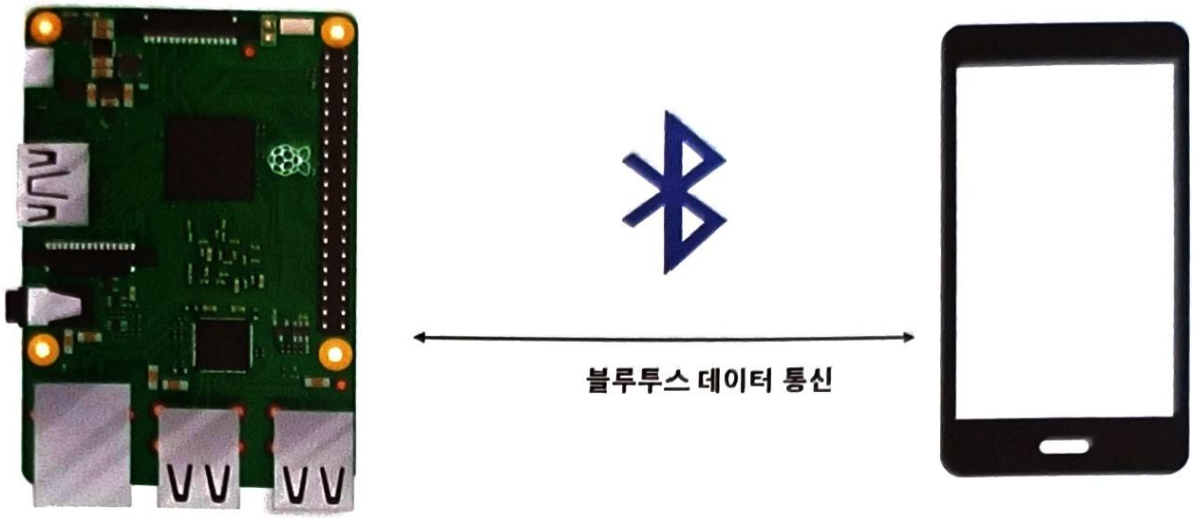
06 블루투스 통신(Bluetooth Communication)

6.1 블루투스란?

블루투스란 휴대폰, 노트북, 이어폰 등의 휴대용 기기들을 서로 연결하여 정보를 교환하는 근거리 무선 기술입니다. 블루투스의 무선 시스템은 ISM(Industrial Scientific and Medical) 주파수 대역인 2400~2483.5MHz를 사용합니다. 라즈베리파이 4는 블루투스 5를 지원합니다. 블루투스 5에는 대역폭을 희생해 도달 거리를 늘리는 방식으로 사용하거나 거리는 줄이고 대역폭을 늘리는 새로운 인터페이스가 도입되어 사용하고자 하는 애플리케이션에 따라서 저전력으로 사용할 수도 있고 전송 속도와 먼 거리를 필요로 하는 애플리케이션도 유연하게 대응이 가능하게 되었습니다.

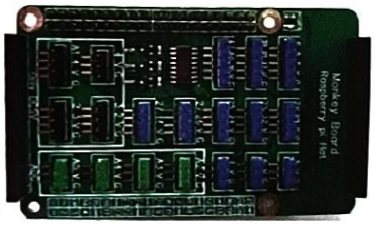
6.2 스마트폰과 블루투스 통신

안드로이드 기반의 스마트폰과 라즈베리파이와의 블루투스 연결을 하고 데이터 양방향으로 데이터 통신을 해보도록 하겠습니다.

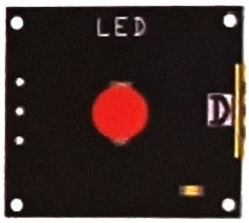


라즈베리파이 3부터 와이파이와 블루투스 기능이 기본적으로 포함되어 있습니다. 라즈베리파이 3에서는 와이파이, 블루투스 기능이 조금 불안한 부분이 있었지만 라즈베리파이 4부터는 조금 더 안정적으로 발전한 것 같습니다.

실험에 필요한 준비물들



라즈베리파이 GPIO HAT



LED Red

(1) 라즈베리파이 블루투스 장치

라즈베리파이 4의 불루투스 장치는 내부적으로 CPU와 UART로 통신할 수 있도록 연결되어 있습니다.

