캡스톤 디자인 결과보고서

Ⅰ.서론

1. 작품 과제의 개요

우리가 보통 날씨를 확인할 때는 기상청 일기예보를 통해 날씨를 확인하지만, 실내공간은 기상청의 예측치와는 다른 경우가 있다. 실제로 우리가 생활하는 공간은 실외보다는 실내이기 때문에 이런 경우 실내환경을 정확히 알고 공기청정기나 가습기, 에어컨 등을 작동하고 싶을 때가 있다. 따라서 이와 같은 문제를 해결하기 위해서 실제 실내 환경을 측정하는 장치를 만들어 보자고 생각을 해보았다. 그래서 직접 온도, 습도, 미세먼지, 자외선 지수를 측정하여 실제 실내환경을 알아내고 그에 맞는 반응을 할 수 있도록 만들어 보았다.

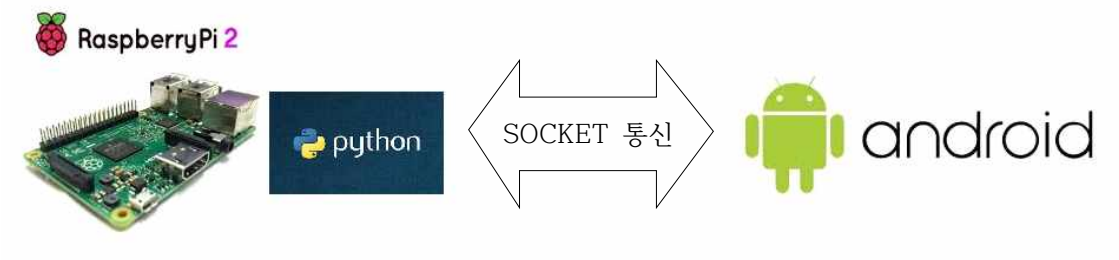
2. 기존 제품과의 비교, 차이점

기존의 스마트 홈, 스마트 오피스는 가전제품(TV, 에어컨, 냉장고 등)을 비롯해 에너지 소 비장치(수도, 전기, 냉난방 등), 보안기기(도어록, 감시카메라 등) 등 다양한 분야에서 모든 것을 통신망으로 연결해 모니터링, 제어할 수 있는 기술을 말한다. 시중의 인공지능(AI) 스피커는 사용자의 음성을 인식해 집 안의 모든 사물인터넷(IoT) 기기를 연결하고 사용자의 특성에 따라 자동으로 작동하거나 원격으로 조종할 수 있다. 자신의 회사나 계약된 회사의 제품에 한해서만 연결할 수 있다. 이 프로젝트는 기존 스마트 홈, 스마트 오피스 등과는 차이점이 있다. 우리가 만든 프로젝트는 220V로 연결만 할 수 있다면 기종(LG,삼성 등)에 상관없이 TV, 에어컨, 선풍기, 스탠드 등, 다양한 기종을 스마트폰을 이용해서 연결해, 제어할 수 있다. 또, 스마트 on/off기능을 이용해 설정된 값에 따라 버튼 클릭 한 번으로 자동으로 장치들을 제어할 수 있는 장점이있다. 집에 홀로 남은 아이를 위해 사용하거나, 많은 방이나 오피스를 관리해야 할 때 유용하게 사용할 수 있을 것이다.

Ⅱ.본론

1.과제 수행방법

OS를 Rasbian으로 사용하는 라즈베리파이(Raspberry Pi 2 Model B)사용한다. uv센서, 미세먼지 센서, 온습도 센서, 스텝모터, 가습기 모듈, LCD센서, 릴레이 모듈을 결선을 통해 라즈베리파이와 연결하고, 파이썬으로 각 센서와 모듈이 동작하게 코딩해 연동한다. 220V장치를 제어하기 위해서 릴레이 모듈과 220V 전선을 연결하는 작업도 수행한다.



<그림 1>

프로젝트는 Server-Client 구조를 사용한다. 라즈베리파이에서 파이썬 언어를 사용해 서버를 구축하고, 안드로이드를 클라이언트로 사용해 소켓 통신을 수행한다.

