컴퓨터네트워크

- 09. Philips Hue(2)-

제 출 일	2018.11.14
학 과	컴퓨터공학과
학 번	201402391
이 름	이 병 만

※ 과제 목표 및 해결 방법

· 과제 목표

- IRC 채팅 봇을 만들어 Hue를 제어한다.
- DHCP 동작 방식을 이해한다.
- DHCP를 이용한 스마트 홈 구현

· 해결 방법

1. bot

```
from socket import *
from phue import Bridge

port_num = 6667
servername = 'chat.freenode.net'

sock = socket(AF_INET, SOCK_STREAM)
sock.connect((servername, port_num))
b = Bridge('192.168.1.139')
b.connect()
lights = b.lights
```

[그림1]

➔ 구현에 필요한 모듈을 import한다. philips hue를 사용하기 위해서 모듈을 추가해주었다. bridge로 hue를 제어할 수 있게 IP주소로 연결해주었다.

```
def hue_set(num, status, light, color_x, color_y):
    num = int(num)
    light = int(light)
    color_x = float(color_x)
    color_y = float(color_y)

    if num == 1:
        if status == 'on':
            lights[0].on = True
            lights[0].brightness = light
            lights[0].xy = [color_x, color_y]
        else:
            lights[0].on = False
    if num == 2:
        if status == 'on':
            lights[1].on = True
            lights[1].brightness = light
            lights[1].xy = [color_x, color_y]
        else:
            lights[1].on = False
    if num == 3:
        if status == 'on':
            lights[2].on = True
            lights[2].brightness = light
            lights[2].xy = [color_x, color_y]
        else:
            lights[2].on = False
```

[그림2]

➔ hue를 제어하는 함수이다. 인자로 받은 hue의 번호에 따라서 켰을 때 어떤 밝기로 무슨 색으로 키는지, 아니면 hue를 끄는지 제어하는 함수이다.

```
sock.send("NICK U201402391\r\n".encode())
sock.send("USER U201402391 U201402391 U201402391 :cnu bot\r\n".encode())
sock.send("JOIN #CNU\r\n".encode())

while 1:
    text = sock.recv(4096)
    text = text.decode()
    parse_text = text.split(' ')
    if text.find(':U201402391_!')!= -1:
        if parse_text[6] == 'on' :
            print(parse_text[4])
            print(parse_text)
            hue_set(parse_text[4], parse_text[6], parse_text[7], parse_text[8], parse_text[9].split('\r\n')[0])
        else:
            hue_set(parse_text[4], parse_text[6], 0, 0, 0)

    print(text)

    if text.find("JOIN")!= -1:
        sock.send(('PRIVMSG #CNU :HELLO [' + text[1:11]+']\r\n').encode())

sock.close()
```

[그림3]

➔ 사용자들이 접속했을 때마다 HELLO를 출력해주는 봇을 구현했었는데 채팅방에 hue를 제어하는 명령어를 입력하면 그 입력한 명령어를 해당 반복문에서 데이터를 읽어서 해당 학생의 학번이면 명령어를 파싱해서 hue를 제어하는 함수에 인자값으로 전달하는 것이다.

hue를 제어하는 명령어의 예는 다음과 같다.

ex) hue 1 set on 110 0.1 0.4 (on)

ex) hue 2 set off (off)

2. DHCP

```
import socket
import struct
from phue import Bridge
import os

hostname = '192.168.0.153'
b = Bridge('192.168.0.150')
b.connect()
lights = b.lights
```

[그림4]

➔ 구현에 필요한 모듈을 import한다. philips hue를 사용하기 위해서 모듈을 추가해주었다. bridge로 hue를 제어할 수 있게 IP주소로 연결해주었다. 위와 아이피가 다른 이유는 2번을 진행 중 다른 공유기로 교체했기 때문이다.

```
def main():
    raw_socket = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0003))
    while True:
        recv_packet = raw_socket.recvfrom(5000)
        ethernet_protocol = struct.unpack('!6s6sH', (recv_packet[0])[:14])[2]

        if ethernet_protocol == 0x800:
            ip_protocol = struct.unpack('!BBHHHBBH4s4s', recv_packet[0][14:34])[6]

            if ip_protocol == 17: # UDP if 6 TCP
                udp_src_port = struct.unpack('!H', (recv_packet[0])[34:34+2])[0]

                if udp_src_port == 68: # (Server -> Client)
                    if (str(recv_packet[0][0:14])).find(r'xf8\xe6\x1a\xc8\x8eq'):
                        lights[0].on = True
                        lights[1].on = True
                        lights[2].on = True
                        print("Ethernet Header : ", recv_packet[0][0:14])
                        print("IPv4 Header : ", recv_packet[0][14:34])
                        print("UDP Header : ", recv_packet[0][34:42])
                        print("DHCP Data : ", recv_packet[0][42:])

                        if b'\x03=' in recv_packet[0][42:]:
                            ip = recv_packet[0][296:300]
                            ip = struct.unpack('!1B1B1B1B',ip)
                            return str(ip[0])+'.'+str(ip[1])+'.'+str(ip[2])+'.'+str(ip[3])
```

[그림5]

➔ DHCP는 해당 디바이스가 공유기(와이파이)에 접속을 하기 위해서 DHCP서버를 찾는다(Discover) 찾으면 서버 주소의 IP와 임대할 IP를 보내준다.(offer) 다시 클라이언트는 IP주소를 할당해달라고 요청한다,(request) 서버는 IP주소와 네트워크 정보를 임대해준다.(Ack) 하나의 IP를 할당받는데 다음과 같은 과정을 거친다.

소켓을 열고 데이터를 받는다 받은 데이터를 파싱한다. 이더넷 헤더, IPv4헤더, UDP헤더, DHCP data로 나눌 수 있다. 각 헤더와 데이터에서 udp_src_port가 68인 경우는 client->server로 받는 과정으로 만약 Ethernet헤더에서 제 핸드폰 MAC주소와 같으면 전구를 킨다. DHCP옵션에서 요청받은 ip주소를 가져와서 반환한다.

```
            elif udp_src_port == 67:
                print('client->server')
                print(recv_packet[0][0:14])
                print(recv_packet[0][14:34])
                print(recv_packet[0][34:42])
                print(recv_packet[0][42:])
```

[그림6]

➔ **포트가 67이면 출력하는 부분이다.**

```
if __name__ == '__main__':
    while True:
        ip_addr = main()
        while True:
            response = os.system("ping -c 3 " + ip_addr)
            if response != 0:
                lights[0].on = False
                lights[1].on = False
                lights[2].on = False
                break
```

[그림6]

➔ **반환받은 ip주소로 계속 핑을 보낸다. 보낸 핑에서 IP주소를 찾지 못하면 네트워크가 끊어진 것이므로 전구를 끈다.**

※ 실행 결과

실행 결과는 링크 첨부로 대체하였습니다.

1. https://www.youtube.com/watch?v=qTOV5UHK3Ws

2. https://www.youtube.com/watch?v=0_KEvhDLC58

다 있습니다.!!