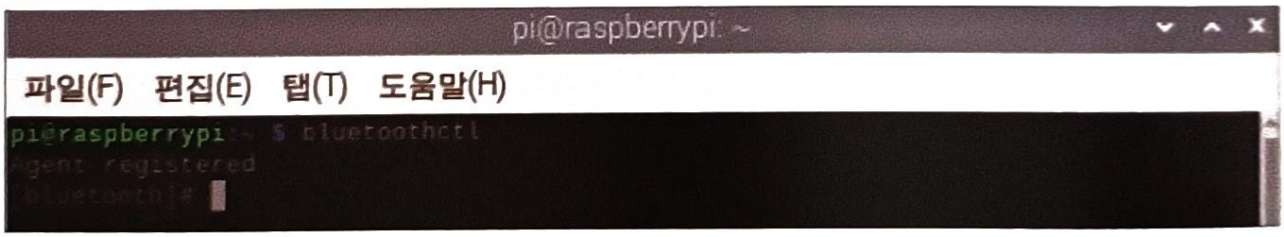
UART 채널	시리얼 포트	장치 이름	용도
UART1	serial1	/dev/ttyAMA0 or /dev/serial1	블루투스

그래서 블루투스 장치를 사용하는 방법이 앞에서 배웠던 시리얼 장치의 사용 방법과 거의 동일하게 사용이 가능합니다. 하지만 시리얼 장치와 다른 점은 무선으로 연결할 장치와 페어링이라는 과정과 연결 과정이 필요합니다. 한 가지 더 다른 점은 시리얼 장치 이름을 "/dev/serial1"으로 사용하지 못하고 "/dev/rfcomm0"라는 이름으로 사용합니다.

(2) 블루투스 장치 페어링

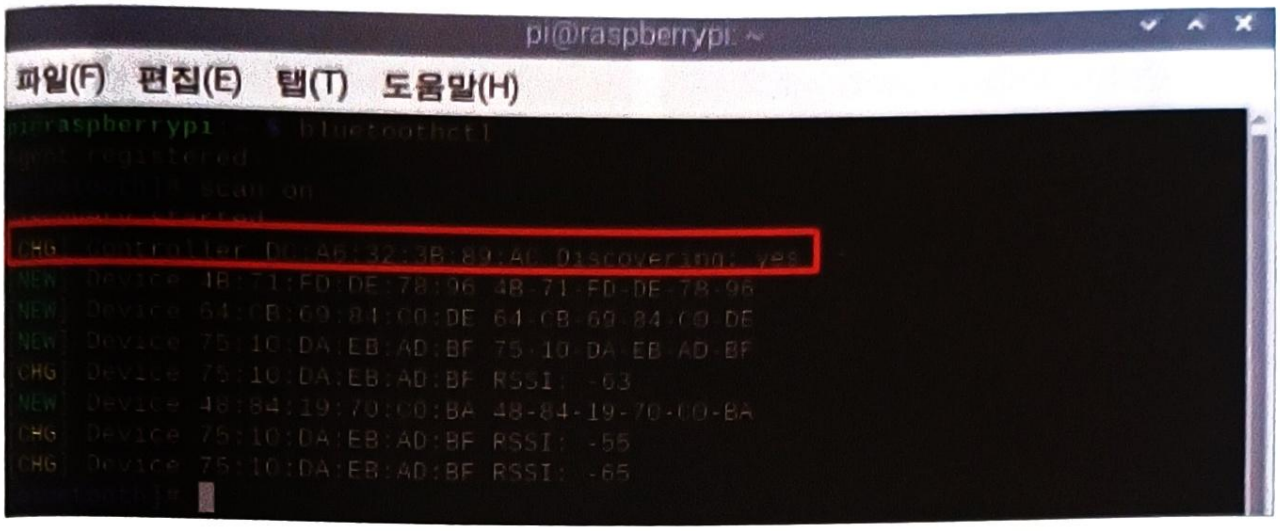
무선으로 라즈베리파이와 스마트폰이 통신을 하기 위해서는 먼저 페어링을 해야 합니다. 무선 장치이기 때문에 주변의 다양한 디바이스가 존재하기 때문에 연결하려고 하는 디바이스를 선택하고 PIN 번호(일종의 암호)를 입력해서 연결해야 합니다. PIN 번호를 물어보지 않는 경우도 있습니다. 페어링 과정은 라즈베리파이의 GUI 환경에서 비교적 쉽게 연결할 수 있습니다. 페어링 과정은 1회만 필요하고 한 번 연결이 완료된 디바이스는 페어링 과정 없이 바로 연결이 가능합니다. 라즈베리파이에서 페어링과 연결 과정을 자세히 확인하기 이해서 블루투스 컨트롤 프로그램을 먼저 실행합니다.

```
pi@raspberrypi:~ $ bluetoothctl
```



프롬프트가 "bluetooth"로 변경되고 이제 블루투스 제어 명령어를 실행할 수 있게 되었습니다. 스마트폰에서 블루투스 장치를 켜고 대기합니다.