2.제작과정

날짜	제작 과정
2019,04.22~04.29	1. 프로젝트 설계 및 구성 2. 사용할 센서확인, 재료점검
2019.04.29.~05.06	1. 재료준비의 부족으로 프로젝트 진행이 안됨 2. 아이디어 추가 및 구체화
2019.05.08.~05.13	1. 재료 준비 및 프로젝트 구성 2. 센서활용한 하드웨어 구성 및 라즈베리 결선
2019.05.15.~05.22	1. 안드로이드 활용한 클라이언트-서버 조성 2. 소켓통신 동작 확인 3. 필요한 하드웨어 구성품 주문
2019.05.22.~05.29	1. 알고리즘 최종 확인 2. 프로젝트 동작 확인 및 즉각적 피드백 3. 결과물 에러 확인 및 업데이트
2019.05.29.~06.09	1. 기대효과 논의 2. 부족한 부분(코드,외관) 보완 및 프로젝트 완성 3. 발표준비 및 발표

3.소켓통신

라즈베리파이에서 파이썬으로 서버를 구축했다. Socket을 생성하고, host주소와 Port번호를 찾아 소켓에 bind한다. 서버에서는 무한 대기하고 있다가 클라이언트에서의 통신 요청이 오면 요청을 받고, recv() 메서드로 데이터를 받는다. 이 데이터를 쪼갠 후 프로토콜에 따라서 명령을 수행한다. 클라이언트는 AsyncTask활용해 UI스레드와 메인 스레드의 커뮤니케이션을 원활하게 하였다. 클라이언트에서는 소켓을 만든 후 서버에서 바인드한 호스트주소와 포트번호로 데이터를 outputStream으로 송신한다. 그리고, inputStream으로 데이터를 수신한다. 클라이언트 내에도 프로토콜이 존재하는데, 이후 내용에 설명할 기능들을 처리하기 위

한 프로토콜이다.

```
HOST = "192.168.0.26"
PORT = 12346
print(HOST)
a = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Socket created')
a.bind((HOST,PORT))
print('Socket bind complete')
a.listen(1)
print('Socket now listening')
```

<코드 1> 서버 소켓 바인드

```
while True:

    conn, addr = a.accept()
    print("Connected by ",addr)

    #data susin
    data = conn.recv(1024)
    data = data.decode("utf8").strip()
    if not data: break
    print("Received: " + data)

    threading._start_new_thread(do_some_stuffs_with_input,(data,))
    time.sleep(1)

    #susin data pi control
    res = do_some_stuffs_with_input(data)
```

<코드 2>서버 소켓 수신

```
MyClientTask myClientTask = new MyClientTask("192.168.0.26", 12346 , "loading...");
myClientTask.execute();
```

<코드 3>클라이언트 AsyncTask상속받은 클래스 실행

```
@Override
protected Void doInBackground(Void... arg0) {

    Socket socket = null;
    myMessage = myMessage.toString();
    try {
        socket = new Socket(dstAddress, dstPort);
        //송신
        OutputStream out = socket.getOutputStream();
        System.out.println();
        out.write(myMessage.getBytes());

        //수신
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(1024);
        byte[] buffer = new byte[1024];
        int bytesRead;
        InputStream inputStream = socket.getInputStream();

        while ((bytesRead = inputStream.read(buffer)) != -1){
            byteArrayOutputStream.write(buffer, 0, bytesRead);
            response += byteArrayOutputStream.toString("UTF-8");

          protocol(response);

        }
```

<코드 4>클라이언트 소켓 송,수신

4. 실시간 확인 기능

실내의 미세먼지값, 온도, 습도, uv값, 날짜와 시간을 실시간으로 확인이 가능하다. <그림 2>의 중간 부분에는 센서에서 측정한 값을 실시간으로 받아와 출력해 주고 있다. 사용자는 바깥 날씨와는 다른 실내환경을 확인할 수 있다.

<그림 2>

날씨누리 본체에서도 값을 확인할 수 있다. <코드 1>의 내용은 LcdDisplay에 실내의 환경값을 출력하는 코드이다. 코드 내용을 살펴보면, 변수 save는 센서에서 측정한 값들을 모아놓은 String 변수이다. 변수 save를 split() 함수를 이용해 각각의 값들을 추출해낼 수 있다. LcdDisplay는 <그림 3>을 15초간 출력하고, lcddriver.py의 lcd_clear() 함수를 사용해 기존 내용을 지운 후 <그림 4>를 출력해 작은 화면에 모든 정보를 출력할 수 있도록 활용했다. 디스플레이는 소켓 통신과는 병행으로 동작해야 되기 때문에 do_Display() 함수는 <코드 2>에서 볼 수 있듯이 스레딩으로 처리해 주었다.

