컴퓨터 네트워크
<Philips_Hue & DHCP>

제출일자 : 15. 11. 26

201102518
최인동

1. Chat을 이용한 Hue 제어

학교 내 아이피에서는 IRC 채팅 접속이 어려워서 기존 select group chat에서 client를 수정하여 만들었습니다. bot은 채팅방에서 메시지를 받는 기능만 할 수 있으며, [비밀번호] [전구번호] [On/Off] [Bright] [Color_x] [Color_y] 형식으로 데이터를 받아 조명을 컨트롤 할 수 있습니다.

- phue_bot.py

```
#!/bin/python
import socket, select, string, sys
from phue import Bridge

b = Bridge('192.168.0.200')
b.connect()

if __name__ == "__main__":

    if(len(sys.argv) < 3) :
        print 'Usage : python telnet.py hostname port'
        sys.exit()

    host = sys.argv[1]
    port = int(sys.argv[2])

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(2)

    try :
        s.connect((host, port))
    except :
        print 'Unable to connect'
        sys.exit()

    print 'Connected to remote host. Start sending messages'

    while 1:
        socket_list = [sys.stdin, s]

        read_sockets, write_sockets, error_sockets = select.select(socket_list , [], [])
```

〈phue_bot.py - part1〉

이 부분에서는 기존 select의 client와 거의 다를 바 없습니다. 기존 기능인 채팅방의 메시지를 출력하기 위해 소켓을 연결하였으며, 메시지를 입력하는 부분을 삭제 하였습니다. 조금 다른 부분은 조명을 연결하기 위해 Bridge('ip')로 미리 connect를 해 놓았습니다.

```
for sock in read_sockets:
    if sock == s:
        data = sock.recv(4096)
        list = data.split()
        if not data :
            print '\nDisconnected from chat server'
            sys.exit()
    elif len(list) == 8:
        if list[3]=='qwer':
            s.send('pass Error')
        else:
            if int(list[4])==1:
                b.set_light(int(list[3]), 'on', True)
                print(list[3] + ' light on')
                s.send(list[3] + ' light on')
            else:
                b.set_light(int(list[3]), 'on', False)
                print(list[3] + ' light off')
                s.send(list[3] + ' light off')

            b.lights[int(list[3]) - 1].brightness = int(list[5])
            print(list[3] + ' light bright : ' + list[5])
            s.send(list[3] + ' light bright : ' + list[5])

            b.lights[int(list[3]) - 1].xy = [float(list[6]), float(list[7])]
            print(list[3] + ' light color : [' + list[6] + ', ' + list[7] + ']')
            s.send(list[3] + ' light color : [' + list[6] + ', ' + list[7] + ']')

            print(list[3] + ' light State Change')
            s.send(list[3] + ' light State Change')

    else:
        sys.stdout.write(data)
```

<phue_bot.py - part2>

이 부분에서는 받은 data를 공백단위로 split()하여서 리스트로 저장하였습니다. 이 리스트의 개수를 가지고 설정 커맨드인지 여부를 파악하였습니다. 그리고 설정 정보들을 나누어 각 조건에 맞게 검사하고, 조명들을 조절 할 수 있게 설정하였습니다. 리스트의 개수가 기존에 정해놓은 control 형식과 다르면 메시지를 출력하게 하였습니다.

```
pi@raspberrypi ~ $ python select_client.py localhost 5000
Connected to remote host. Start sending messages

<You> qwer 1 1 100 0.1 0.1

<('127.0.0.1', 36146)> 1 light on
<('127.0.0.1', 36146)> 1 light bright : 100
<('127.0.0.1', 36146)> 1 light color : [0.1, 0.1]
<('127.0.0.1', 36146)> 1 light State Change
<You> qwer 2 1 100 0.3 0.8

<('127.0.0.1', 36146)> 2 light on
<('127.0.0.1', 36146)> 2 light bright : 100
<('127.0.0.1', 36146)> 2 light color : [0.3, 0.8]
<('127.0.0.1', 36146)> 2 light State Change
<You> qwer 3 1 100 0.7 0.3

<('127.0.0.1', 36146)> 3 light on
<('127.0.0.1', 36146)> 3 light bright : 100
<('127.0.0.1', 36146)> 3 light color : [0.7, 0.3]
<('127.0.0.1', 36146)> 3 light State Change
<You>
```

```
pi@raspberrypi ~ $ python phue_bot.py localhost 5000
Connected to remote host. Start sending messages
[127.0.0.1:36147] entered room
1 light on
1 light bright : 100
1 light color : [0.1, 0.1]
1 light State Change
2 light on
2 light bright : 100
2 light color : [0.3, 0.8]
2 light State Change
3 light on
3 light bright : 100
3 light color : [0.7, 0.3]
3 light State Change
```