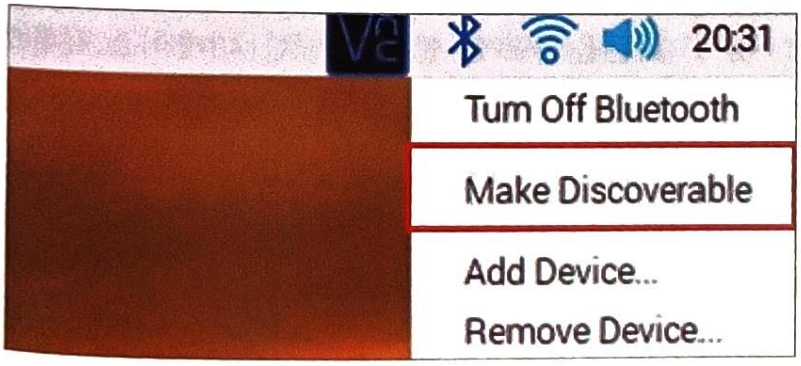
```
[bluetooth]# scan on    -- 블루투스 장치 검색 시작
```



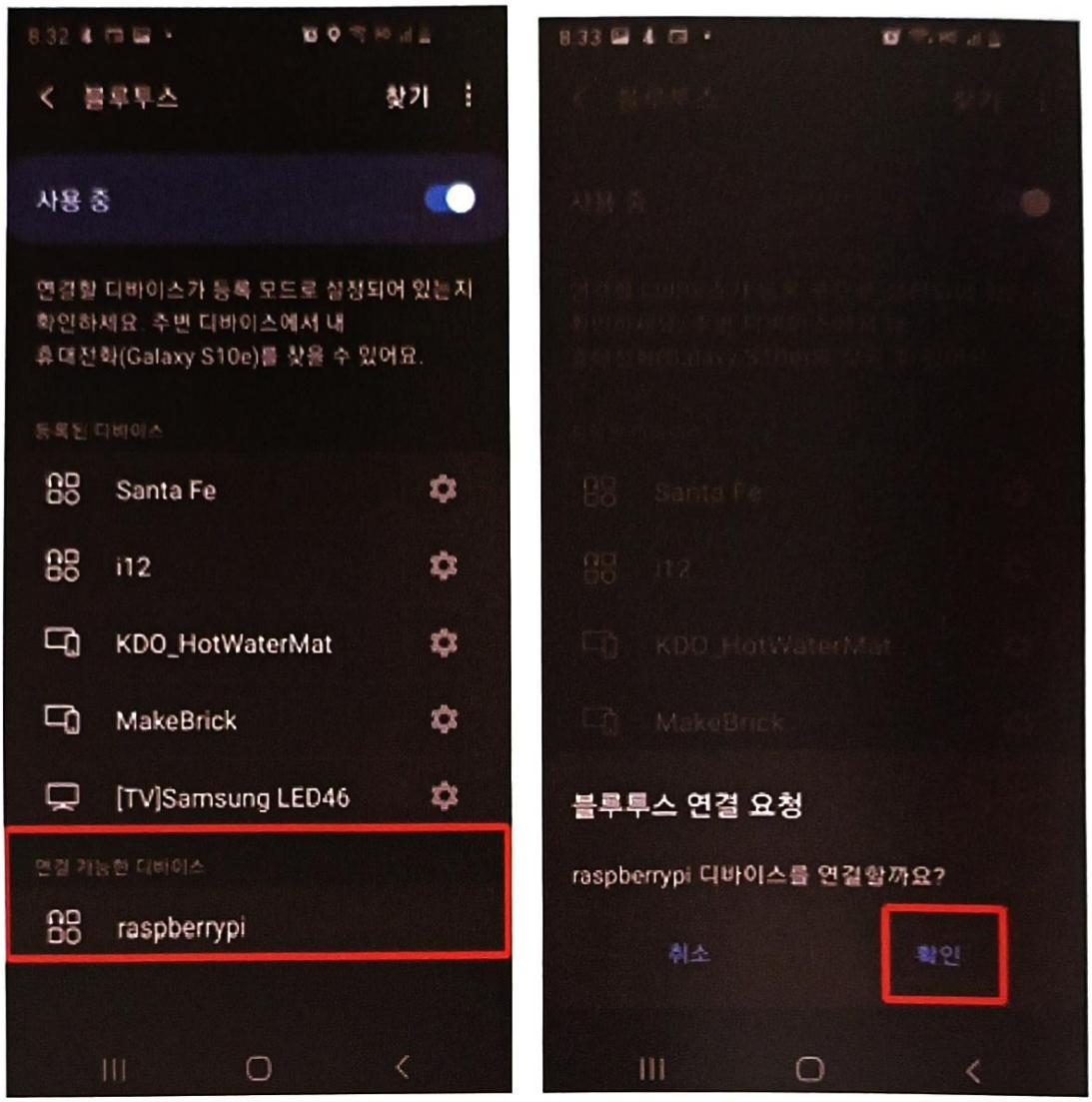
첫 번째 줄의 Controller 옆의 6자리 16진수가 라즈베리파이의 블루투스 하드웨어 MAC 주소입니다. 모든 블루투스 장치의 하드웨어 MAC 주소는 중복되지 않은 고유한 주소를 가지고 있어서 디바이스를 연결하고 관리하는 데 사용이 됩니다.

```
[bluetooth]# scan off    -- 블루투스 장치 검색 중단
[bluetooth]# quit        -- bluetoothctl 명령어 종료
```

