

---

# Raspberry Pi Communication with Python

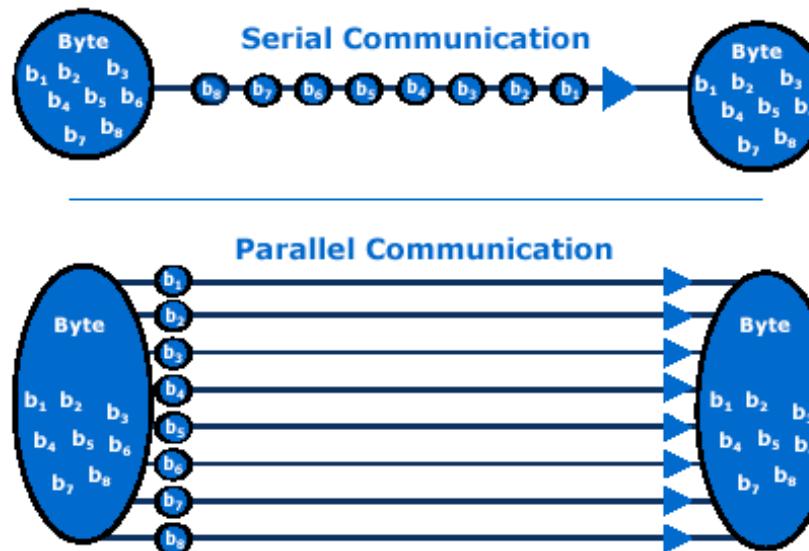
Rev. R610

이세우 (dltpdn@gmail.com)

1. Serial 통신
2. UART
3. I2C
4. SPI
5. Bluetooth
6. BLE - Beacon

## ❖ 직렬통신 Vs 병렬통신

- 직렬(Serial) 통신
  - 여러 비트를 순차적으로 전송
  - 속도가 느리다
  - 통신 회선은 1개
- 병렬(Parallel) 통신
  - 동시에 여러 비트를 전송
  - 속도가 빠르다
  - 통신 회선은 전송 비트 수 만큼 필요



# Serial 통신

Raspberry-Pi Communication

## ❖ 직렬통신

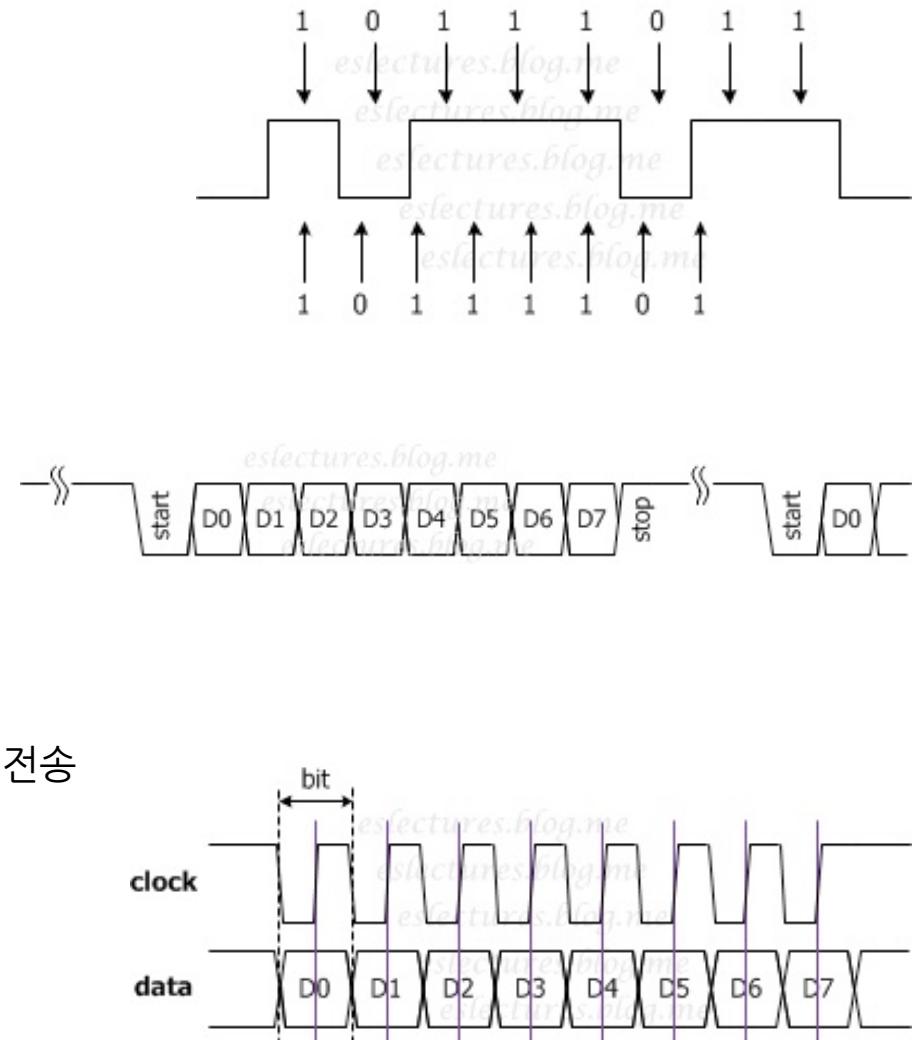
- 데이터를 한 비트씩 순차적으로 전송
- 어느 시점을 한 비트인지 구분 할 방법 필요
- 송수신자 간 비트 구분 시점에 대한 방식

## ❖ 비동기적(Asynchronous) 직렬통신

- 데이터 구분 주기를 서로 약속
- 클럭 신호를 따로 보내지 않음
- 양단간 통신속도가 맞지 않으면 통신 불능
- 시작비트와 정지비트가 추가로 필요
- 1:1 통신만 가능
- RS-232(UART) 통신 프로토콜이 대표적

## ❖ 동기적(Synchronous) 직렬통신

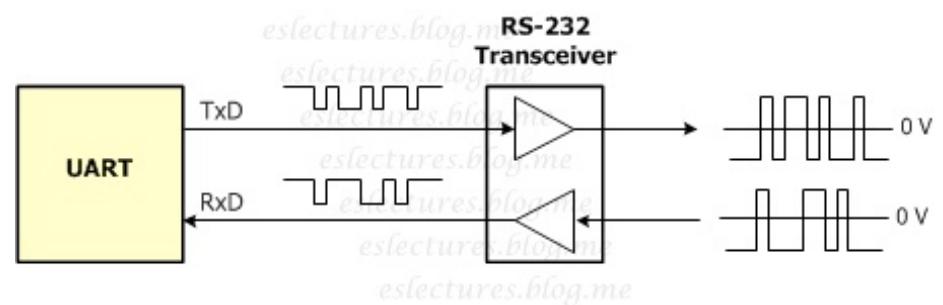
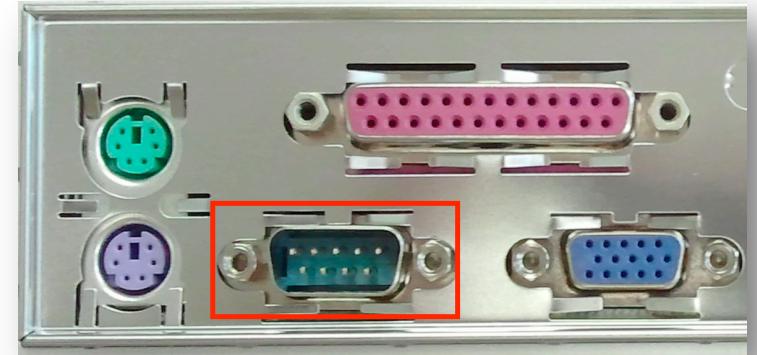
- 데이터 신호와 비트 구분신호(Clock)를 별도로 전송
- 클럭에 마춰서 데이터 신호 인식
- 양단간 속도 약속 불필요
- 최고 속도 제한
- 1:N 통신 가능,
- Master/Slave 관계, Master가 클럭 주도
- I<sup>2</sup>C, SPI 통신 프로토콜이 대표적