<그림 3>

<그림 4>

```
def do_display(display_string):
    global save
    global display1


    try:
        newList = save.split(':')
        if(len(newList)>8):

            new1List = newList[4].split('H')
            new2List = newList[5].split('P')
            new3List = newList[6].split('P')
            new4List = newList[7].split('u')

            while True:
                display1.lcd_display_string("Temp:"+new1List[0]+"uv:"+newList[8],1)
                display1.lcd_display_string("Humidity : "+new2List[0]+"  ",2)
                time.sleep(15)
                display1.lcd_clear()                            # Clear the display of any data
                time.sleep(1)
                display1.lcd_display_string("PM2.5 : "+new3List[0],1)
                display1.lcd_display_string("PM10  : "+new4List[0],2)
                time.sleep(15)
                display1.lcd_clear()                            # Clear the display of any data
                time.sleep(2)
        else:
            while True:
                display1.lcd_clear()                            # Clear the display of any data
                time.sleep(1)
    except KeyboardInterrupt: # If there is a KeyboardInterrupt (when you press ctrl+c), exit the program and cleanup
        print("Cleaning up!")
        display1.lcd_clear()
```

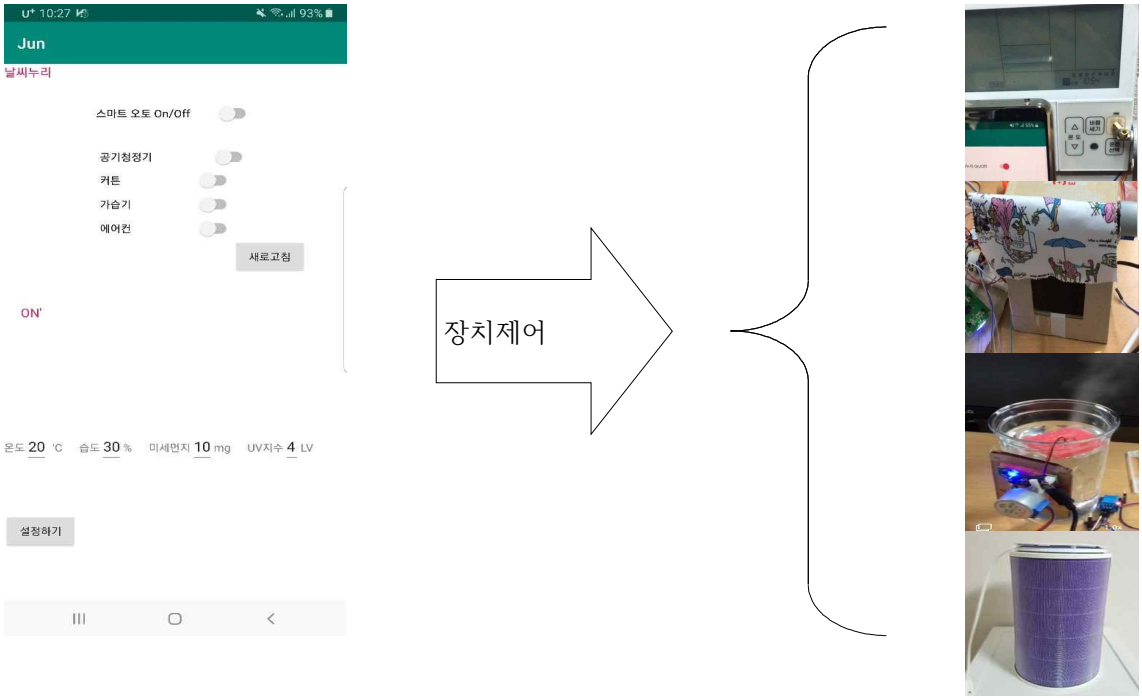
<코드 1>display

```
threading._start_new_thread(do_display,(res,))
```

<코드 2>display 스레드 설정

5. 원격 장치제어

공기청정기, 커튼, 가습기, 에어컨 4가지 장치에 대해서 스마트폰을 이용해 원격으로 장치를 제어할 수 있다. 현재 장치가 작동중 이라면 switch slide가 ON으로 표시돼 간편하게 확인할 수 있다. 장치를 끄고 싶다면, switch slide를 OFF로 바꾸고 끄면 된다. <코드 7>은 서버에서 수신한 데이터를 프로토콜에 따라 나눈 모습이다.