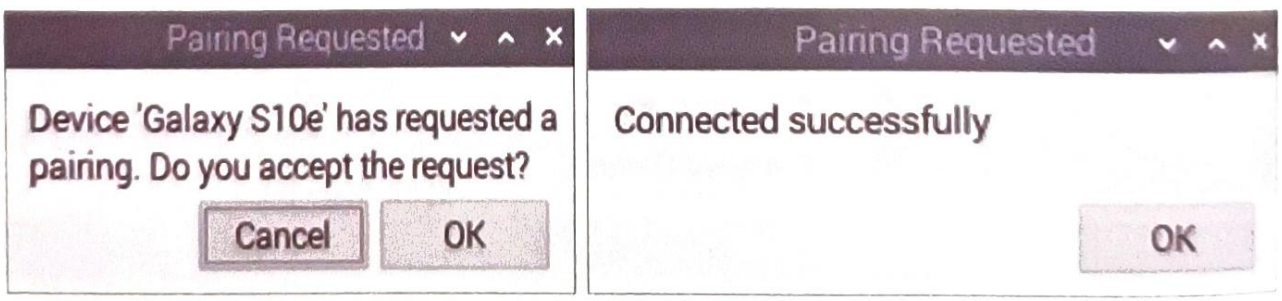
여기까지 문제없이 진행되었다면 라즈베리파이의 블루투스 장치에 문제가 없는 것입니다. 이제 라즈베리파이 GUI 환경에서 스마트폰과 연결해 보도록 하겠습니다.



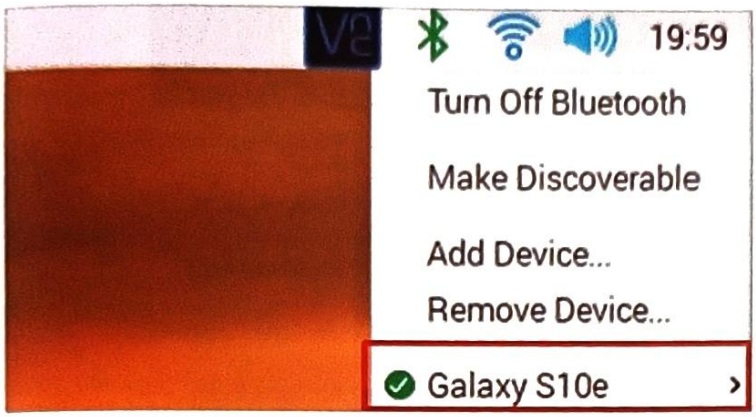
라즈비안 OS 상단의 블루투스 아이콘을 클릭하고 "Make Discoverable"를 선택합니다. 이렇게 하면 스마트폰에서 라즈베리파이의 블루투스 장치를 검색할 수 있습니다.