<phue_bot과 client의 통신>

〈실제 전구가 켜진 사진 (전구 1, 전구 2, 전구 3)〉

2. DHCP 모니터링을 통한 스마트홈 구현

이번 DHCP 모니터링을 통한 구현에서는 먼저 DHCP 패킷들을 잡아보고 필요한 헤더들의 위치를 알아보기로 하였습니다.

Source	Destination	Protocol	Length	Info
192.168.0.101	255.255.255.255	DHCP	342	DHCP Inform
192.168.0.101	192.168.0.1	DHCP	342	DHCP Release
0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover
192.168.0.1	192.168.0.101	DHCP	590	DHCP Offer
0.0.0.0	255.255.255.255	DHCP	354	DHCP Request
192.168.0.1	192.168.0.101	DHCP	590	DHCP ACK
192.168.0.101	255.255.255.255	DHCP	342	DHCP Inform
192.168.0.101	255.255.255.255	DHCP	342	DHCP Inform

〈DHCP 패킷 - wireshark〉

노트북에 wireshark를 켜놓고, 설치한 공유기로 와이파이를 연결하면서 DHCP들을 잡아 봤습니다. 실습 pt에 있는 Discover, Offer, Request, Ack가 순서대로 잡히는 것을 볼 수 있고, 그것 말고도 Inform도 잡히는 것을 볼 수 있습니다. Inform에서는 현재 연결된 AP에 일정 주기로 보내는 패킷 이었습니다.
AP에 와이파이를 잡을 때 새로 잡는 와이파이가 아니면 주로 Request 패킷이 잡히는 것을 확인해서, 그 패킷만을 구분하기로 하였습니다.

- dhcp.py

```
#!/bin/python
import socket, struct, commands, time
from phue import Bridge
s = socket.socket( socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0003) )

while(True):
    packet = s.recvfrom(65565)

    eth_protocol = struct.unpack('!6s6sH', (packet[0])[:14])
    if (eth_protocol[2] == 0x0800) :
        ip_protocol = struct.unpack('!B', (packet[0])[23])
        ip_src = struct.unpack('!4s', (packet[0])[26:30])
        ip_dst = struct.unpack('!4s', (packet[0])[30:34])

        if (ip_protocol[0] == 17 and socket.inet_ntoa(ip_src[0]) == '0.0.0.0' and socket.inet_ntoa(ip_dst[0])
            udp_protocol_port = struct.unpack('!HH', (packet[0])[34:38])
            dhcp_type = struct.unpack('!B', (packet[0])[284])

            if (udp_protocol_port[0] == 68 and udp_protocol_port[1] == 67 and dhcp_type[0] == 3) :
                dhcp_mac = struct.unpack('!6s', (packet[0])[288:294])
                dhcp_ip = struct.unpack('!4s', (packet[0])[296:300])
                print socket.inet_ntoa(dhcp_ip[0]) + ' ' + dhcp_mac[0].encode('hex')
                print 'Connecting...'
```

