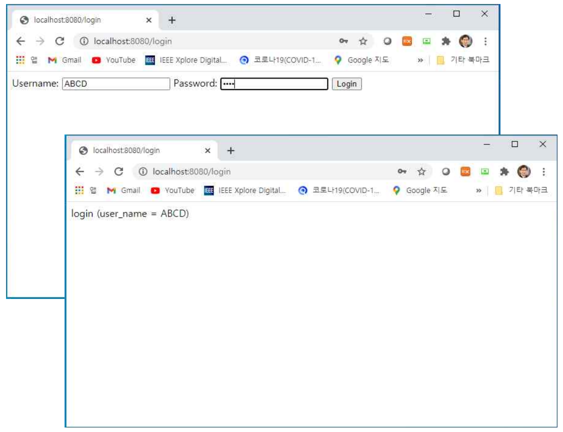
RGB_value = 0 0 0

```
# web_remote_control_server.py - Simple Web Application with Python (2)

@route('/') # invoked
def do_win_index():
    print("do_win_index('/') is invoked ==> ./index.html will be executed ...")
    return static_file("index.html", win=".")

@route('/demo') # invoked  by localhost:8080/demo
def do_demo():
    print("do_demo('/demo') was invoked ...")
    return "<H2>do_demo('/demo') was invoked ...</H2>"

@route('/login', method='GET')
def login():
    return '''
        <form action="/login" method="post">
        Username: <input name="username" type="text"/>
        Password: <input name="password" type="password" />
        <input value="Login" type="submit" />
        </form>
    '''

@route('/login', method='POST')
def do_login():
    username = request.forms.get('username')
    passwd = request.forms.get('password')
    return "login (user_name = {}, passwd = {})".format(username, passwd)
```


localhost:8080/demo

do_demo('/demo') was invoked ...


Username: KKK Password: •••• Login

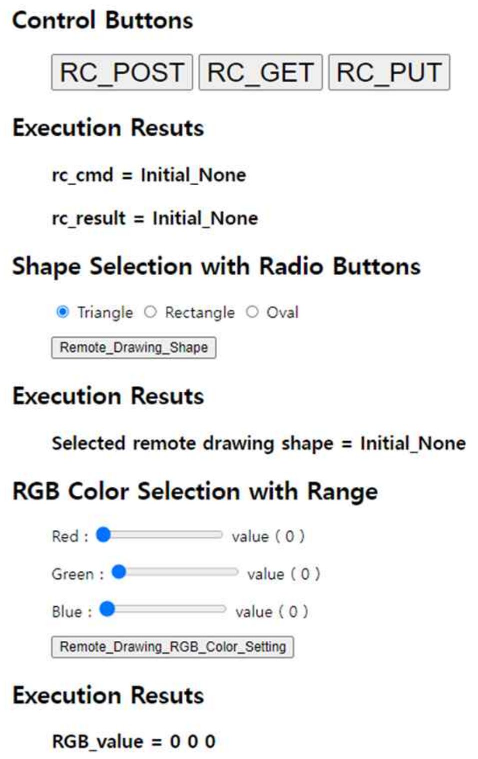

localhost:8080/login

Username: ABCD Password: •••• Login

localhost:8080/login

login (user_name = ABCD)

```
# web_remote_control_server.py - Simple Web Application with Python (3)

@route('/remote_control', method='POST')
def rc_POST():
    recv_cmd=request.forms.get('command')
    print("rc_POST({}) was invoked ...".format(recv_cmd))
    return_msg = "result of {}".format(recv_cmd)
    print("return_msg = {}".format(return_msg))
    return return_msg

@route('/remote_control', method='GET')
def rc_GET():
    print("rc_GET() was invoked ...")
    return_value = '7'
    return_msg = "result of RC_GET = {}".format(return_value)
    print("return_msg = {}".format(return_msg))
    return return_msg

@route('/remote_control', method='PUT')
def rc_PUT():
    recv_cmd=request.forms.get('put_value')
    print("rc_PUT({}) was invoked ...".format(recv_cmd))
    return_msg = "result of {}".format(recv_cmd)
    print("return_msg = {}".format(return_msg))
    return return_msg
```



Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Execution Resuts

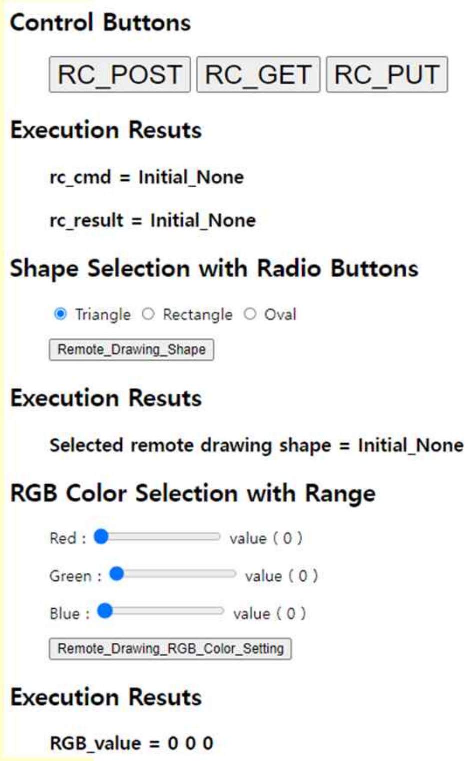
Selected remote drawing shape = Initial_None

RGB Color Selection with Range
Red : value (0)
Green : value (0)
Blue : value (0)
Remote_Drawing_RGB_Color_Setting

Execution Resuts

RGB_value = 0 0 0

```
# web_remote_control_server.py - Simple Web Application with Python (3)

@route('/remote_drawing_shape', method='POST')
def remote_drawing_shape_POST():
    shape_name = request.forms.get('remote_drawing_shape')
    print("Web Server::remote_drawing shape({}) was invoked ...".format(shape_name))
    msg_to_rc_drawing = "change_shape " + shape_name
    sock_conn.send(bytes(msg_to_rc_drawing.encode()))
    return_msg = "Web server::remote_drawn_shape({})".format(shape_name)
    print("return_msg = {}".format(return_msg))
    return return_msg

@route('/remote_drawing_color', method='POST')
def rgb_color_set_POST():
    rgb_value=request.forms.get('rgb_value')
    print("/remote_drawing - rgb_color_set_POST({}) was invoked ...".format(rgb_value))
    msg_to_rc_drawing = "change_color " + rgb_value
    sock_conn.send(bytes(msg_to_rc_drawing.encode()))
    return_msg = "Web server::rgb_color_set ({})".format(rgb_value)
    print("return_msg = {}".format(return_msg))
    return return_msg

#------------------------------

run(host='', port=Web_host_port, server='paste') # using Paste multi-thread web-server module
```



Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

● Triangle ○ Rectangle ○ Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

Green : value (0)

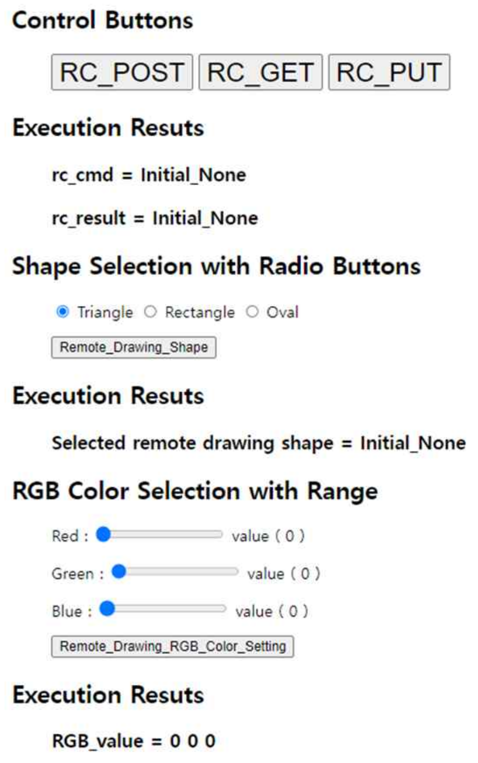
Blue : value (0)