<그림 5>

"loading..."은 안드로이드에서 switch slide 어떤 것을 ON하든간에 작동하는데, 센서에서 측정한 값들을 save 스트링 변수에 담아 화면에 출력하거나, 스마트 on/off기능에 활용된다. "off" 에서는 GPIO 23번과 18번을 TRUE로 바꾸는데, 릴레이 모듈과 연결되어 220v 전류가 끊어지게끔 코딩하였다(반대로 FALSE일 때 전류가 연결됨). relay에 220v전선을 직접 연결해 사용하였다. "cuttonOn"과 "cuttonOff"는 각각 파일에 스텝모터의 동작에 대해 코딩했다. 각각 파일의 스텝모터는 서로 반대 방향으로 움직인다. "airCleanerOn"과 "airCLeanerOff"는 릴레이 모듈로 장치를 제어한다. "humidyOn", "humidyOff", "airConditionOn", "airConditionOff"는 <그림 6>, <그림 7>에서 보는 '스마트 스위치'를 활용한다. 스마트 스위치는 스텝모터를 활용한 스위치로, 릴레이 모듈을 통한 전류 제어만으로 켜지지 않고, 전원버튼을 눌러야 하는 제품을 제어하기 위해 만들었다.

<코드 8>에서 do_some_stuffs_with_input(input_string)메서드는 소켓 통신과는 별개로 장치제어하는 프로토콜을 처리하는 메서드이기 때문에 스레드로 처리해 주었다.

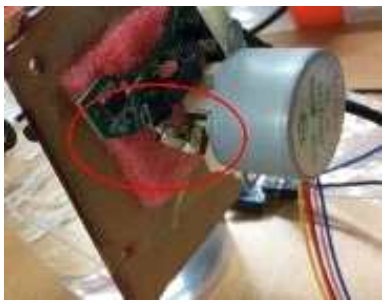
```
def do_some_stuffs_with_input(input_string):
    global save
    if input_string == "loading...":
        dustValue = []
        dustValue = dustFile.values1
        result = instance.read()

        if result.is_valid():
            temp = str(result.temperature)
            hum = str(result.humidity)
            uvValue = str(uv.analog(0))
            input_string = "Today : " + str(datetime.datetime.now())+"\n"
            +" Temperature :" + temp + "'C  Humidity :" + hum +"%\n"
            +"PM2.5:"+ str(dustValue[0]) + "mg/m3  PM10:"+ str(dustValue[1])
            + "mg/m3"+"\n"+"uv :"+uvValue+"Level"
            save= input_string

    elif input_string == "off":
        GPIO.output(23,True)
        GPIO.output(18,True)
        input_string = "OFF"

    elif input_string == "airCleanerOn":
        GPIO.output(23,False)
        input_string = save+"&airCleanerOn"

    elif input_string == "airCleanerOff":
        GPIO.output(23,True)
        input_string = save+"&airCleanerOff"

    elif input_string == "cuttonOn":
        cuttonMotorOpen.openMotor(ControlPin2,seq)
        input_string = save+"&CuttonOn"

    elif input_string == "cuttonOff":

        cuttonMotorClose.closeMotor(ControlPin,seq)
        input_string = save+"&cuttonOff"

    elif input_string == "humidyOn":
        GPIO.output(18,False)
        humidyOn.humidyMotor1(humidyPin,humidyPin2,seq)
```

<코드 7>서버 프로토콜

```
threading._start_new_thread(do_some_stuffs_with_input,(data,))
```

<코드 8>스레드 지정

<그림 6>



<그림 7>

<코드 9>, <코드 10>, <코드 11> 은 스마트 스위치와 같은 모터를 동작시키는 코드이다. <코드 9>에서 GPIO핀번호를 지정한다. <코드 10>에서 GPIO핀을 setup하고 스텝모터 seq 4차원 배열을 정의한다. <코드 11>에서 모터 돌아갈 바퀴수를 지정할 수 있다.

```
ControlPin = [5,22,17,4]
```