1. Serial 통신
2. UART
3. I2C
4. SPI
5. Bluetooth
6. BLE - Beacon

### ❖ UART/RS-232

- UART(Universal Asynchronous Receiver Transmitter)
  - 비동기 통신을 위한 기계장치
  - 대부분의 MCU 하드웨어 기능 내장
  - 클럭과 시작/정지 비트를 소프트웨어로도 구현 가능
  - 통신속도 지정 : Baud Rate
  - 2개의 데이터 회선 사용
    - TxD(송신), RxD(수신)
- RS-232
  - 미국 EIA(Electronic Industries Association)
  - UART에 대한 전기적, 기계적(커넥터) 특성에 대한 표준
  - PC에서의 시리얼 포트를 대표
- UART/RS-232 통신
  - MCU UART TTL(Transistor to Transistor Logic) : 0~5V
  - RS232 : -12V ~ +12V
  - RS232 - TTL 변환칩
    - MAX232 / Maxim사
  - USB - TTL 변환칩
    - FT232R / FTDI Chip사
    - CP2102 / Silicon Lab사



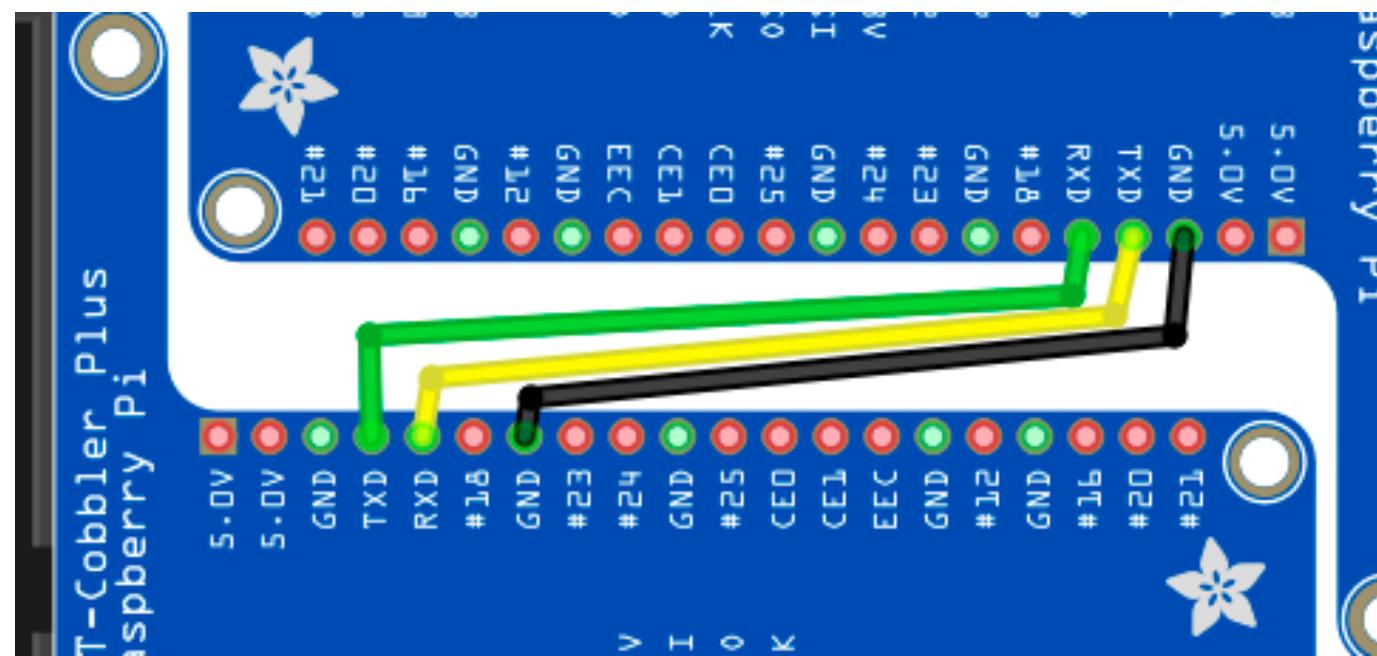
## ❖ RaspberryPi UART

- UART 활성화
  - sudo vi /boot/config.txt (백업 필수)
    - enable\_uart=1
  - sudo vi /boot/cmdline.txt (백업 필수)
    - 아래 내용 삭제 (getty disable)
    - console=serial0,115200
- sudo reboot
- ls /dev/serial\*
  - 아래 내용 확인
    - serial0 --> ttyS0(UART)
    - serial1 --> ttyAMA0(Bluetooth)
  - serial0에 연결된 장치 이름 사용

```
pi@raspberrypi:~ $ ls -l /dev/serial*
lrwxrwxrwx 1 root root 5 Nov 3 19:37 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Nov 3 19:37 /dev/serial1 -> ttyAMA0
```

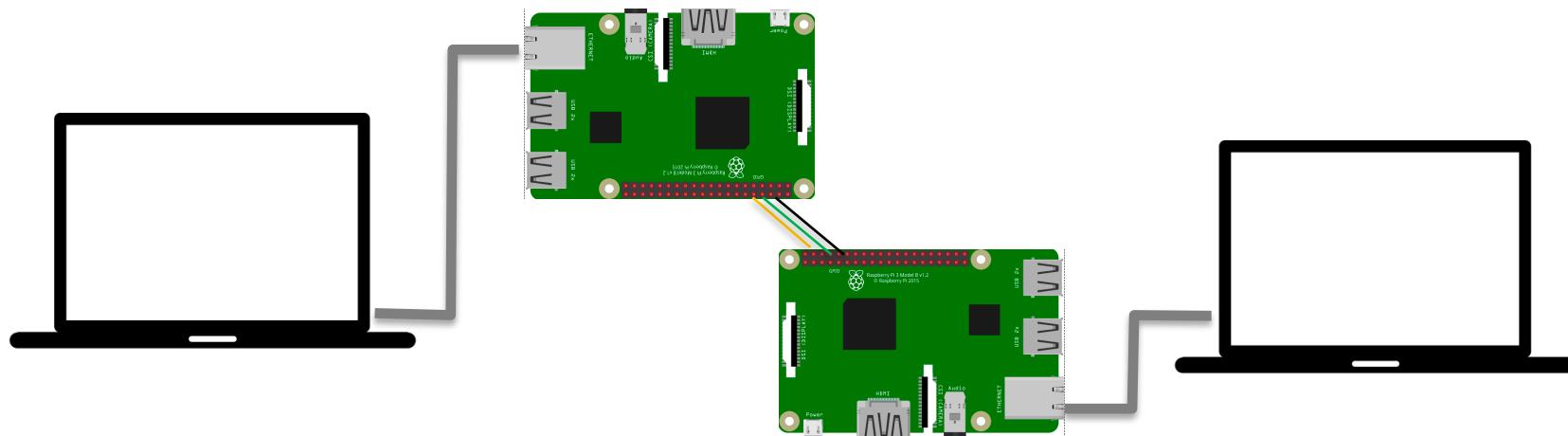
### ❖ RaspberryPi 2대간 URAT 통신

- 회로 구성(Tx -Rx 교차 연결)
  - Tx - Rx
  - Rx - Tx
  - GND -GND



### ❖ RaspberryPi 2대간 URAT 통신

- serial 모듈
  - port = serial.Serial('/dev/xxx', baudrate=115200, timeout=1.0)
  - port.isOpen()
  - port.write('str')
  - ch = port.read()
  - port.close()



### ❖ RaspberryPi 2대간 URAT 통신

- 송신 측

```
import serial

port = serial.Serial( "/dev/ttyS0", baudrate=115200, timeout=3.0)
try:
    if port.isOpen():
        print 'serial writer is ready.'

    while True:
        line = raw_input(">")
        print line
        port.write(line + '\r')
finally:
    port.close()
```

```
pi@raspberrypi:~ $ python uart_writeonly.py
serial writer is ready.
>hello
hello
>hi
hi
>world
world
>한글
한글
```