Execution Resuts

RGB_value = 0 0 0

index.html

```
<html>
<head>
   <meta charset="UTF-8">
   <title>Web-based Remote Control</title>
   <meta name="viewport" content="width=200, initial-scale=1, maximum-scale=1">
</head>

   <script type="text/javascript">
      function rc_post(value) {
         var req_post = new XMLHttpRequest();
         var cmd_msg = "command=" + value;
         req_post.open('POST', '/remote_control', false);
            // 'false' makes the request synchronous
         req_post.setRequestHeader('Content-Type',
           'application/x-www-form-urlencoded');
         req_post.setRequestHeader('Content-Length', cmd_msg.length);
         req_post.setRequestHeader('Connection', 'close');
         req_post.send(cmd_msg);
         document.getElementById('rc_cmd').innerHTML = value;
         var res_msg = req_post.responseText;
         //alert("rc_post (" + value + ")_result = " + res_msg);
         document.getElementById('rc_result').innerHTML = res_msg;
      }
   </script>
```


Control Buttons

RC_POST RC_GET RC_PUT

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

● Triangle ○ Rectangle ○ Oval

Remote_Drawing_Shape

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

Green : value (0)

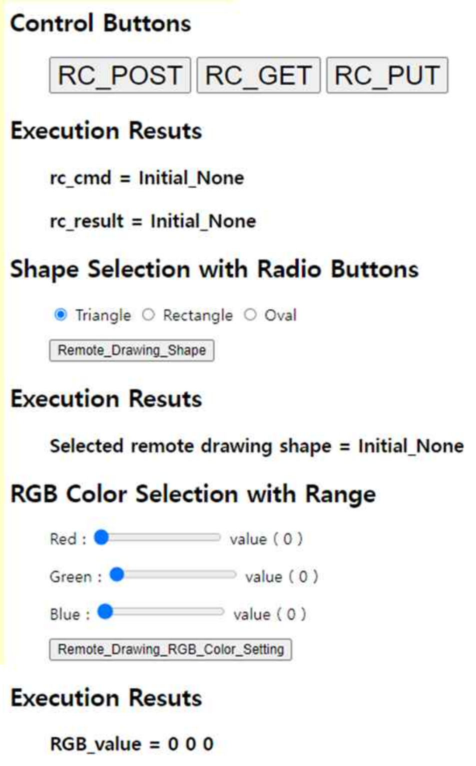
Blue : value (0)

Remote_Drawing_RGB_Color_Setting

Execution Resuts

RGB_value = 0 0 0

```
<script type="text/javascript">
    function rc_get(value) {
       var req_get = new XMLHttpRequest();
       var rc_cmd = "command=" + value;
       //alert("rc_get (" + value + ") invoked => rc_cmd = (" + rc_cmd + ")" );
       req_get.open('GET', '/remote_control', false); // 'false' makes the request synchronous
       req_get.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
       req_get.setRequestHeader('Content-Length', rc_cmd.length);
       req_get.setRequestHeader('Connection', 'close');
       req_get.send(rc_cmd);
       document.getElementById('rc_cmd').innerHTML = value;
       var res_msg = req_get.responseText;
       //alert("rc_get (" + value + ")_result = " + res_msg);
       document.getElementById('rc_result').innerHTML = res_msg;
    }
  </script>
```


Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

(● Triangle) ○ Rectangle ○ Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

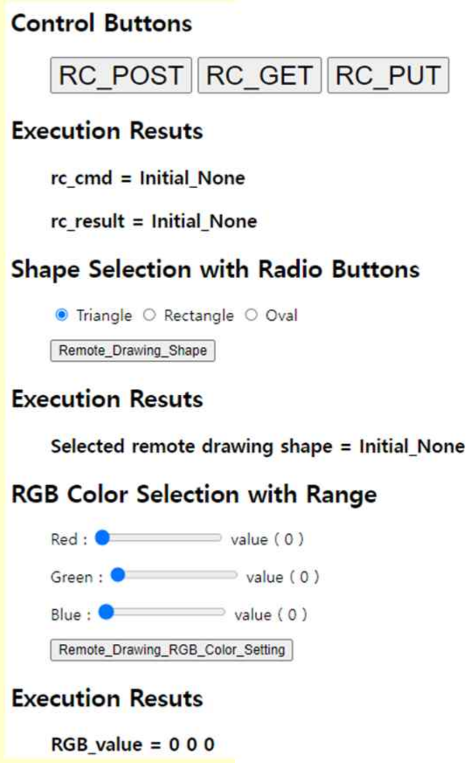
Green : value (0)

Blue : value (0)

Execution Resuts

RGB_value = 0 0 0

```
<script type="text/javascript">
  function rc_put(value) {
    var req_put = new XMLHttpRequest();
    var rc_cmd = "put_value=" + value;
    //alert("rc_put (" + value + ") invoked => rc_cmd = (" + rc_cmd + ")" );
    req_put.open('PUT', '/remote_control', false);
    // 'false' makes the request synchronous
    req_put.setRequestHeader('Content-Type',
      'application/x-www-form-urlencoded');
    req_put.setRequestHeader('Content-Length', rc_cmd.length);
    req_put.setRequestHeader('Connection', 'close');
    req_put.send(rc_cmd);
    document.getElementById('rc_cmd').innerHTML = value;
    var res_msg = req_put.responseText;
    //alert("rc_put (" + value + ")_result = " + res_msg);
    document.getElementById('rc_result').innerHTML = res_msg;
  }
</script>
```



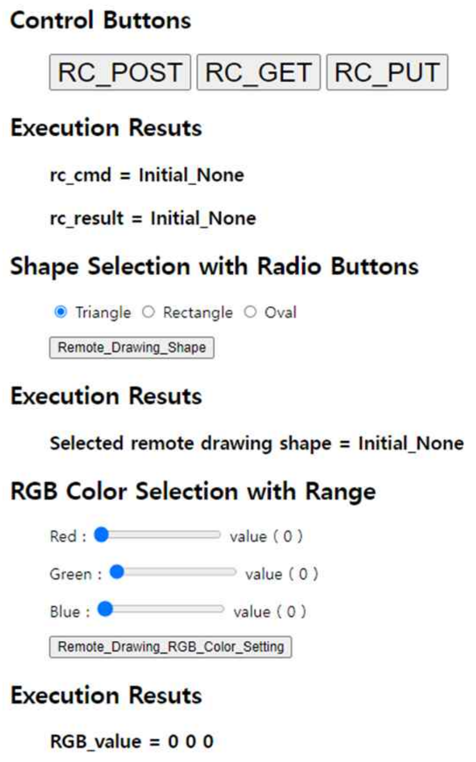
rc_cmd = Initial_None

rc_result = Initial_None

Selected remote drawing shape = Initial_None

RGB_value = 0 0 0

```
<script type="text/javascript">
  function Remote_Drawing_Shape() {
        //alert("Remote_Drawing_Shape")
    var radio_btn = document.getElementsByName("remote_drawing_shape")
        var radio_btn_check = 0;
        var selected_shape;
        for (var i=0; i<radio_btn.length; i++){
          if (radio_btn[i].checked == true) {
            selected_shape = radio_btn[i].value;
          }
        }
        document.getElementById('remote_drawing_shape').innerHTML = selected_shape;
        //alert("remote_drawing_shape (" + selected_shape + ")")
    var req_post = new XMLHttpRequest();
    var remote_drawing_cmd = "remote_drawing_shape=" + selected_shape;
    req_post.open('POST', '/remote_drawing_shape', false);
      // 'false' makes the request synchronous
    req_post.setRequestHeader('Content-Type',
      'application/x-www-form-urlencoded');
    req_post.setRequestHeader('Content-Length', remote_drawing_cmd.length);
    req_post.setRequestHeader('Connection', 'close');
    req_post.send(remote_drawing_cmd);

    var res_msg = req_post.responseText;
    //alert("remote_drawing_shape_(" + selected_shape + ")_result => "
    //    + res_msg);
  }
</script>
```


Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

[✓ Triangle] Rectangle Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

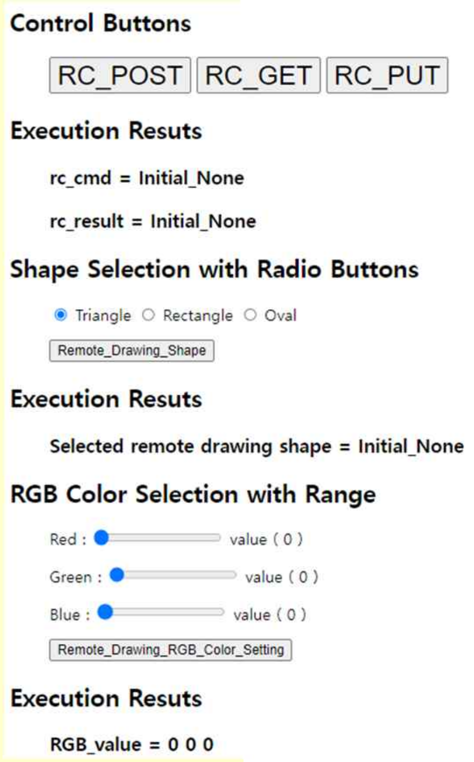
Green : value (0)