스마트폰에서 블루투스를 켜고 "찾기" 버튼을 누르면 연결 가능한 디바이스 목록에 "raspberrypi"가 나타나면 이 디바이스를 선택하고 연결합니다.



라즈비안 OS 화면에 위와 같은 페어링 연결 요청이 나오고 "OK" 버튼을 눌러서 페어링을 완료합니다.

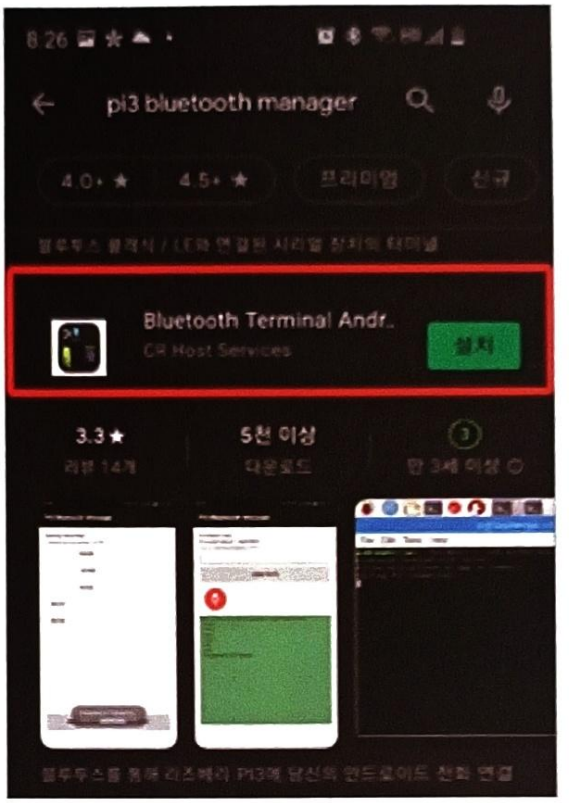


연결이 완료되면 왼쪽 그림과 같이 블루투스 디바이스 이름이 나타나게 됩니다.

스마트폰 디바이스와 페어링 및 연결이 완료되었습니다. 본 교재에서와 반대로 라즈베리파이에서 스마트폰을 먼저 연결 요청들 보낼 수도 있지만 여러 번 테스트 결과 스마트폰에서 먼저 연결 요청들 보내는 방식이 연결하는데 문제가 덜 발생하였습니다. 이제 라즈베리파이에서 UART 통신으로 스마트폰과 데이터 통신을 해보도록 하겠습니다.

(3) 안드로이드 디바이스용 블루투스 통신 앱 설치

앞에서 라즈베리파이와 안드로이드 스마트기기가 페어링 절차를 거쳐서 연결까지는 되었지만 라즈베리파이와 블루투스로 데이터를 주고받기 위해서 안드로이드 디바이스에 추가로 앱을 설치해야 합니다. 일반적으로 검색 사이트에서 찾아보면 BlueTerm이라는 앱을 많이 이용하고 있는데 2021년 1월 현재 최신 버전의 안드로이드 버전에서는 BlueTerm 앱이 제대로 동작하지 않습니다. 그래서 여러 앱을 테스트해 본 결과 "pi3 bluetooth manager"라는 앱이 라즈베리파이와 통신이 잘 되는 것을 확인하였습니다.