### ❖ RaspberryPi 2대간 URAT 통신

- 수신 측

```
import serial

port = serial.Serial("/dev/ttyS0", baudrate=115200, timeout=3.0)
def readline():
    line = ""
    while True:
        ch = port.read()
        line +=ch
        if ch =='\r' or ch == "":
            if len(line) > 0 :
                return line

try:
    if port.isOpen() :
        print 'serial is ready.'
        while True:
            print "recv:" + readline()
finally:
    port.close()
```

```
[pi@raspberrypi:~ $ python uart_READONLY.py
serial is ready.
recv:hello
recv:hi
recv:world
recv:한글
```

### ❖ RaspberryPi 2대간 URAT 통신

- 양방향 송수신

```
import serial, threading
port = serial.Serial("/dev/ttyS0", baudrate=115200, timeout=3.0)
```

```
class ReadThread(threading.Thread):
    flag = True
```

```
    def run(self):
        while self.flag:
            line = self.readline()
            if len(line) > 0 :
                print "recv:" +line
            print "read thread died."
```

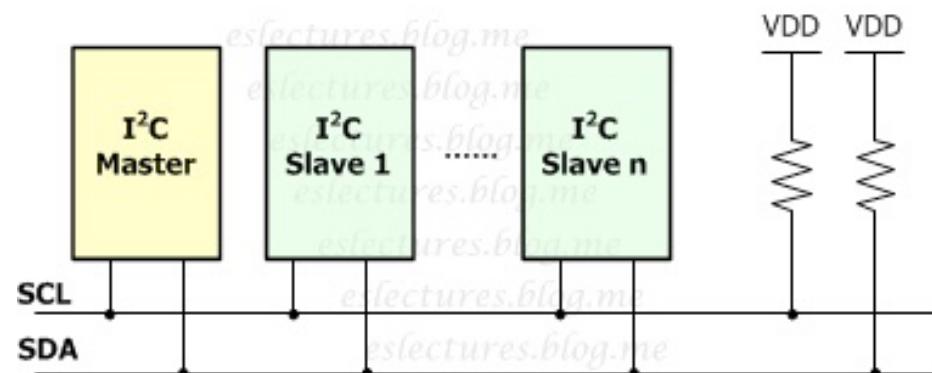
```
    def readline(self):
        line = ""
        while True:
            ch = port.read()
            line +=ch
            if ch =='\r' or ch == ":":
                return line
```

```
try:
    if port.isOpen():
        t = ReadThread()
        t.start()
        print 'serial is ready.'
    while True:
        line = raw_input(">")
        if line == 'exit':
            t.flag = False
            break
        print line
        port.write(line + '\r')
        print 'bye'
finally:
    port.close()
```

1. Serial 통신
2. UART
3. I<sup>2</sup>C
4. SPI
5. Bluetooth
6. BLE - Beacon

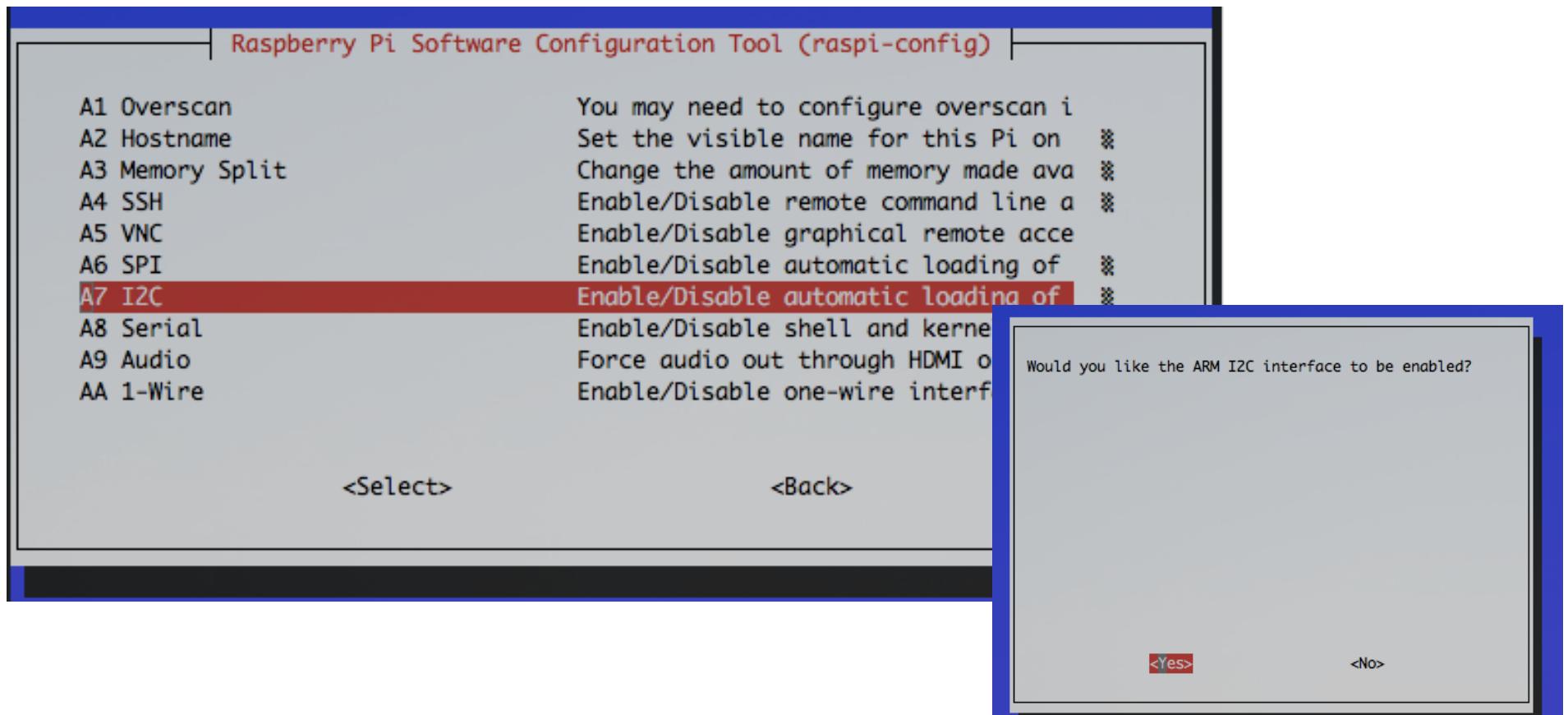
## ❖ I<sup>2</sup>C(Inter Integrated Circuit) 통신

- 동기적 직렬통신
- MCU와 주변장치간의 통신을 위한 프로토콜
- Philips사 개발
- 통신 회선 : 2회선(TWI : Two Wire Interface)
  - SCL(Serial Clock)
  - SDA(Serial Data)
  - 각 회선 풀업저항 필요
- 실제 통신 회선은 1개(SDA)
  - 반이중 통신만 가능
  - 전송 속도 느림
- 하나의 버스에 여러장치 연결 가능
  - Slave 장치 고유의 주소(7비트)
  - 최대 128개( $2^7$ ) 슬레이브 장치
  - 장치 추가에 따른 회선 추가 없음



## ❖ RaspberryPi 환경 설정

- I2C 활성화
  - sudo raspi-config
  - Advanced Options > A7 I2C



## ❖ RaspberryPi 환경 설정

- 장치 활성화 확인
  - ls -l /dev/i2c\*
- kernel에 module 로딩 확인
  - sudo vi /etc/modules
    - “i2c-dev”
    - 내용 유무 확인, 없으면 추가
  - lsmod
    - 목록에서 i2c\_dev 확인
- I2C 통신 허용
  - sudo vi /etc/modprobe.d/raspi-blacklist.conf
    - “blacklist i2c-bcm2708”
    - 내용 삭제 또는 주석처리

```
[pi@raspberrypi:~ $ ls -l /dev/i2c*
crw-rw---- 1 root i2c 89, 1 Nov 22 08:28 /dev/i2c-1
```