Blue : value (0)

Execution Resuts

RGB_value = 0 0 0

```
<script type="text/javascript">
  function Remote_RGB_Color_Set() {
    var req_post = new XMLHttpRequest();
          var rgb_value = document.getElementById('RGB_value').innerHTML
    var rc_rgb_cmd = "rgb_value=" + rgb_value;
    //alert("RGB_btn () invoked => rc_rgb_cmd = (" + rc_rgb_cmd + ")" );
    req_post.open('POST', '/remote_drawing_color', false);
     // 'false' makes the request synchronous
    req_post.setRequestHeader('Content-Type',
      'application/x-www-form-urlencoded');
    req_post.setRequestHeader('Content-Length', rc_rgb_cmd.length);
    req_post.setRequestHeader('Connection', 'close');
    req_post.send(rc_rgb_cmd);
    document.getElementById('rc_rgb_cmd').innerHTML = rgb_value;
    var res_msg = req_post.responseText;
    //alert("RGB_btn_(" + rgb_value + ")_result => " + res_msg);
  }
</script>

<script type="text/javascript">
  function Update_RGB_value()
  {
          //alert("Update_RGB_valued ()")
          var rd = document.getElementById('RGB_red_value').innerHTML;
          var gr = document.getElementById('RGB_green_value').innerHTML;
          var bl = document.getElementById('RGB_blue_value').innerHTML;
          var rgb = rd + " " + gr + " " + bl
          //alert("Update_RGB_valued (" + rgb + ")")
    document.getElementById('RGB_value').innerHTML = rgb
  }
</script>
```


Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

[✓ Triangle] Rectangle Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

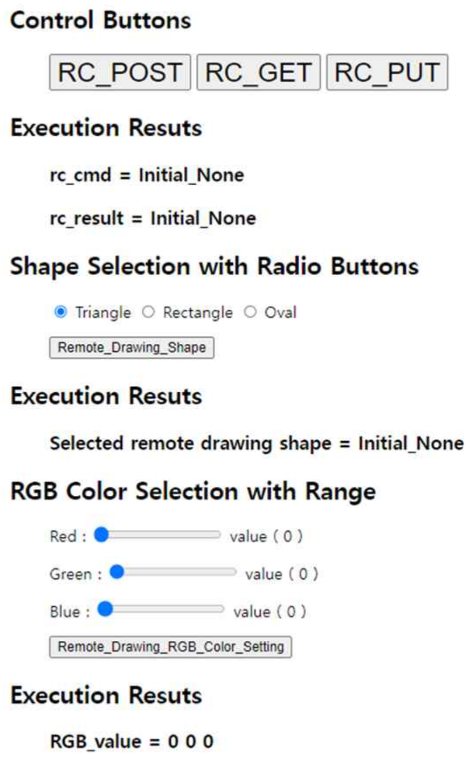
Green : value (0)

Blue : value (0)

Execution Resuts

RGB_value = 0 0 0

```
<script type="text/javascript">
  function Set_RGB_Red(value)
  {
        //alert("Set_RGB_Red (" + value + ")")
    document.getElementById('RGB_red_value').innerHTML = value
        Update_RGB_value()
  }
</script>
<script type="text/javascript">
  function Set_RGB_Green(value)
  {
        //alert("Set_RGB_Green (" + value + ")")
    document.getElementById('RGB_green_value').innerHTML = value
        Update_RGB_value()
  }
</script>
<script type="text/javascript">
  function Set_RGB_Blue(value)
  {
        //alert("Set_RGB_Blue (" + value + ")")
    document.getElementById('RGB_blue_value').innerHTML = value
        Update_RGB_value()
  }
</script>
```


Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

[✓ Triangle] Rectangle Oval

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

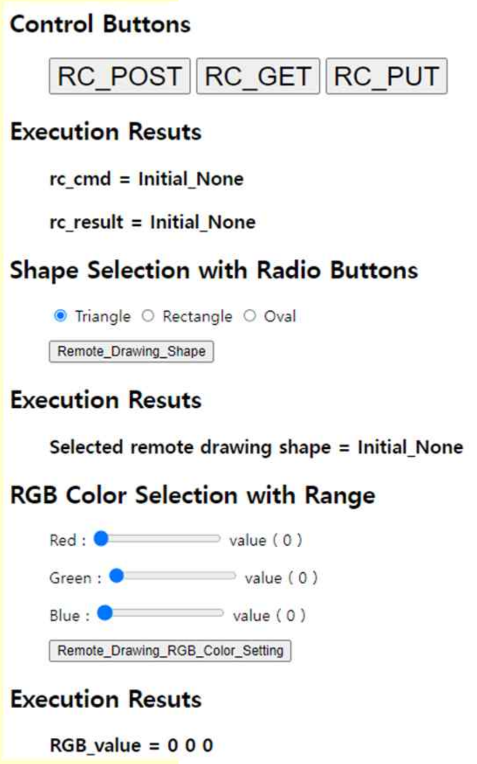
Green : value (0)

Blue : value (0)

Execution Resuts

RGB_value = 0 0 0

```
<body>
  <H2>Control Buttons</H2>
  <div align="left" stype="margin:0 0 10px 10px">
     <ul>
       <input type="button" style=font-size:20pt; width:70;height:60
          value="RC_POST" onClick="rc_post('RC_POST');">
       <input type="button" style=font-size:20pt; width:70;height:60
          value="RC_GET" onClick="rc_get('RC_GET');">
       <input type="button" style=font-size:20pt; width:70;height:60
          value="RC_PUT" onClick="rc_put('RC_PUT');">
     </ul>
  </div>
  <H2>Execution Resuts</H2>
  <div>
    <ul>
      <H3> rc_cmd  = <span id="rc_cmd"> Initial_None </span> </H3>
      <H3> rc_result  = <span id="rc_result"> Initial_None </span> </H3>
    </ul>
  </div>
  <H2>Shape Selection with Radio Buttons</H2>
  <div>
    <ul>
      <input type="radio" id="triangle" name= "remote_drawing_shape" value="triangle" checked>
      <lable for="triangle">Triangle</label>
      <input type="radio" id="rectangle" name= "remote_drawing_shape"  value="rectangle" >
      <lable for="rectangle">Rectangle</label>
      <input type="radio" id="oval" name= "remote_drawing_shape" value="oval">
      <lable for="oval">Oval</label>
    </ul>
  </div>
```


Control Buttons

RC_POST RC_GET RC_PUT

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

● Triangle ○ Rectangle ○ Oval

Remote_Drawing_Shape

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

Green : value (0)

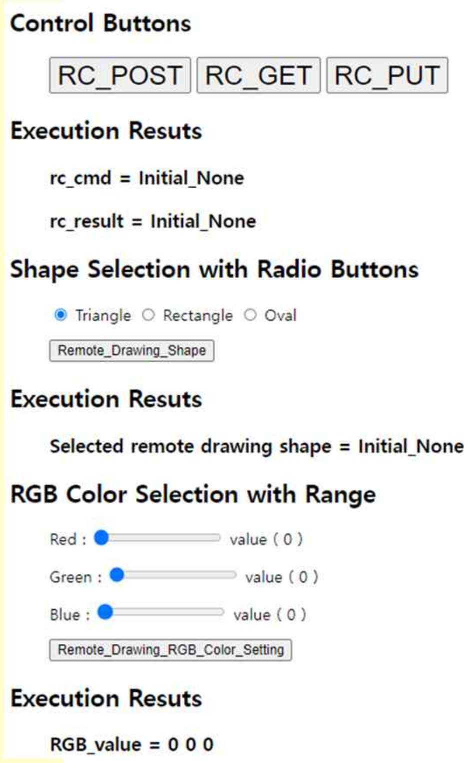
Blue : value (0)