<코드 9>GPIO 핀번호

```
for pin in ControlPin:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, 0)

seq=[[1,0,0,0],
     [1,1,0,0],
     [0,1,0,0],
     [0,1,1,0],
     [0,0,1,0],
     [0,0,1,1],
     [0,0,0,1],
     [1,0,0,1] ]
```

<코드 10>seq배열

```
def humidyMotor1(c1,c2,s):

    for i in range (24):
        for halfstep in range (8):
            for pin in range (4):
                GPIO.output(c2[pin], s[halfstep][pin])
                time.sleep(0.001)
```

<코드 11>humidyMotor 동작

6.스마트 ON/OFF

스마트 ON/OFF의 상태를 ON으로 활성화할 경우, <그림 >에서 하단의 설정된 값에 따라 각 장치들을 자동으로 ON상태로 바꿔준다. 사용자는 자신에게 가장 적합한 환경이나, 계절에 따라 쾌적하게 실내환경을 유지할 수 있는 장점이 있다.

현재 미세먼지 값은 15.7mg이고 , 현재 UV값은 4LV이다. <그림 8>의 설정된 미세먼지 값은 1mg이고, UV값은 8LV이기 때문에 공기청정기가 ON이고, 커튼이 OFF인 상태이다. <그림 9>의 설정된 미세먼지 값은 30mg, UV값은 2LV이기 때문에 공기청정기가 OFF이고, 커튼이 ON인 상태가 된다. 설정된 값이 달라지니 연결된 장치들의 상태도 달라지는 것을 확인할 수 있다. <표 1>에 정리된것과 같다.

<그림 8>

<그림 9>

비교	현재 값	<그림 8> 설정값	<그림 9> 설정값
미세먼지 값	15.7mg	1mg	30mg
공기청정기	X	ON	OFF
UV 값	4LV	8LV	2LV
커튼	X	OFF	ON

<표 1>스마트 on/off 비교

<코드 12>는 안드로이드 어플에서 설정값을 주었을 때 설정값과 현재 값을 비교한 알고리즘이다. UI를 변경하기 위해 UI스레드에 넘기는 모습도 볼 수 있다. 어플 내에서 비교를 한 후 서버에 해당하는 장치에 명령한다.

```
if (temp >= temperatureValue) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {

                airCondition.setChecked(true);

        }
    });

    MyClientTask myClientTask = new MyClientTask("192.168.0.26", 12346, "airConditionOn");
    myClientTask.execute();

if(hum <=humidyValue){


    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            humidy.setChecked(true);
        }
    });
    HumidyState = 1;
    MyClientTask myClientTask = new MyClientTask("192.168.0.26", 12346, "humidyOn");
    myClientTask.execute();
```

<코드 12> 스마트 on/off 알고리즘

III.결론

1. 기대 효과

이 프로젝트의 첫 번째 기대효과는 220V를 사용할 수 있는 가전 기기만 있다면 다양한 가전 제품들을 사용할 수 있다. 새로운 가전제품이 추가되어도 사용자가 멀티탭에 코드만 꼽아 손쉽게 연동할 수 있다. 두 번째 기대효과는 에어컨과 공기청정기, 가습기, 그리고 커튼등을 스마트폰과 연결시켜 원격으로 동작시킬 수 있는 것이다. 사용자 편리성을 제공할 것으로 기대된다. 세 번째 기대효과는 스마트 ON/OFF 기능을 이용해 사무실이 여러개인 큰 기업, 호텔, 공공기관, 학교 등에서 분산되어있는 전자제품들을 스마트하게 관리가 가능하다. 사람이 직접 전자제품을 키고 끄는 번거로움 없이 사전에 설정한 값에 따라 무인으로 전자제품을 제어 가능하다. 네 번째 기대효과는 릴레이 모듈을 활용함으로써 사용하지 않는 전자제품에 대해서는 대기전력을 발생하지 않게 해 전력을 효율적으로 관리할 수 있다.

2. 개선방향

앞서 스마트스위치는 유선으로 연결되었기 때문에 거리에 제약이 있고, 선이 외관으로

나와있어 불편함이 따른다. 하지만 무선으로 바꾸면 어떨까? 블루투스와 와이파이를 이용해서 모터를 제어할 수 있다면, 거리가 먼 전자제품을 더욱 효과적으로 제어할 수 있을 것이다. 다음 개선할 점은 스마트 on/off기능에 추가할 사항이다. 스마트 on/off기능은 설정한 값에 따라 장치를 켜는 기능인데, 설정한 값을 일정시간 유지한다면, 자동으로 장치를 종료시켜 전력을 아낄 수 있게 기능을 추가하면 좋을 것 같다. 이 부분은 소프트웨어 적인 부분으로 충분히 해결할 수 있을 것이다.