## ❖ Arduino

### ▪ 코드 스케치

```
#include <Wire.h>

#define ADDR 0x04
#define PIN_LED 13

int cnt = 0;
int lastVal = -1;

void setup(){
    Wire.begin(ADDR);
    Wire.onReceive(onRecv);
    Wire.onRequest(onReq);

    pinMode(PIN_LED, OUTPUT);
    Serial.begin(9600);
    Serial.println("ready.");
}

void loop(){
    delay(100);
}
```

```
void onRecv(int bytes){
    int val;
    while(Wire.available()){
        val = Wire.read();

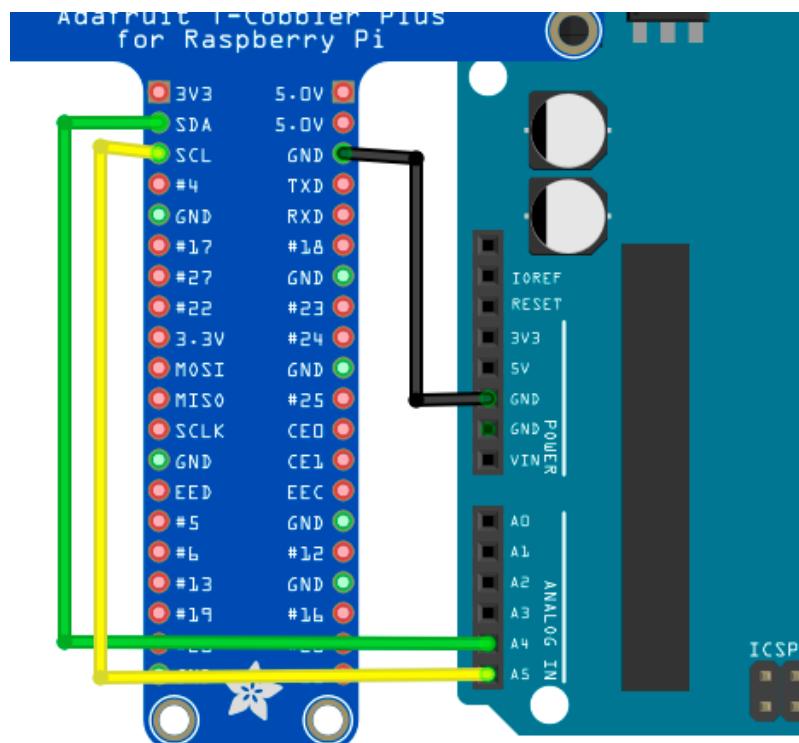
        Serial.print("read:");
        Serial.println(val);
        if(lastVal != val){
            lastVal = val;
            cnt++;
            if(val == 1){
                digitalWrite(PIN_LED, HIGH);
            }else if(val == 0){
                digitalWrite(PIN_LED, LOW);
            }
        }
    }
}

void onReq(){
    Serial.print("req:");
    Serial.println(cnt);
    Wire.write(cnt);
}
```

## ❖ RaspberryPi - Arduion 간 I<sup>2</sup>C 통신

- 회로 연결

R-Pi	Arduino
3(SDA1)	A4(SDA)
5(SDL1)	A5(SCL)



## ❖ RaspberryPi 환경 설정

- I2C Tool 설치
  - \$ sudo apt-get install i2c-tools
- I2C Bus Scan
  - \$ i2cdetect -y 1
    - Arduino 코드에서 지정한 Slave 주소(0x04) 출력 확인

```
pi@raspberrypi:~ $ i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- 04 - - - - - - - - - - - - - - - - - - - -
10: - - - - - - - - - - - - - - - - - - - - - - - -
20: - - - - - - - - - - - - - - - - - - - - - - - -
30: - - - - - - - - - - - - - - - - - - - - - - - -
40: - - - - - - - - - - - - - - - - - - - - - - - -
50: - - - - - - - - - - - - - - - - - - - - - - - -
60: - - - - - - - - - - - - - - - - - - - - - - - -
70: - - - - - - - - - - - - - - - - - - - - - - - -
```

- Python module 설치
  - sudo apt-get install python-smbus
  - sudo apt-get install python3-smbus (python3의 경우)

## ❖ Raspberry Pi

- i2c.py

```
import smbus
bus = smbus.SMBus(1)

address = 0x04

def writeNumber(value):
    bus.write_byte(address, value)

def readNumber():
    number = bus.read_byte(address)
    return number

while True:
    var = input("0=Led Off, 1=Led On, 3=Get Count > ")
    if var < 2:
        writeNumber(var)
        print "Led ", var
        continue
    else:
        number = readNumber()
        print "Arduino LED Count", number
```

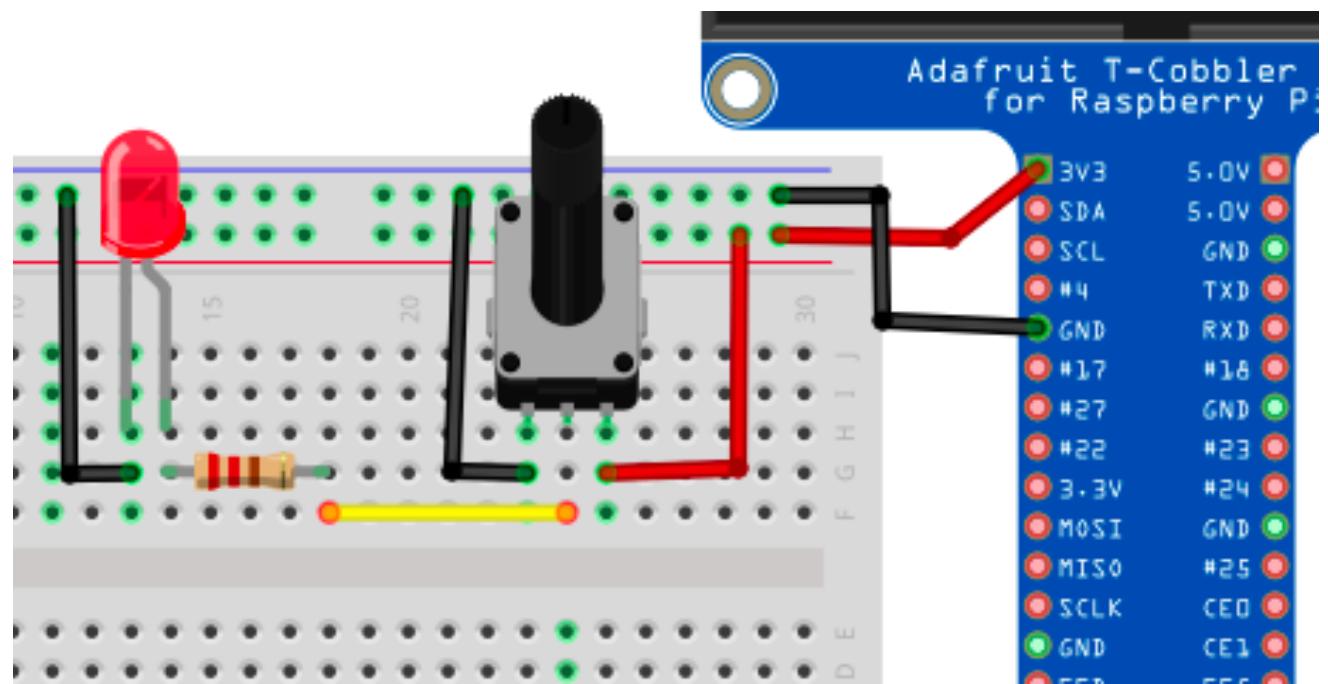
```
[pi@raspberrypi:~ $ python i2c.py
[0=Led Off, 1=Led On, 3=Get Count > 1
 Led 1
[0=Led Off, 1=Led On, 3=Get Count > 0
 Led 0
[0=Led Off, 1=Led On, 3=Get Count > 3
 Arduino LED Count 2
```

```
ready.
read:1
read:0
req:2
read:1
read:1
req:3
read:1
req:3
```