Remote_Drawing_RGB_Color_Setting

Execution Resuts

RGB_value = 0 0 0

```
<div>
  <ul>
    <input type="button" style=font-size:10pt; width:70;height:60
      value="Remote_Drawing_Shape" onClick="Remote_Drawing_Shape();">
  </ul>
</div>
<H2>Execution Resuts</H2>
<div>
  <ul>
    <H3> Selected remote drawing shape  = <span id="remote_drawing_shape">
        Initial_None </span> </H3>
  </ul>
</div>

<H2>RGB Color Selection with Range</H2>
<div>
  <ul>
   <label for="Red"> Red   : </label>
    <input type="range" id="red" name="RGB_red_value" min="0" max="255" value=0
      onchange="Set_RGB_Red(this.value);">
  <label for="value"> value ( <span id="RGB_red_value"> 0 </span> ) </label>
  </ul>
  <ul>
    <lable for="green">Green : </label>
   <input type="range" id="green" name="RGB_green_value" min="0" max="255" value=0
       onchange="Set_RGB_Green(this.value);">
  <label for="value"> value ( <span id="RGB_green_value"> 0 </span> )
  </label>
  </ul>
```



Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Execution Resuts

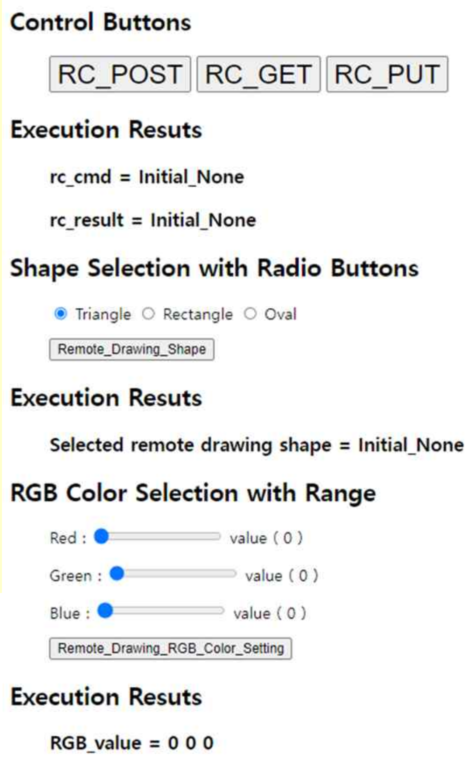
Selected remote drawing shape = Initial_None

RGB Color Selection with Range
Red : value (0)
Green : value (0)
Blue : value (0)
Remote_Drawing_RGB_Color_Setting

Execution Resuts

RGB_value = 0 0 0

```
      <ul>
        <lable for="blue">Blue  : </label>
        <input type="range" id="blue" name="RGB_blue_value" min="0" max="255" value=0
            onchange="Set_RGB_Blue(this.value);">
        <label for="value"> value ( <span id="RGB_blue_value"> 0 </span> ) </label>
      </ul>
      <ul>
         <input type="button" style=font-size:10pt; width:70;height:60
         value="Remote_Drawing_RGB_Color_Setting" onClick="Remote_RGB_Color_Set();">
      </ul>
   </div>
   <H2>Execution Resuts</H2>
   <div>
      <ul>
              <H3> RGB_value  = <span id="RGB_value"> 0 0 0 </span> </H3>
      </ul>
   </div>
</body>
</html>
```


Control Buttons

RC_POST RC_GET RC_PUT

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

[✓ Triangle] Rectangle Oval

Remote_Drawing_Shape

Execution Resuts

Selected remote drawing shape = Initial_None

RGB Color Selection with Range

Red : value (0)

Green : value (0)

Blue : value (0)

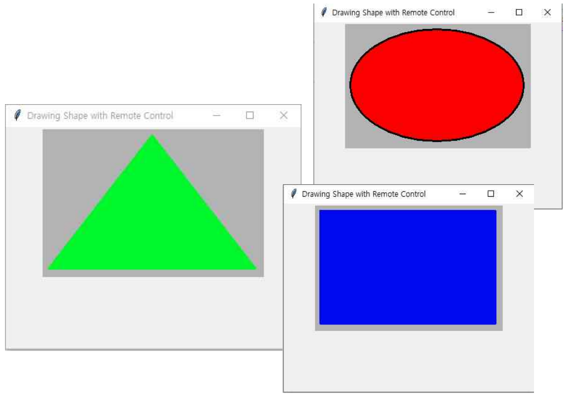
Remote_Drawing_RGB_Color_Setting

Execution Resuts

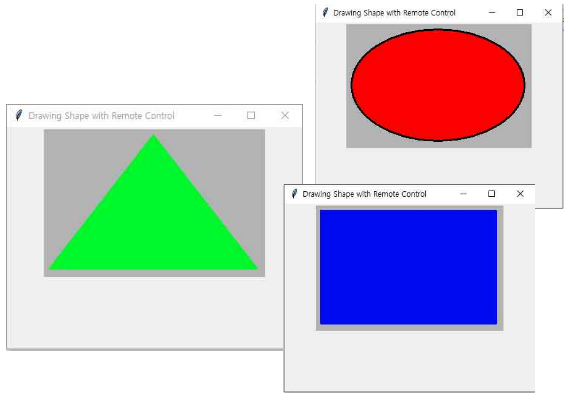
RGB_value = 0 0 0

RemoteControlled_Drawing

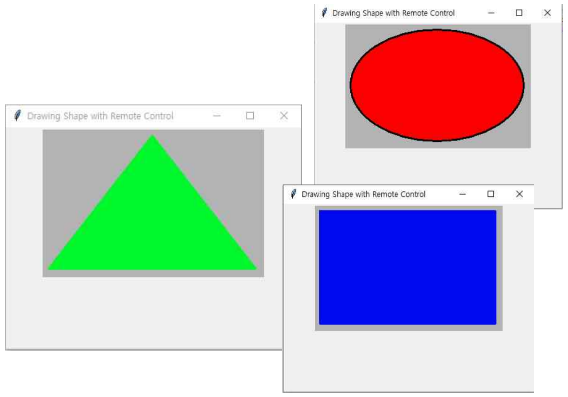
```
# RemoteControlledDrawing (1)
import time
from threading import Thread
from tkinter import *
import socket

class RemoteControlledDrawing:
    def __init__(self):

        self.win = win = Tk()
        self.win.geometry("400x300")
        self.win.wm_title('Drawing Shape with Remote Control')

        frame = Frame(self.win)
        frame.pack()
        self.canvas = Canvas(self.win, bg="grey70", width=300, height=200)
        self.canvas.pack()

        self.red = self.green = self.blue = 0
        self.color_sequence = 0
        self.colors = [(255,255,255), (255,0,0), (0,255,0), (0,0,255), (0,0,0)]
        self.shapes = ["oval", "triangle", "rectangle"]
        self.shape_name = "oval" # initial/default shape_name
        self.shape = self.canvas.create_oval(10, 10, 290, 190, fill="white", width=3)

        rc_drawing_agent_thread = Thread(target=self.rc_drawing_agent, daemon=True)
        rc_drawing_agent_thread.start()
```


Drawing Shape with Remote Control
Drawing Shape with Remote Control
Drawing Shape with Remote Control

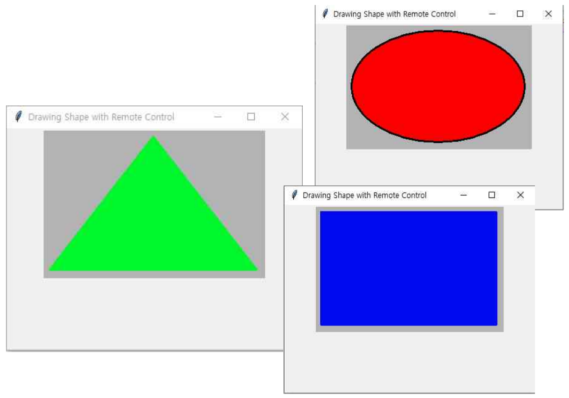
```
# RemoteControlledDrawing (2)
    def rc_drawing_agent(self):
        RC_drawing_port = 8088
        hostname = socket.gethostname()
        hostAddr = socket.gethostbyname(hostname)
        #servAddr_str = input("Server IP addr = ")
        servAddr_str = "165.229.185.251"
        cliSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        cliSock.connect((servAddr_str, RC_drawing_port ))
        servAddr = cliSock.getpeername()
        print("RC_drawing client is connected to server({})".format(servAddr))

        while True:
            recvMsg = cliSock.recv(100).decode()
            L_cmd = recvMsg.split()
            cmd = L_cmd[0]
            print("L_cmd = {}".format(L_cmd))
            if cmd == "change_shape":
                self.changeShape(L_cmd[1])
            if cmd == "change_color":
                red_str, green_str, blue_str = L_cmd[1], L_cmd[2], L_cmd[3]
                red, green, blue = int(red_str), int(green_str), int(blue_str)
                self.changeColor_RGB((red, green, blue))

        cliSock.close()

    def changeColor(self):
        color_seq = (self.color_sequence) % len(self.colors)
        (rd, gr, bl) = self.colors[color_seq]
        self.red = rd
        self.green = gr
        self.blue = bl
        color = "#%02x%02x%02x"%(self.red, self.green, self.blue)
        self.canvas.itemconfig(self.shape, fill=color)
        self.color_sequence += 1
```