〈dhcp.py - part1〉

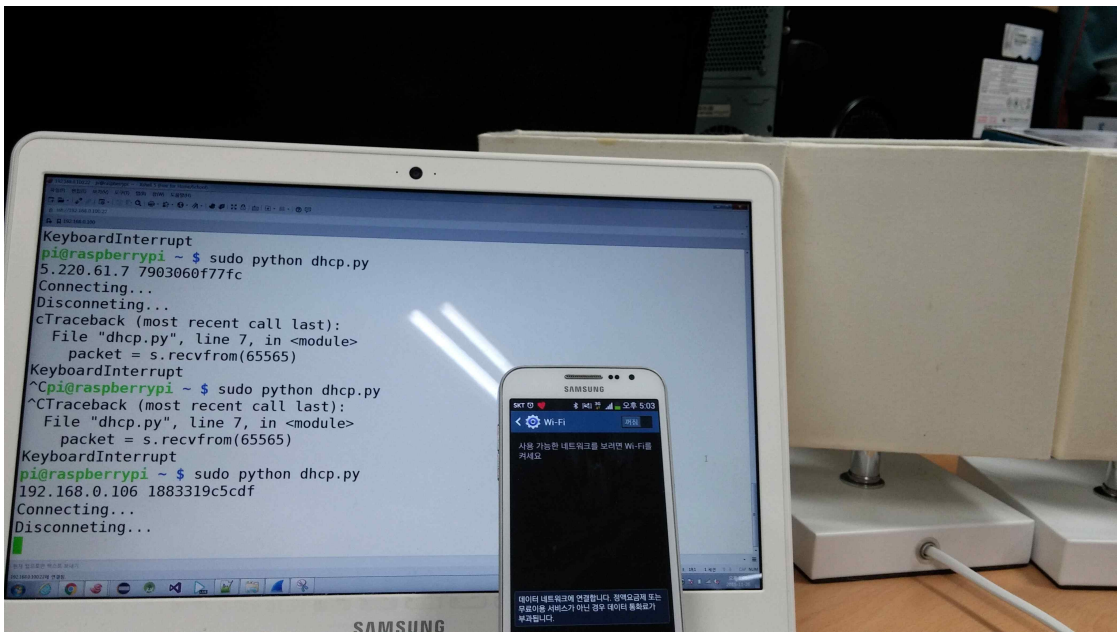
저는 Request 패킷의 특징들을 가지고 packet을 구분 하였습니다.
Ethernet Protocol(IP = 0x0800),
IP Protocol(UDP = 17, src = 0.0.0.0, dst = 255.255.255.255),
udp protocol(src port = 68, dst port = 67),
bootstrap protocol(DHCP Message Type = Request(3))
조건문을 통해서 이 조건들을 검사하고 정상적인 Request 패킷이 오면 할당한 IP와 MAC주소를 출력하게 하였습니다.

```
b = Bridge('192.168.0.200')
b.connect()
b.set_light(1, 'on', True)
b.set_light(2, 'on', True)
b.set_light(3, 'on', True)

b.lights[0].brightness = 100
b.lights[1].brightness = 100
b.lights[2].brightness = 100

b.lights[0].xy = [0.2, 0.05]
b.lights[1].xy = [0.2, 0.05]
b.lights[2].xy = [0.2, 0.05]

time.sleep(2)

while(True):
    ping_com = commands.getoutput('ping -c 1 -w 1 ' + socket.inet_ntoa(dhcp_ip[0]))
    if ping_com.find('0 received') != -1:
        print 'Disconneting...'
        b.set_light(1, 'on', False)
        b.set_light(2, 'on', False)
        b.set_light(3, 'on', False)
        break
```

〈dhcp.py - part2〉

조건이 맞을 경우 Bridge로 연결하고, 조명들을 원하는 옵션대로 켜줍니다. 그리고 반복문을 통해 해당 아이피로 ping을 보내줍니다. ping이 끊겼을 경우(연결이 끊겼을 경우), ping의 출력화면에서 '0 received'를 찾아서 전등을 끄게 하고 반복문을 빠져나옵니다.

〈와이파이가 연결됬을 경우〉



〈와이파이가 끊겼을 경우〉

4. 후기

이번 과제에서는 눈에 직접 보이는 전등을 컨트롤 할 수 있어서 다른 과제들보다 완성했을 때 뿌듯함이 좀 더 컸습니다. DHCP를 공부하느라 시간을 많이 썼고, 다른 것보다 패킷들의 주소들을 찾아 unpack하는데 많이 어려 웠습니다.