1. Serial 통신
2. UART
3. I2C
4. SPI
5. Bluetooth
6. BLE - Beacon

### ❖ 로터리 가변저항(Rotary Potentiometer)

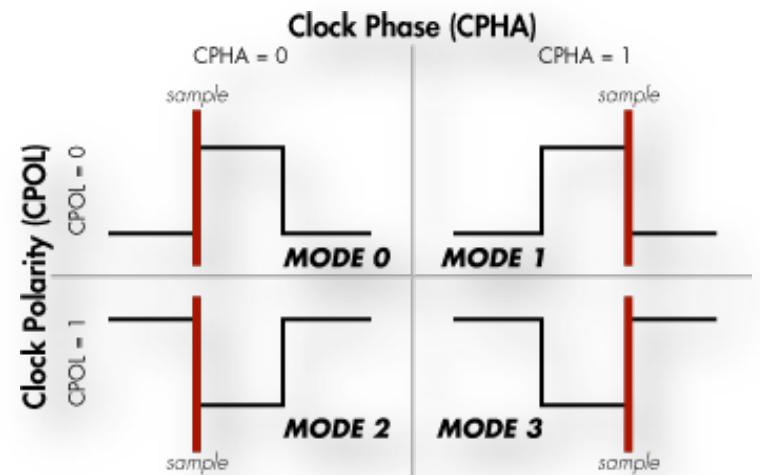
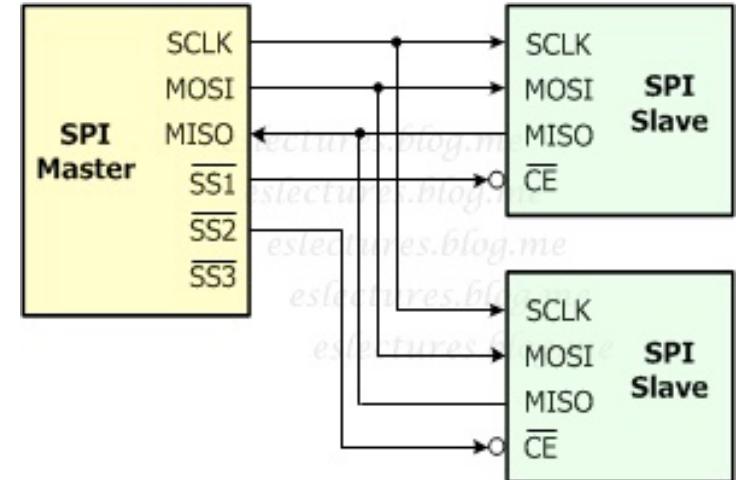
- 손잡이를 돌려 저항값 변경
- Pin1 : 5V, Pin2 : OUT, Pin3 : GND
- 가변 저항 출력을 이용해서 LED Fading



## ❖ SPI(Serial Peripheral Interconnect) 통신

- 동기적 직렬통신
- 모토롤라에 의해 개발
- 통신 회선 : 4회선
  - SCK, SCLK(Serial Clock) : 마스터가 클럭 전송
  - MOSI(Master Output Slave Input) : 마스터 전송
  - MISO(Master Input Slave Output) : 마스터 수신
  - SS(Slave Select) : 마스터가 슬레이브 선택
    - 슬레이브 장치마다 고유의 1회선 추가
    - 선택한 슬레이브에만 '0', 나머지는 '1'
- 실제 통신 회선은 2개(MOSI, MISO)
- 전이중 통신, 전송 속도 빠름
- SPI Mode
  - 시작 비트(0,1), 클럭 비트(0,1) 설정

SPI Mode	CPOL(Clock Polarity)	CPHA(Clock Phase)
0	0	0
1	0	1
2	1	0
3	1	1



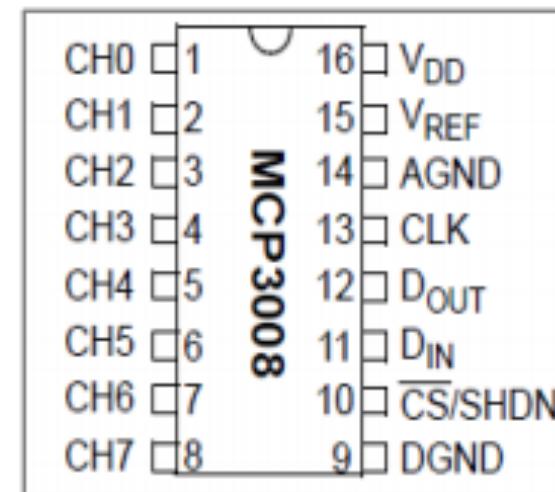
### ❖ MCP3008

- ADC (Analog Digital Convertor)
- 10bit 해상도 (0~ 1023)
- SPI 통신
- 가변 저항의 Analog 신호를 입력 실습



### ❖ 회로연결

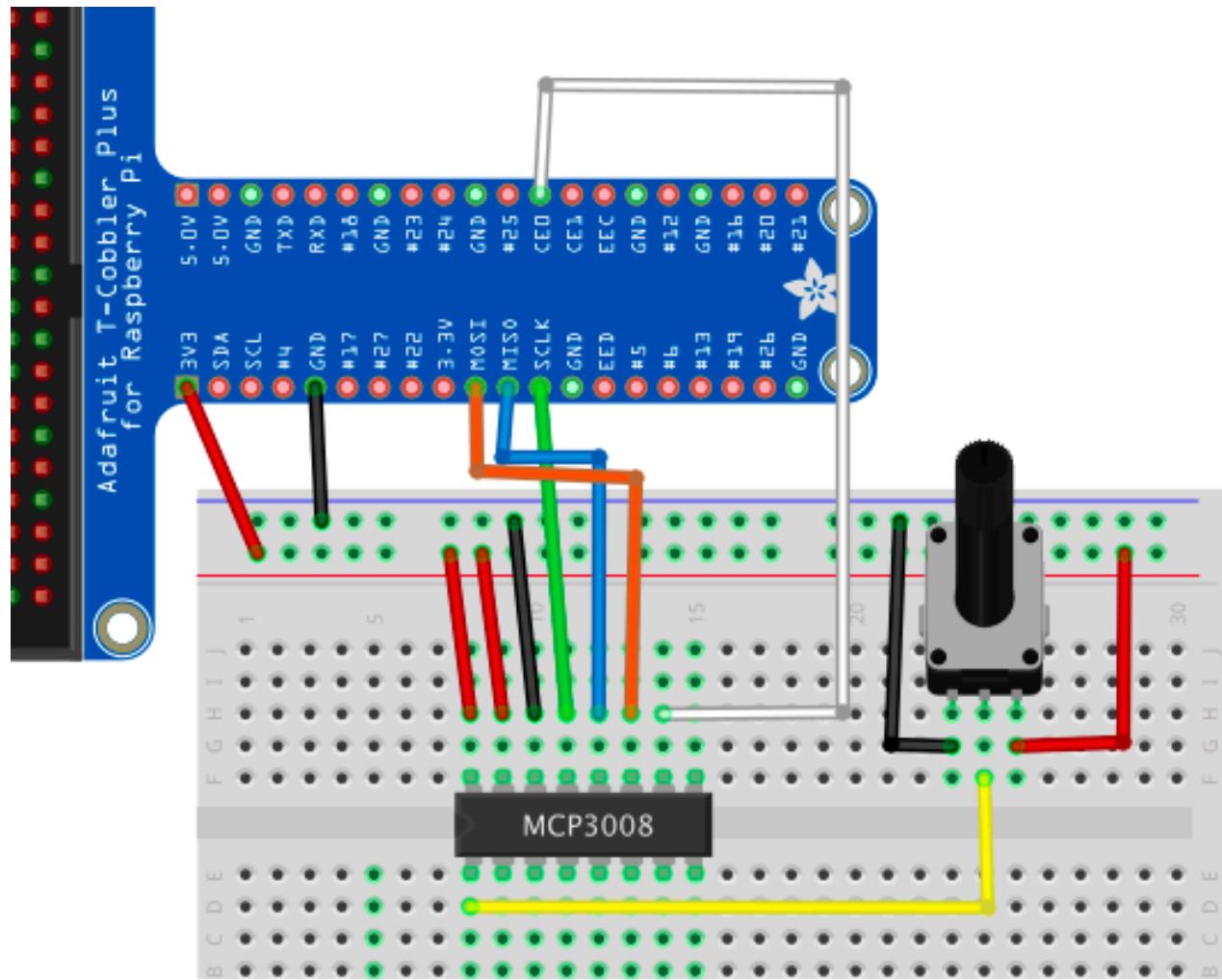
- Vdd - 3.3v
- Vref - 3.3v
- AGND - GND
- CLK - CLK(GPIO11)
- Dout - MISO(GPIO09)
- Din - MOSI(GPIO10)
- CS - CE0(GPIO08)
- CH0 - 가변저항 OUT