Drawing Shape with Remote Control
Drawing Shape with Remote Control
Drawing Shape with Remote Control

```
# RemoteControlledDrawing (3)

    def changeColor_RGB(self, color_code):
        (rd, gr, bl) = color_code
        self.red = rd
        self.green = gr
        self.blue = bl
        color = "#%02x%02x%02x"%(self.red, self.green, self.blue)
        self.canvas.itemconfig(self.shape, fill=color)
        self.color_sequence += 1

    def draw_oval(self, color_code):
        self.canvas.delete("all")
        (rd, gr, bl) = color_code
        color = "#%02x%02x%02x"%(rd, gr, bl)
        self.shape = self.canvas.create_oval(10, 10, 290, 190, outline=color, \
            fill="white", width=3)
        self.canvas.itemconfig(self.shape, fill=color)

    def draw_triangle(self, color_code):
        self.canvas.delete("all")
        (rd, gr, bl) = color_code
        color = "#%02x%02x%02x"%(rd, gr, bl)
        points = [10, 190, 290, 190, 150, 10]
        self.shape = self.canvas.create_polygon(points, outline=color, \
         fill="white", width=3)
        self.canvas.itemconfig(self.shape, fill=color)

    def draw_rectangle(self, color_code):
        self.canvas.delete("all")
        (rd, gr, bl) = color_code
        color = "#%02x%02x%02x"%(rd, gr, bl)
        points = [10, 190, 290, 190, 290, 10, 10, 10]
        self.shape = self.canvas.create_polygon(points, outline=color, \
          fill="white", width=3)
```


Drawing Shape with Remote Control
Drawing Shape with Remote Control
Drawing Shape with Remote Control

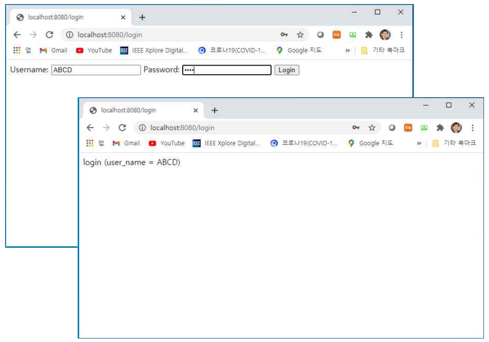
```
# RemoteControlledDrawing (4)

    def changeShape(self, shape_name):
        if shape_name in self.shapes:
            self.shape_name = shape_name
        else:
            self.shape_name = "oval" # default shape
        color = (self.red, self.green, self.blue)
        if self.shape_name == "triangle":
            self.draw_triangle(color)
        elif self.shape_name == "rectangle":
            self.draw_rectangle(color)
        else:
            self.draw_oval(color)


if __name__ == "__main__":
    global app

    app = RemoteControlledDrawing()
    app.win.mainloop()
```


Drawing Shape with Remote Control
Drawing Shape with Remote Control
Drawing Shape with Remote Control

Web Browser에서의 실행결과

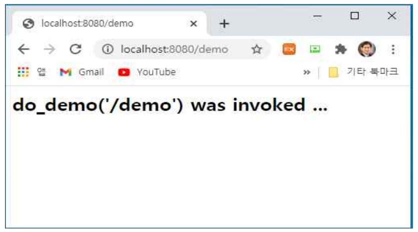
◆ Web Browser에서 실행

- http://localhost:8080/login
- http://localhost:8080/demo
- http://localhost:8080

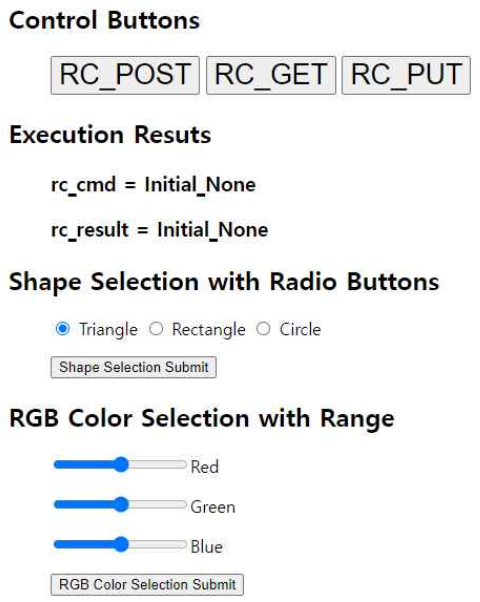

localhost:8080/login
localhost:8080/login
앱 Gmail YouTube IEEE Xplore Digital... 코로나19(COVID-1... Google 지도 기타 북마크
Login

Username: ABCD Password: ••••

login (user_name = ABCD)



do_demo('/demo') was invoked ...



Control Buttons

Execution Resuts

rc_cmd = Initial_None

rc_result = Initial_None

Shape Selection with Radio Buttons

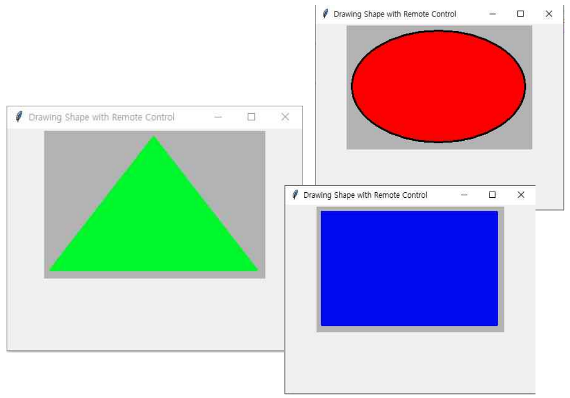
[✓ Triangle] Rectangle Circle

RGB Color Selection with Range

Red

Green

Blue


Drawing Shape with Remote Control
Drawing Shape with Remote Control
Drawing Shape with Remote Control

파이썬 Flask 모듈 기반의
Web Server 구현

Flask – micro web framework

◆ Flask 모듈

- https://flask.palletsprojects.com/en/2.0.x/
- Python 프로그램으로 Web application 개발을 할 수 있는 가벼운 micro web framework
- 파이썬 Flask 사용법 (기초) : https://hleecaster.com/flask-introduction/
- Raspberry Pi OS (Raspbian)에 포함되어 있음

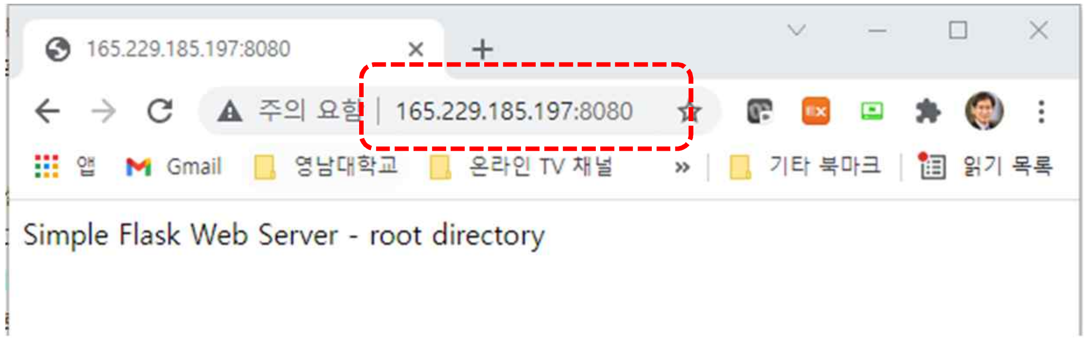
Python Flask 모듈 기반의 Web Server 구현

◆ SimpleWebServer_Flask.py

```python
# Simple Web Server with Flask

from flask import Flask

# use current module name (__name__) in Flask constructor
simple_web = Flask(__name__)

# route() function of Flask defines the URL path of the invoked request
@simple_web.route('/')
def simpleWebServer_Flask():
    return 'Simple Flask Web Server - root directory'

@simple_web.route('/hello')
def simpleWebServer_Hello():
    return "Simple Flask Web Server - hello directory"

# run() method of Flask class executes the web application.
# if host='0.0.0.0' defined, it allows external access to the Web server
# port defines the port number for the web service request

if __name__ == '__main__':
    simple_web.run(host='0.0.0.0', port=8080, debug=True)
```