pi3 bluetooth manager 앱

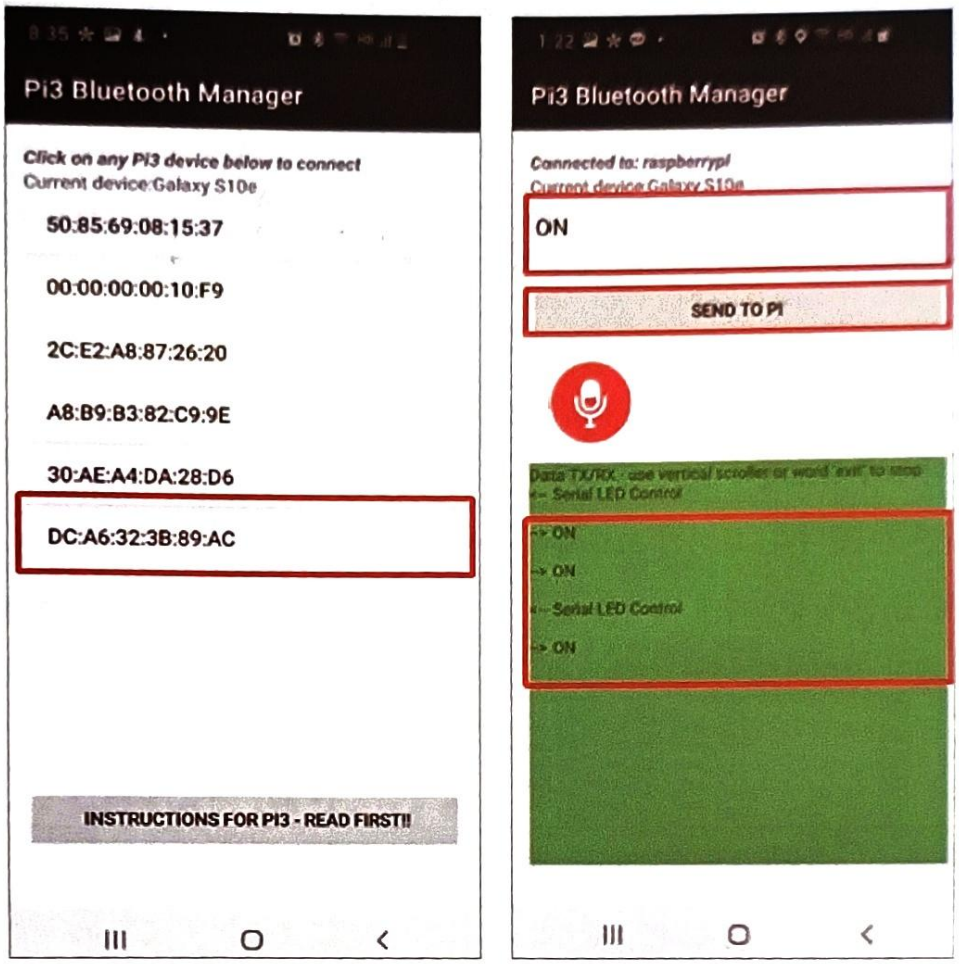
앱을 설치하고 실행합니다. 물론 안드로이드 디바이스에서 앞에서 진행했던 라즈베리파이와 페어링 과정을 먼저 수행한 이후에 진행해야 합니다.

(4) 라즈베리파이와 블루투스 연결 및 데이터 통신

pi3 bluetooth manager 앱에서 라즈베리파이와 연결을 하기 전에 먼저 라즈베리파이에서 아래과 같이 "rfcomm watch all &" 명령어를 먼저 수행해야 합니다. 참고로 명령어의 마지막에 붙은 "&"는 rfcomm watch all 명령어를 백그라운드로로 실행하라는 의미입니다.

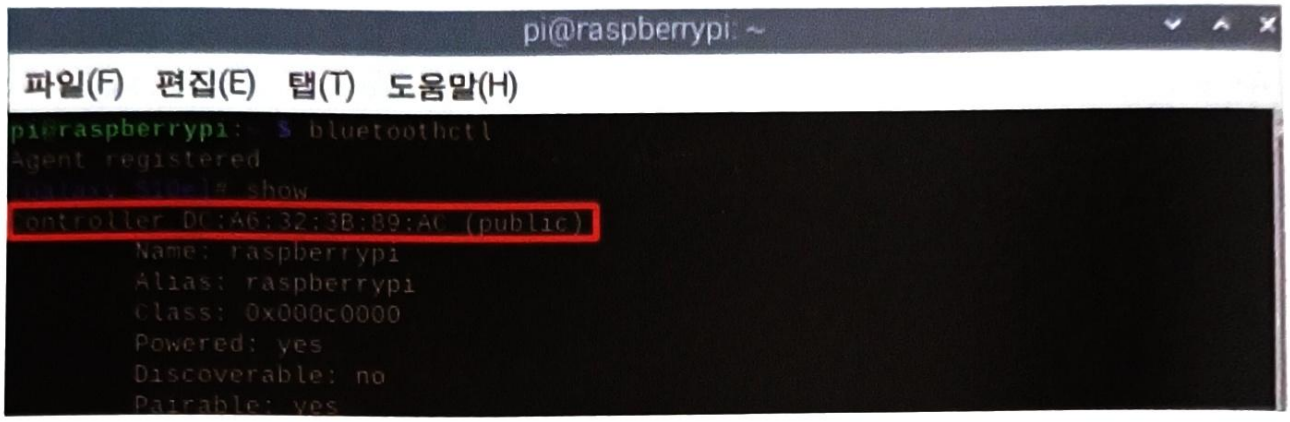
```
pi@raspberrypi:~ $ sudo rfcomm watch all &
[1] 1593
pi@raspberrypi:~ $ Waiting for connection on channel 1
```

pi3 bluetooth manager 앱에서 아직 연결 요청을 하지 않았기 때문에 Waiting 상태에서 대기하고 있습니다. pi3 bluetooth manager 앱에서 페어링이 완료된 라즈베리파이와 연결합니다.