SPI

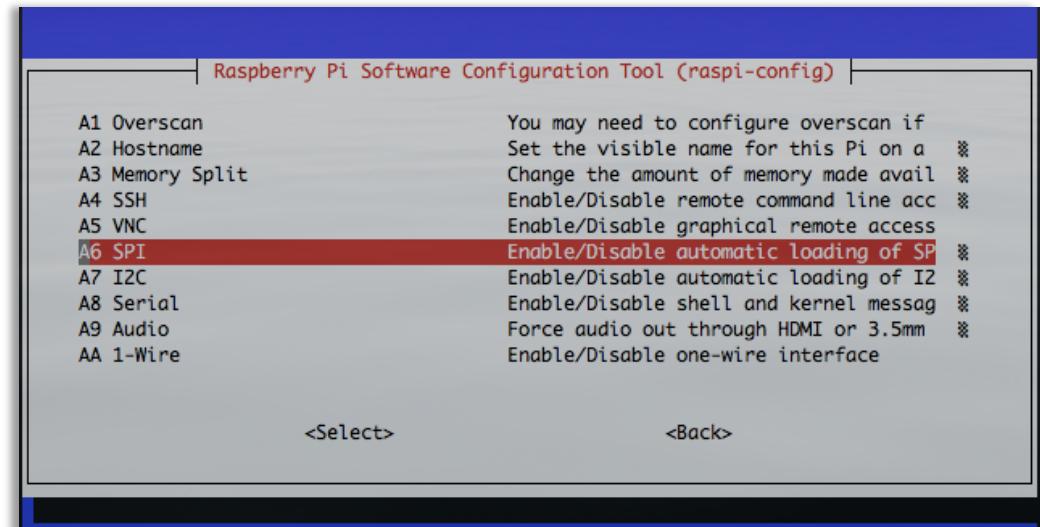
## *Raspberry-Pi Communication*

## ❖ 가변저항 MCP3008로 읽기



## ❖ SPI 활성화 및 모듈 설치

- Raspberry-pi SPI 포트 활성화
  - sudo raspi-config
    - 8.Advanced Options > SPI > Yes
  - ls /dev/spi\*
- Py-spidev 모듈 설치
  - spidev 제어 모듈 (/dev/spidev0.0 , /dev/spidev0.1)
  - <https://github.com/doceme/py-spidev>
  - git clone git://github.com/doceme/py-spidev
  - cd py-spidev
  - Sudo python setup.py install
  - 또는
  - Pip install py-spidev

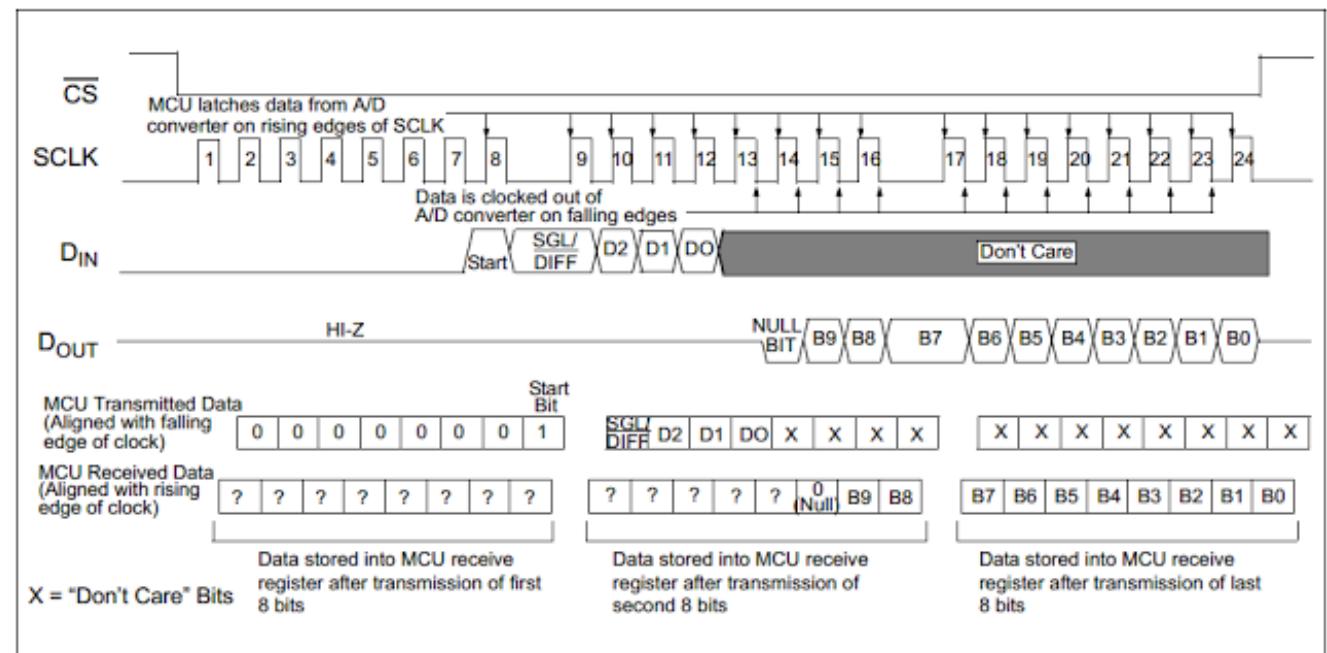


## ❖ MCP3008 통신

- <http://www.farnell.com/datasheets/808965.pdf>
- 3Byte(24bit) 전송/수신
- 전송
  - 1: Start bit : 1
  - 2: Channel configuration
  - 3: Ignored
- 수신
  - 1: Ignored
  - 2-3 : 10bit value

TABLE 5-2: CONFIGURE BITS FOR THE MCP3008

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7



## ❖ MCP3008로 가변저항 읽기

- open(port, dev)
  - Port : 0
  - Dev : CE0=0, CE1=1
- xfer2([byte\_1, byte\_2, byte\_3])
  - byte\_1 : 1
  - byte\_2 : channel config
    - 1000 000 : channel 0
  - byte\_3 : 0(ignore)
- adc\_out
  - r[0] : ignored
  - r[1] : 10bit의 최상위 2bit 값
  - r[2] : 10bit의 하위 8bit 값

```
import spidev, time

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

try:
    while True:
        reading = analog_read(0)
        voltage = reading * 3.3 / 1024
        print("Reading=%d\tVoltage=%f" % (reading,
                                         voltage))
        time.sleep(1)
finally:
    spi.close()
```