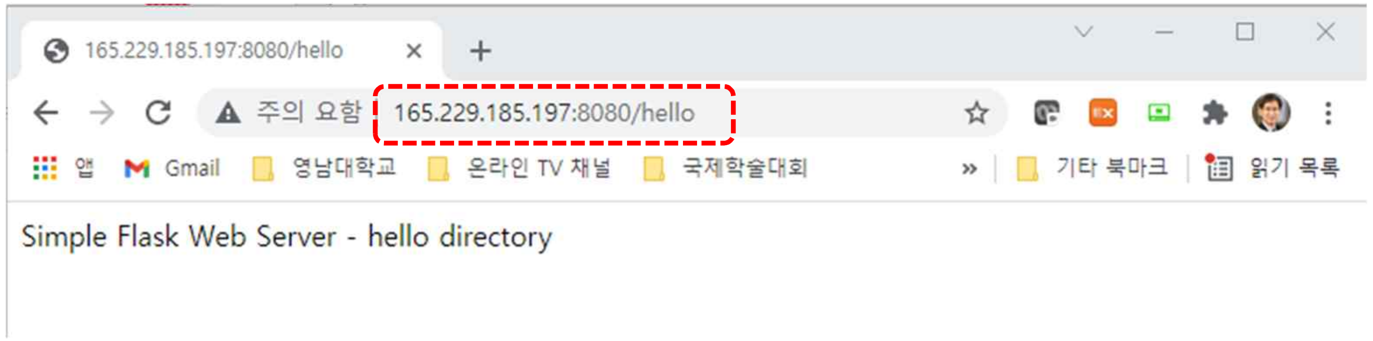
실행결과

◆ http://RPi_addr:8080



Simple Flask Web Server - root directory

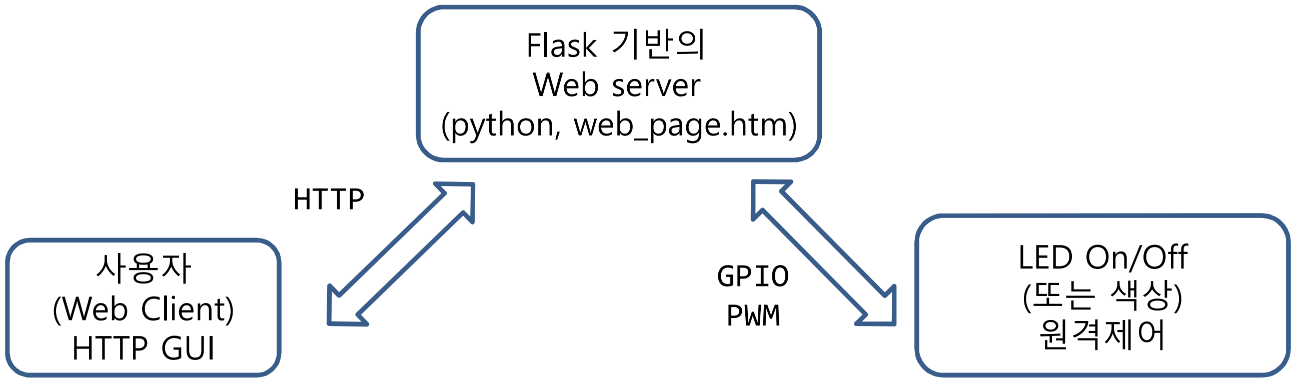
◆ http://RPi_addr:8080/hello

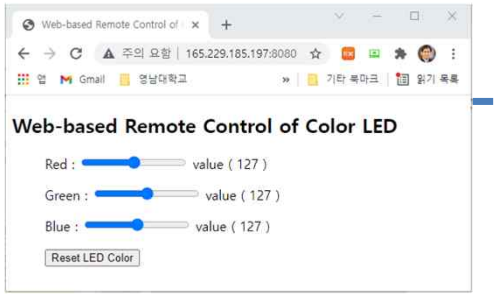


Simple Flask Web Server - hello directory

웹서버를 통한 그래픽 원격 제어

◆ 구성도





Web-based Remote Control of Color LED

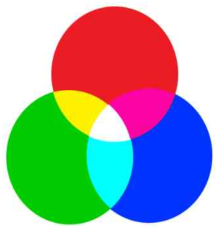
Red : value (127)

Green : value (127)

Blue : value (127)