Ⅳ.부록

1. 기타 보고 사항

우리가 지원받은 총 지원금은 300,000원이 지원 되었지만, 처음에 프로젝트를 구상하고 재료를 사면서 잘못된 재료들을 구입하면서 재료 준비에 있어 어려움이 있었고 마지막 재료를 사는데 예산이 부족해 예산을 초과하게 되었다. 300,000원에서 10650원을 초과해 총 310,650원을 사용했다.

2 참고 문헌

릴레이모듈과 220V 연결 : https://redmuffler.tistory.com/422
릴레이모듈 이해 영상 : https://www.youtube.com/watch?v=fi9LJDeiNzA
LCD 센서 사용 : https://www.youtube.com/watch?v=fR5XhHYzUK0
LCD 센서 코드 : https://github.com/adafruit/Adafruit_Python_CharLCD
미세먼지 센서 사용 : https://geeksvoyage.com/raspberry%20pi/nova-sds011-for-pi/
DC 모터 사용 : https://github.com/Richard-Kirby/dc-motor
DC 모터 사용한 커튼 구상 참조 : https://blog.naver.com/project307/221066718245
UV 센서 사용 : https://github.com/idreturned1/Arduino-UVSensor
가습기 모듈 관련 영상 : https://www.youtube.com/watch?v=OrYSHA7MDT0
https://www.youtube.com/watch?v=CKNsJOtYAJ8

3. 증빙 사진

완성 작품 및 사용 센서

uv 센서

스텝 모터

릴레이모듈

온습도 센서

가습기 모듈

미세먼지 센서

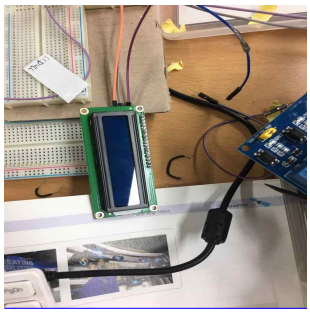
lcd 센서

구매 재료

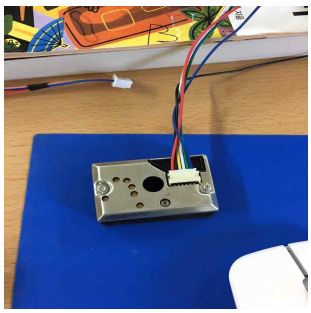
케이블

금속용 풀

DC모터



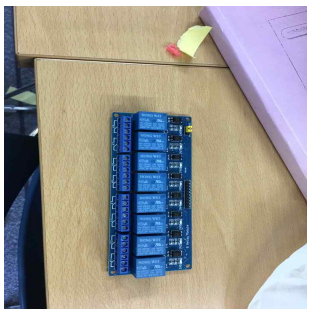
LCD 센서



미세먼지 센서

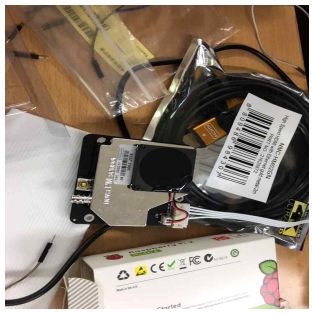
니퍼

가습기 모듈



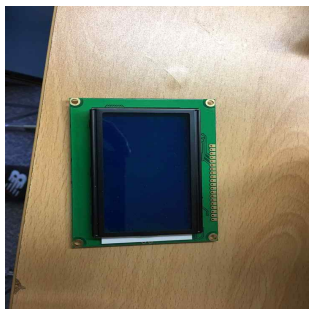
릴레이모듈

멀티탭

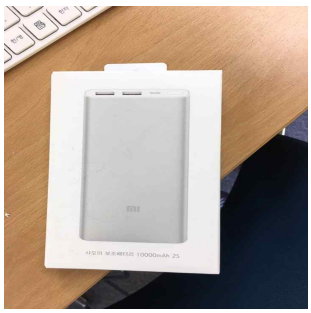


미세먼지센서

충전기



LCD 센서



보조배터리

공기청정기

1구 멀티탭