라즈베리파이의 블루투스 하드웨어 MAC 주소를 찾아서 터치하면 위의 2번째 그림과 같이 화면이 바뀌게 됩니다. 라즈베리파이의 하드웨어 MAC 주소는 다음의 명령어로 알 수 있습니다.

```
pi@raspberrypi:~ $ bluetoothctl
Agent registered
[Galaxy S10e]# show
Controller DC:A6:32:3B:89:AC (public)
    Name: raspberrypi
    Alias: raspberrypi
    Class: 0x000c0000
```



pi3 bluetooth manager 앱과 라즈베리파이가 정상적으로 연결되면 아래와 같이 연결된 블루투스 디바이스의 하드웨어 MAC 주소가 표시되고 "/dev/rfcomm0"라는 새로운 디바이스가 생성이 되고 연결이됩니다. 파이썬 코드에서는 단지 "/dev/rfcomm0" 디바이스와 시리얼로 연결하면 시리얼 통신과 동일하게 제어를 할 수 있습니다. 모든 리눅스 기반 운영체제의 장점으로 어떤 하드웨어 장치라도 마치 소프트웨어 입장에서는 파일을 다루듯이 사용할 수 있다는 것입니다.

```
pi@raspberrypi:~ $ sudo rfcomm watch all &
[1] 1593
pi@raspberrypi:~ $ Waiting for connection on channel 1
Connection from 64:7B:CE:6A:4E:2D to /dev/rfcomm0
Press CTRL-C for hangup
```

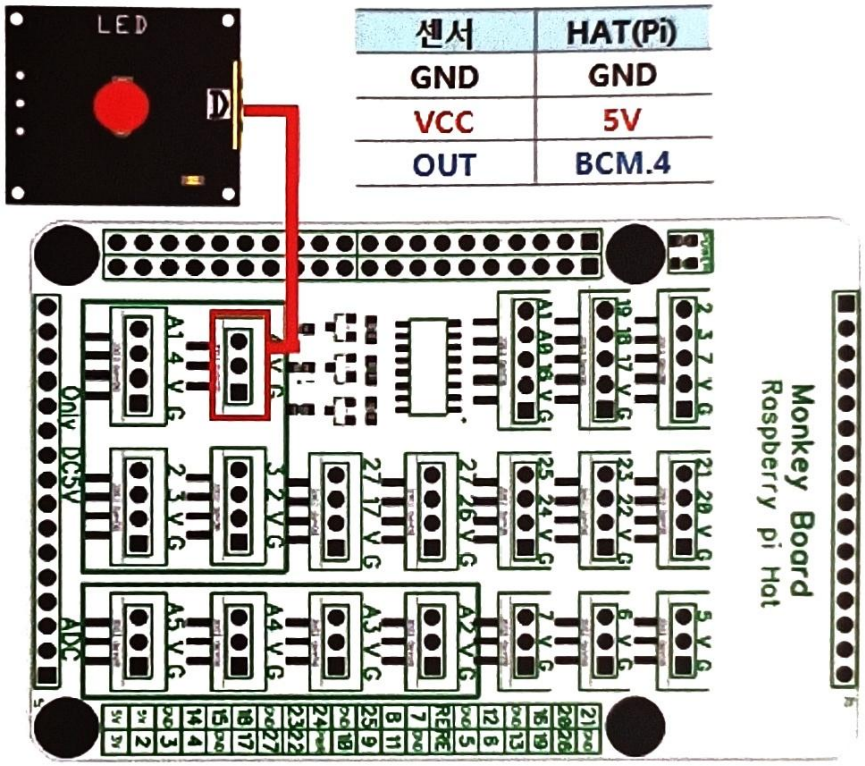
/dev/rfcomm0 장치가 존재해야 파이썬에서 serial.Serial('/dev/rfcomm0', 115200)와 같이 연결이 가능합니다. /dev/rfcomm0 장치는 존재하지 않다가 블투투스 디바이스와 연결이 완료되면 생성되고 연결이 종료되면 다시 사라지게 됩니다.

```
pi@raspberrypi:~ $ ls /dev/rfcomm0
/dev/rfcomm0
```

파이썬 코드를 작성하기 전에 반드시 /dev/rfcomm0 장치가 존재하는지 확인하고 진행하시기 바랍니다. 이제 pi3 bluetooth manager 앱에서 라즈베리파이에 블루투스로 접속을 해서 라즈베리파이에 연결된 LED를 ON/OFF 하는 실습을 해보도록 하겠습니다.