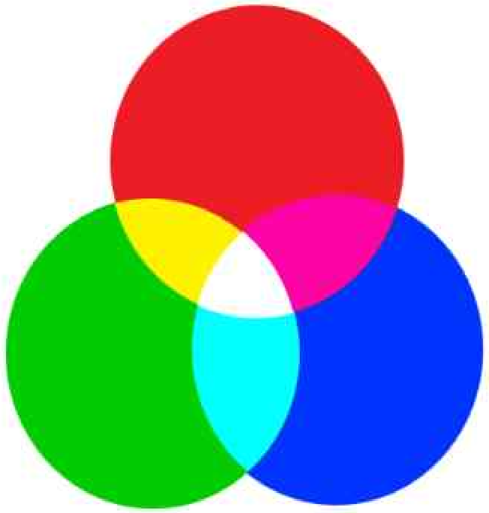
Reset LED Color

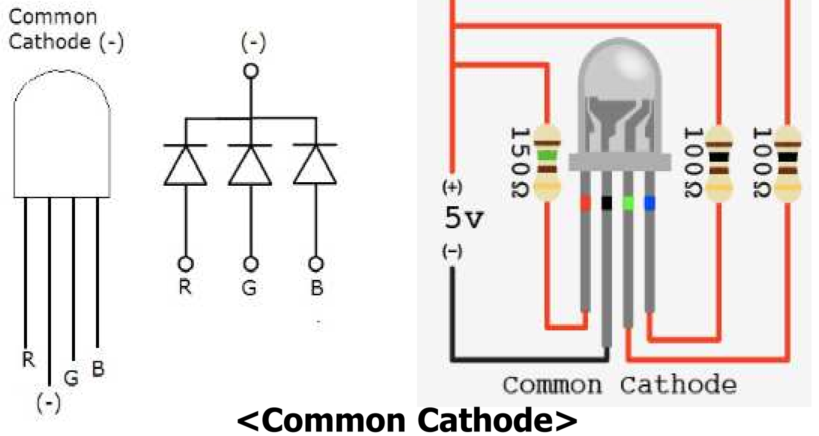
LED On/Off 및
색상의 원격 제어



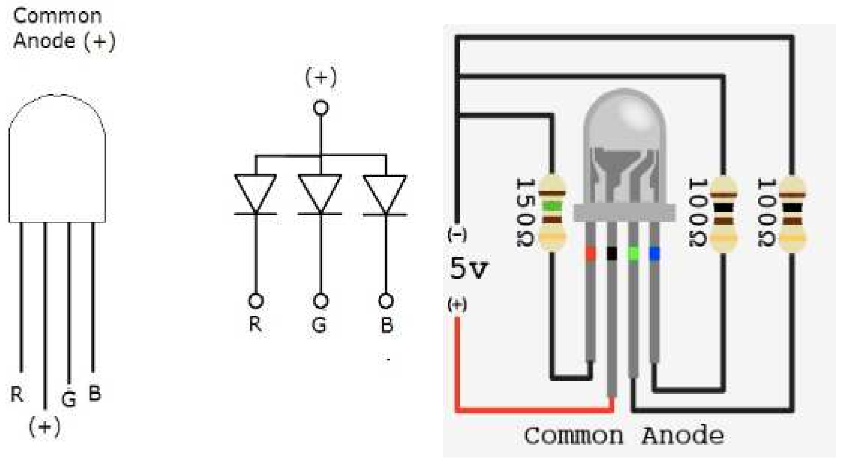
RGB Color LED

◆ Color LED





<Common Cathode>



<Common Anode>

webRC_ColorLED.py

```
# Web-based remote control of Color LED (1)

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

GPIO.setmode(GPIO.BCM)
RUNNING=True
red=13
green=19
blue=26

GPIO.setup(red, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)
GPIO.setup(blue, GPIO.OUT)

Freq=256
PWM_RED=GPIO.PWM(red, Freq)
PWM_GREEN=GPIO.PWM(green, Freq)
PWM_BLUE=GPIO.PWM(blue, Freq)

PWM_RED.start(0)
PWM_GREEN.start(0)
PWM_BLUE.start(0)
```

```
# Web-based remote control of Color LED (2)

PWM_RED.ChangeDutyCycle(50)
PWM_GREEN.ChangeDutyCycle(50)
PWM_BLUE.ChangeDutyCycle(50)

webRemoteControlServer = Flask(__name__)

@webRemoteControlServer.route("/")
def index():
    return render_template('rc_LED_index.html')

@webRemoteControlServer.route("/rc_LED", methods=["POST"])
def request_data():
    cmd=request.form.get("rc_LED")
    color, value = cmd.split()
    print("color = {}, value = {}".format(color, value))
    value = int(value)
    if color=="red":
        PWM_RED.ChangeDutyCycle(value)
    elif color=="green":
        PWM_GREEN.ChangeDutyCycle(value)
    elif color=="blue":
        PWM_BLUE.ChangeDutyCycle(value)
    return json.dumps({'status':'OK'})
```

```
# Web-based remote control of Color LED (3)

@webRemoteControlServer.route("/reset_RGB", methods=["POST"])
def reset_RGB():
    cmd=request.form.get("reset_RGB")
    red_value, green_value, blue_value = cmd.split()
    print("red = {}, green = {}, blue = {}".format(red_value, green_value, blue_value))
    PWM_RED.ChangeDutyCycle(int(red_value))
    PWM_GREEN.ChangeDutyCycle(int(green_value))
    PWM_BLUE.ChangeDutyCycle(int(blue_value))
    return json.dumps({'status':'OK'})


#------------------------------------
try:
    webRemoteControlServer.run(host="0.0.0.0", port=8080, debug=True)
finally:
  time.sleep(1)
  GPIO.cleanup()
  PWM_RED.stop()
  PWM_GREEN.stop()
  PWM_BLUE.stop()
  print("Terminating remote control of Color LED")
```

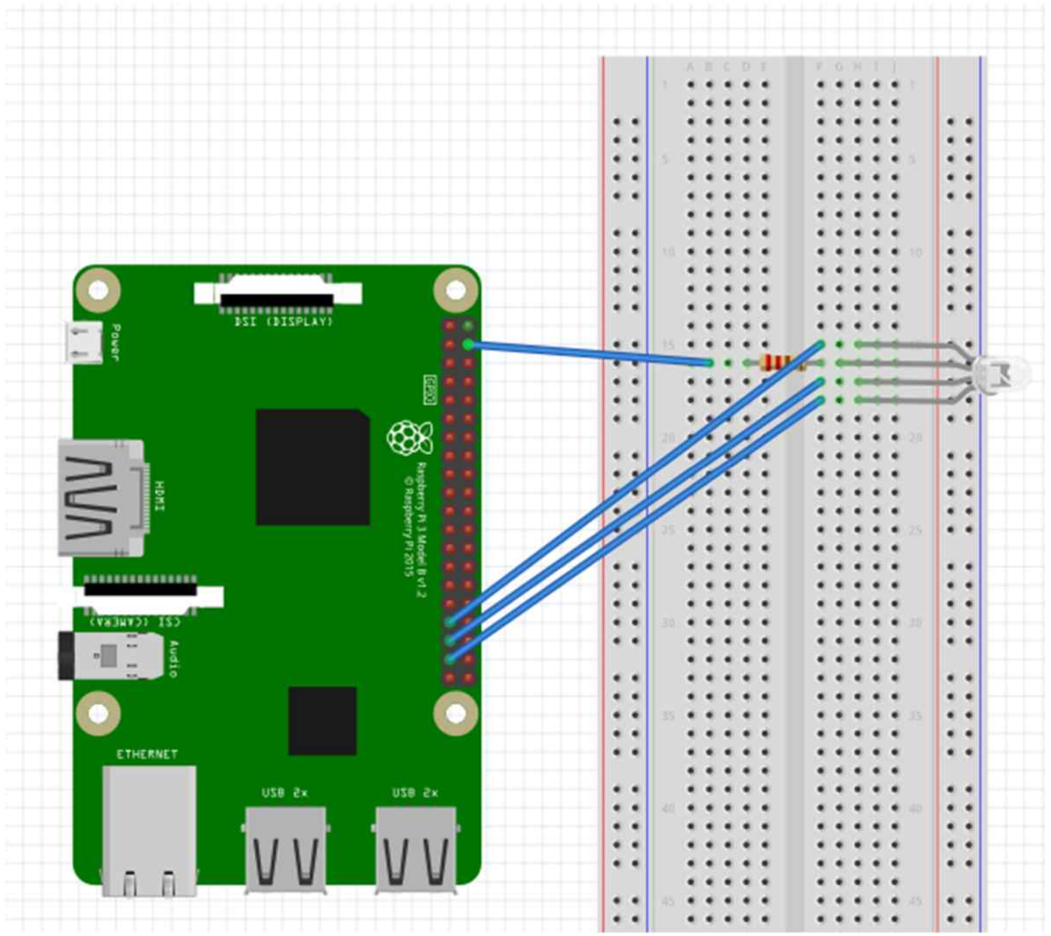
./templates/rc_LED_index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title> Web-based Remote Control of Color LED </title>
  </head>
  <body>
   <table style = "width:100%; text-align:center;">
   <H2> Web-based Remote Control of Color LED </H2>
   <div>
      <ul>
         <label for "Red"> Red  : </label>
         <input type="range" id="red" name="Red" min="0" max="100"
           onchange = 'red_change(this)'>
         <label for="value"> value ( <span id="RGB_red_value"> 0 </span> ) </label>
      </ul>
      <ul>
         <label for "Green"> Green : </label>
         <input type="range" id="green" name="Green" min="0" max="100"
           onchange ='green_change(this)'>
          <label for="value"> value ( <span id="RGB_green_value"> 0 </span> ) </label>
      </ul>
      <ul>
         <label for "Blue"> Blue  : </label>
         <input type="range" id="blue" name="Blue" min="0" max="100"
           onchange='blue_change(this)'>
         <label for="value"> value ( <span id="RGB_blue_value"> 0 </span> ) </label>
      </ul>
      <ul>
         <input type="button" value="Reset LED Color" onCLick="reset_RGB();">
      </ul>
     </div>
    </table>
```

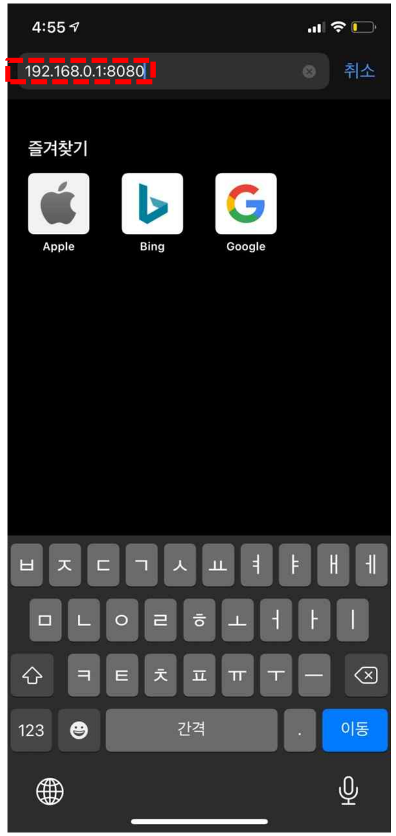
```
<script type = "text/javascript">
  function red_change(obj) {
     document.getElementById('RGB_red_value').innerHTML=obj.value;
     var request = new XMLHttpRequest ();
     var cmd = “rc_LED=red "+obj.value;
     request.open ("POST", "/rc_LED", false);
     request.setRequestHeader ("Content-Type", "application/x-www-form-urlencoded");
     request.send (cmd);
  }

  function green_change(obj) {
    document.getElementById('RGB_green_value').innerHTML=obj.value;
    var request = new XMLHttpRequest ();
    var cmd = "rc_LED=green "+obj.value;
    request.open ("POST", "/rc_LED", false);
    request.setRequestHeader ("Content-Type", "application/x-www-form-urlencoded");
    request.send (cmd);
  }

  function blue_change(obj) {
    document.getElementById('RGB_blue_value').innerHTML=obj.value;
    var request = new XMLHttpRequest ();
    var cmd = "rc_LED=blue "+obj.value;
    request.open ("POST", "/rc_LED", false);
    request.setRequestHeader ("Content-Type", "application/x-www-form-urlencoded");
    request.send (cmd);
  }
```