1. Serial 통신
2. UART
3. I2C
4. SPI
5. Bluetooth
6. BLE - Beacon

### ❖ Bluetooth

- 개인 근거리 무선 통신을 위한 표준
  - 1994년 Ericsson(스웨덴 통신장비 제조사)
  - SIC : Bluetooth Special Interest Group
  - <https://www.bluetooth.com/>
- Protocol
  - RFCOMM
    - Radio Frequency Communication
    - RS232 시리얼 포트를 무선으로 대체
    - TCP와 유사
    - 30개의 포트(채널) 사용 가능
  - L2CAP
    - Logical Link Control and Adaption Protocol
    - 패킷 방식, 최대 672Bytes
    - UDP와 유사
    - RFCOMM의 Transport Layer로 사용됨
  - ACL
    - Asynchronous Connection-oriented Logical
    - IP 프로토콜과 유사, L2CaP의 하부 프로토콜로 사용
  - SCO
    - Synchronous Connection-Oriented
    - 오디오 전송 프로토콜(64kb/s)



### ❖ BlueZ

- <http://www.bluez.org>
- 리눅스 공식 블루투스 프로토콜 스택 구현 라이브러리
- bluez 설치
  - sudo apt-get install bluez
- bluez 관련 명령어
  - hciconfig
    - Bluetooth 장치 설정
  - hcitool
    - Bluetooth 장치 connection과 command 전송을 위한 도구
  - l2ping
    - 주어진 주소에 L2CAP echo/ping 요청
  - sdptool
    - Bluetooth 장치에 대한 SDP(Service Discovery Protocol) 질의 제공
  - frcomm
    - Bluetooth 하부 시스템의 RFCOMM 설정 및 관리
    - Bluetooth 장치를 Serial(RS232) 장치처럼 사용
  - bluetoothd
    - Bluetooth 장치의 모든 관리를 담당하는 서비스 데몬
    - bluetoothd -v

## ❖ Raspberry Pi 2대간 통신 설정

- Server
  - sudo hciconfig
    - 설치된 Bluetooth 장치와 주소 확인
  - sudo hciconfig hci0 name "MyName"
    - hci0 장치를 스캔목록에 보여질 이름 설정
  - sudo hciconfig hci0 piscan
    - hci0 장치를 스캔목록에 보이도록 설정
  - sudo hciconfig hci0 noscan
    - hci0 장치를 스캔목록에 보이지 않도록 설정
- Client
  - hcitool scan
    - 주변의 사용 가능한 장치목록 스캔
  - sudo l2ping XX:XX:XX:XX:XX:XX

### ❖ Raspberry Pi 2대간 통신 설정

- Server
  - sudo sdptool browse local
    - SDP에 Serial Port 서비스가 있는지 확인
  - sudo sdptool add -channel=22 SP
    - 채널 22로 Serial Port 서비스 추가
  - sudo rfcomm listen /dev/rfcomm0 22
    - /dev/rfcomm0 22번 채널로 대기
    - listen 상태에서 client 접속 후 별도의 터미널 접속 필요

```
pi@raspberrypi:~ $ sudo rfcomm listen /dev/rfcomm0 22
Waiting for connection on channel 22
Connection from B8:27:EB:14:18:CC to /dev/rfcomm0
Press CTRL-C for hangup
```

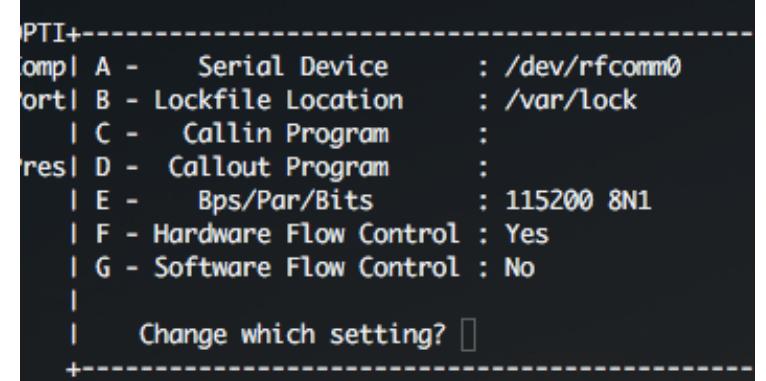
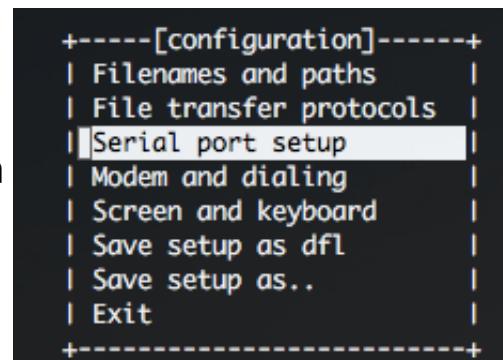
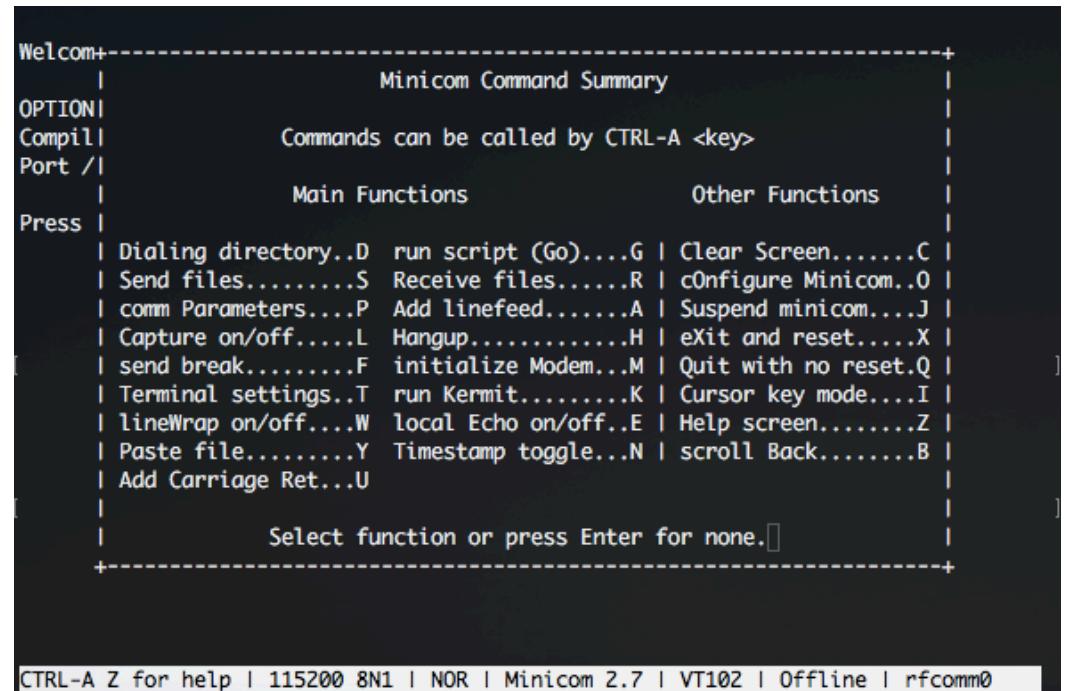
- sudo minicom /dev/rfcomm0
  - /dev/rfcomm0로 시리얼 통신
- Client
  - sudo rfcomm bind /dev/rfcomm0 XX:XX:XX:XX:XX 22
    - 원격주소의 22채널을 /dev/rfcomm0에 바인드
  - sudo minicom /dev/rfcomm0

# Bluetooth

## Raspberry-Pi Communication

### ❖ minicom 설정

- 설치
  - sudo apt-get install minicom
- 실행
  - sudo minicom
- 시리얼 포트 설정
  - Ctrl + A, Z
  - O : cOnfiguration
  - Serial port setup 선택
  - A : Serial Device
    - /dev/rfcomm0
  - Save setup as dfl
  - Exit
- E : local Echo on/off
- A : Add linefeed
- U : Add Carriage Return
- 재 시작