- "ON\n" 문자열 수신 : LED 켜기
- "OFF\n" 문자열 수신 : LED 끄기

배선도 및 회로



라즈베리파이 GPIO HAT 연결

실습 파일 : examples/serial/6.2_bluetooth_1.py

```
1  import RPi.GPIO as GPIO                    # RPi.GPIO 패키지 사용
2  import serial                              # py-serial 패키지 사용
3
4  LED=4 # LED포트 정의
5
```

```
6   GPIO.setmode(GPIO.BCM)                          # BCM모드 사용
7   GPIO.setup(LED, GPIO.OUT)                       # LED를 출력으로 설정
8
9   ser = serial.Serial('/dev/rfcomm0', 115200)     # 115200bps 통신 속도로 설정
10  ser.close()                                     # 포트를 먼저 Close
11  ser.open()                                      # 시리얼 포트 Open
12
13  str = b'Bluetooth LED Control\r\n'              # PC로 전송할 바이트 데이터
14  n = ser.write(str)                              # 바이트 데이터 전송
15
16  try:
17      while True:                                 # PC로부터 받은 데이터가 있다면
18          if ser.readable():                      # \r\n 문자까지 읽기
19              response = ser.readline()           # "ON" 데이터 수신
20              if response == b'ON\n':             # LED 켜기
21                  GPIO.output(LED, True)          # OFF 데이터 수신
22              elif response == b'OFF\n':          # LED 끄기
23                  GPIO.output(LED, False)
24
25              print(response
26  except KeyboardInterrupt:
27      pass
28  finally:
29      ser.close()
```

〈실행 결과〉

```
# PC로부터 전송받은 데이터를 출력
b'OFF\n'
b'ON\n'
```

라즈베리파이로 전송할 문자열을 입력하고 "SEND TO PI" 버튼을 누르면 됩니다. 파이썬 코드에서 readline() 시리얼 데이터를 구분하기 때문에 텍스트를 입력할 때 "ON\n" 형식으로 입력해야 합니다. 참고로 "\n"을 입력하는 방법은 스마트기기 키보드에서 "엔터" 키를 입력하면 됩니다.

파이썬 코드를 보면 이전에 실습했던 시리얼 통신 테스트와 연결할 디바이스 이름이 바뀐 점이 외에는 동일합니다.

참고

블루트스 연결을 하기 위해서 먼저 "sudo rfcomm watch all &" 명령어를 수행해야 하는데 조금 불편합니다. 라즈베리파이가 부팅하면 자동으로 실행될 수 있도록 라즈비안 OS의 서비스에 등록해 놓는 것이 좋습니다.

vi 에디터로 "/lib/systemd/system/rfcomm.service" 파일을 새로 생성합니다.

```
pi@raspberrypi:~ $ sudo vi /lib/systemd/system/rfcomm.service
```

/lib/systemd/system/rfcomm.service 파일의 내용을 작성합니다.

```
[Unit]
Description=RFCOMM service
After=bluetooth.service

[Service]
ExecStart=/usr/bin/rfcomm watch all &

[Install]
WantedBy=multi-user.target
```

라즈비안 OS의 서비스에 등록합니다.

```
pi@raspberrypi:~ $ sudo systemctl enable rfcomm.service
```

이렇게 하면 라즈베리파이가 부팅하면 자동으로 서비스 형태로 실행됩니다. 재부팅 후에 ps 명령어로 정상적으로 실행되고 있는지 확인합니다.

```
pi@raspberrypi:~ $ ps -ef|grep 'rfcomm'
root       389     1  0 14:17 ?        00:00:00 /usr/bin/rfcomm watch all &
root       430     2  0 14:17 ?        00:00:00 [krfcommd]
pi        1047   983  0 14:21 pts/0    00:00:00 grep --color=auto rfcomm
```