```
    function reset_RGB()
    {
      document.getElementById('RGB_red_value').innerHTML=0
      document.getElementById('RGB_green_value').innerHTML=0
      document.getElementById('RGB_blue_value').innerHTML=0
      var request = new XMLHttpRequest ();
      var cmd = "reset_RGB=0 0 0";
      request.open ("POST", "/reset_RGB", false);
      request.setRequestHeader ("Content-Type", "application/x-www-form-urlencoded");
      request.send (cmd);
    }
  </script>
  </body>
</html>
```

Web기반 Color LED의 원격 제어 회로 구성



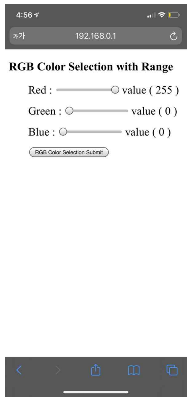
web에서 Color LED 제어 결과(1)


4:55

192.168.0.1:8080 취소

즐겨찾기

192.168.0.1:8080 접속

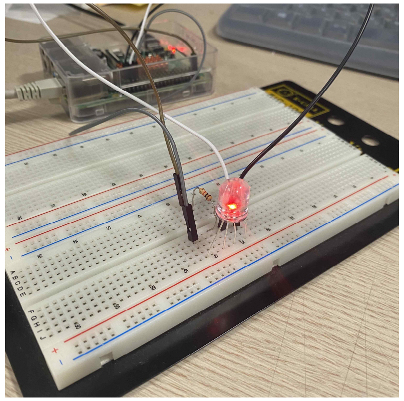


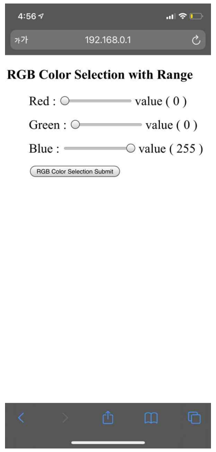
RGB Color Selection with Range

Red : value (255)

Green : value (0)

Blue : value (0)



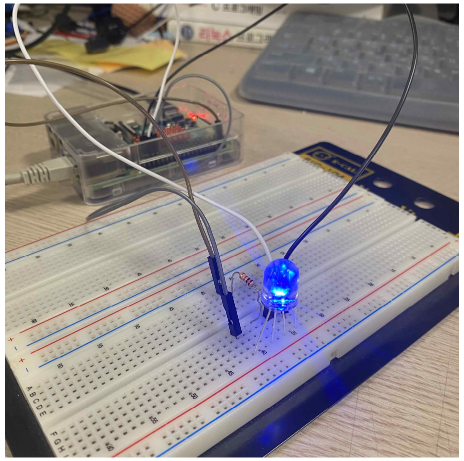


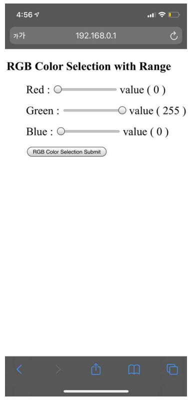
RGB Color Selection with Range

Red : value (0)

Green : value (0)

Blue : value (255)



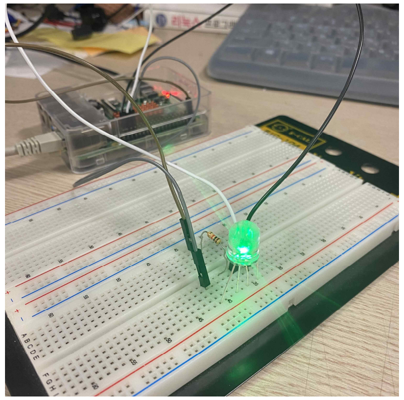


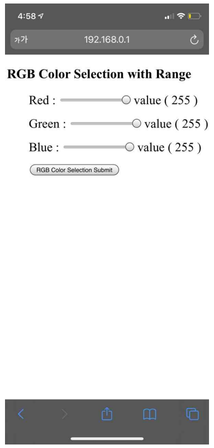
RGB Color Selection with Range

Red : value (0)

Green : value (255)

Blue : value (0)



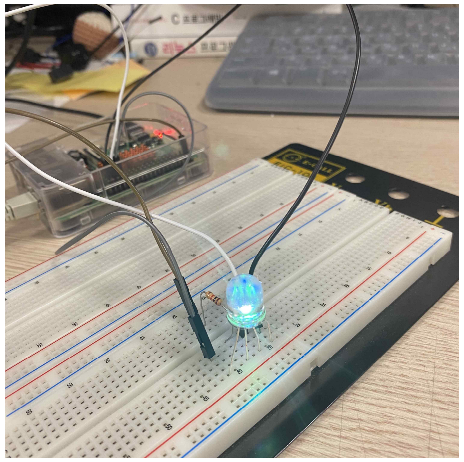


RGB Color Selection with Range

Red : value (255)

Green : value (255)

Blue : value (255)



References

<Web server>

[1] Python(Flask)을 이용한 Raspberry Pi Webserver, https://cho-raspberry.blogspot.com/p/python-web-server.html.

[2] gpiozero 이용 LED on/off, https://blog.naver.com/emperonics/221831160948.

[3] Python을 사용하여 간단한 웹서버 구축 – Simple Web Server, https://webisfree.com/2019-11-19/python-%EC%82%AC%EC%9A%A9%ED%95%98%EC%97%AC-%EA%B0%84%EB%8B%A8%ED%95%9C-%EC%9B%B9%EC%84%9C%EB%B2%84-%EA%B5%AC%EC%B6%95%ED%95%98%EA%B8%B0-simple-web-server.

[4] 로컬 테스트 서버 설치하기, https://developer.mozilla.org/ko/docs/Learn/Common_questions/set_up_a_local_testing_server

[5] Python 예제: Python 서버 코드(server.py), https://docs.aws.amazon.com/ko_kr/polly/latest/dg/example-Python-server-code.html.

[6] bottle web server, http://zetcode.com/python/bottle/.

[7] bottle: Python Web Framework, https://bottlepy.org/docs/dev/.

[8] http://jun.hansung.ac.kr/CWP/htmls/HTML%20Input%20Types.html.

[9] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/range

[10] HTML tutorial, https://www.w3schools.com/html/default.asp.

[11] Creating a Range Slider, https://www.w3schools.com/howto/howto_js_rangeslider.asp.

Homework 16

Homework 16

16.1 Web 서버 기반 원격 제어 기능 구현

- tkinter를 사용하여 붉은색 공이 canvas위에 이동하는 그래픽 에니메이션을 구현하라.
- 이 그래픽 에니메이션 기능을 web server에 연결하고, 본인의 스마트 폰을 사용하여 공을 이동시키는 간단한 원격 제어 기능을 구현하라.

16.2 Web 서버 기반 원격 스마트 조명 제어 기능 구현

- Raspberry Pi에 color LED를 접속하고, 파이썬 프로그램으로 색상 및 밝기를 조절할 수 있도록 구현하라.
- Raspberry Pi에 무선 LAN WiFi AP 기능을 구성하라.
- Raspberry Pi에 Web server 기능을 구성하라.
- Raspberry Pi의 color LED 색상 및 밝기 조절 프로그램과 web server을 소켓 통신으로 연결하도록 하라.
- 본인의 스마트 폰을 사용하여 color LED의 색상 및 밝기 조절 기능을 수행하도록 구성하고, 기능을 확인하라.