# Bluetooth

## Raspberry-Pi Communication

### ❖ Serial 통신

- Server

- sudo rfcomm listen /dev/rfcomm0 22 python bt\_serial.py

- Client

- python bt\_serial.py

```
import serial, threading

port = serial.Serial("/dev/rfcomm0", baudrate=9600,
timeout=3.0)

class ReadThread(threading.Thread):
    flag = True

    def run(self):
        while self.flag:
            line = self.readline()
            if len(line) > 0 :
                print "recv:" +line
        print "read thread died."
```

```
def readline(self):
    line = ''
    while True:
        ch = port.read()
        line +=ch
        if ch =='\r' or ch == '':
            return line

try:
    if port.isOpen() :
        t = ReadThread()
        t.start()
        print 'serial is ready.'
        while True:
            line = raw_input(">")
            if line == 'exit':
                t.flag = False
                break
            print line
            port.write(line + '\r')
        print 'bye'
finally:
    port.close()
```

- ❖ Bluetooth Socket

- 주변 장치 스캔 목록

```
import bluetooth as bt

print "scanning..."

scan_list = bt.discover_devices(lookup_names = True)

print "found %d devices" % len(scan_list)

for addr, name in scan_list:
    print "%s (%s)" % (addr, name)
```

```
[pi@raspberrypi:~ $ python scan.py
scanning...
found 2 devices
B8:27:EB:AD:74:74 (raspberrypi)
A4:84:31:F6:47:73 (Samsung Galaxy S7)
```

### ❖ Bluetooth Socket

- Server Socket , 특정 포트(3)로 대기

```
import bluetooth as bt

server = bt.BluetoothSocket( bt.RFCOMM )

server.bind((" ", 3 ))
server.listen(1)

print "ready to connect..."
socket, address = server.accept()
print "client connected : ", address

data = socket.recv(1024)
print "received :%s" % data
socket.send('Good bye~')
print 'sent data'
socket.close()
server.close()
```

```
[pi@raspberrypi:~ $ python server.py
ready to connect...
client connected : ('B8:27:EB:14:18:CC', 3)
received :Hello Bluetooth World.
sent data
```

- ❖ Bluetooth Socket

- Client Socket, 특정 주소, 포트로 접속

```
import bluetooth as bt

socket = bt.BluetoothSocket( bt.RFCOMM )

socket.connect(("B8:27:EB:AD:74:74", 3))
print "connected."
socket.send("Hello Bluetooth World.")
print "sent data"
recv = socket.recv(1024)
print "recv data :%s" % recv

socket.close()
```

```
[pi@raspberrypi:~ $ python client.py
connected.
sent data
recv data : Good bye~
```

### ❖ Bluetooth Socket

- ServerSocket, 특정 서비스(MyService)로 접속 대기
  - sudo vi /lib/systemd/system/bluetooth.service (백업 필수)
    - ‘bluetoothd’ 뒤에 ‘-C’ 추가
    - bluez5와 SDP간의 버전차이로 동작 오류

```
import bluetooth as bt

server = bt.BluetoothSocket( bt.RFCOMM )
server.bind((" ", bt.PORT_ANY ))
server.listen(1)

bt.advertise_service(server, "MyService",
                     service_classes=[bt.SERIAL_PORT_CLASS],
                     profiles=[bt.SERIAL_PORT_PROFILE])

print "ready to connect..."
socket, address = server.accept()
print "client connected : ", address

data = socket.recv(1024)
print "received :%s" % data
socket.send('Good bye~')
print 'sent data'
socket.close()
server.close()
```

### ❖ Bluetooth Socket

- ClientSocket, 특정 서비스(MyService)로 접속

```
import bluetooth as bt

svc_name = "MyService"
print "finding service : %s" % svc_name
services=bt.find_service(name=svc_name,
                         uuid=bt.SERIAL_PORT_CLASS)
print "Found %d services" % len(services)
for i in range(len(services)):
    match=services[i]

    if(match[ "name"]== "MyService"):
        port=match[ "port"]
        name=match[ "name"]
        host=match[ "host"]
        print "Found service :, name, host, port

        socket=bt.BluetoothSocket( bt.RFCOMM )
        socket.connect((host, port))
        socket.send("Hello world 2")
        data = socket.recv(1024)
        print "recv :, data
        socket.close()

    break
```

1. Serial 통신
2. UART
3. I2C
4. SPI
5. Bluetooth
6. BLE - Beacon

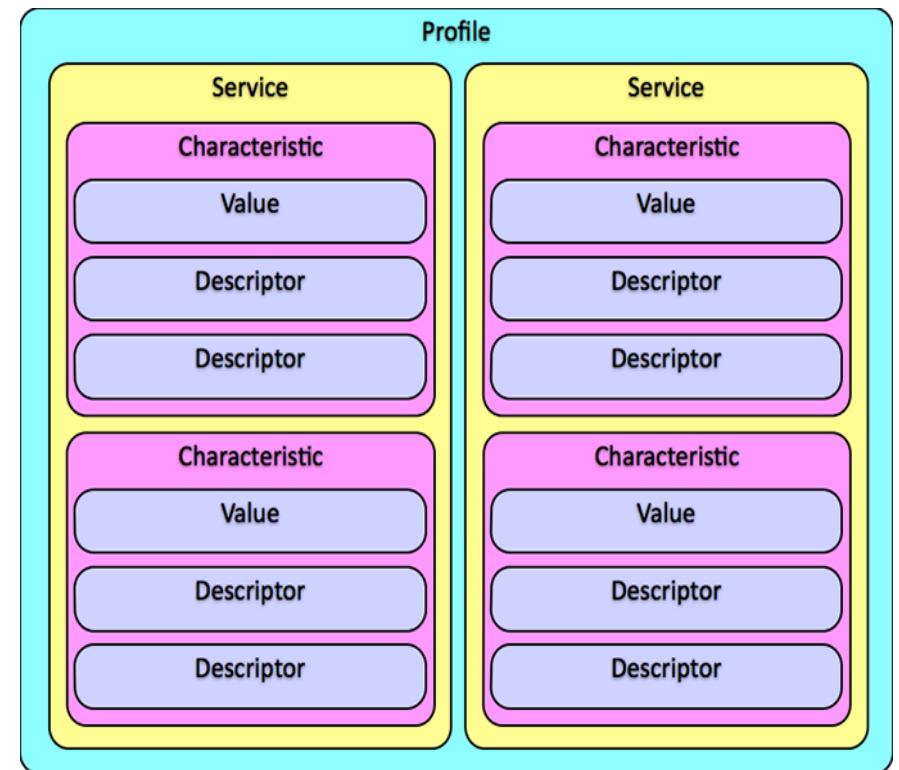
## ❖ BLE

- Bluetooth Low Energy
  - Bluetooth 4.0 표준
  - Bluetooth SMART (BLE 장치, 센서)
  - Bluetooth SMART Ready (Laptop, Smart Phone)
- Profile
  - 블루투스 동작 및 제어를 위한 표준의 개념적 구분
  - GAP, Generic Access Profile
  - GATT, Generic Attribute Profile
  - ATT, Attribute Protocol
- GAP
  - 장치간 연결 제어
  - 동작구분
    - Advertising : 다른 장치에게 브로드캐스팅
    - Connection : 두 장치 간의 연결 방법
  - 역할 구분
    - Central Device : PC, Smart Phone
    - Peripheral Device : 센서 등



### ❖ BLE

- GATT
  - 장치간 통신제어
  - Service, Characteristic 을 이용, 데이터 전송 방법 정의
  - Attribute
    - 데이터 송수신의 최소 단위
  - Profile
    - 제품
  - Service
    - 특정 기능
  - Characteristic
    - 하나의 특성을 하나의 값으로 표현
- 역할 구분
  - GATT Server(Slave) :
    - 데이터를 제공하는 장치
    - 센서 장치
  - GATT Client
    - Central 장치
    - PC , Smart Phone
  - GATT Client →(요청) → GATT Server



# Bluetooth

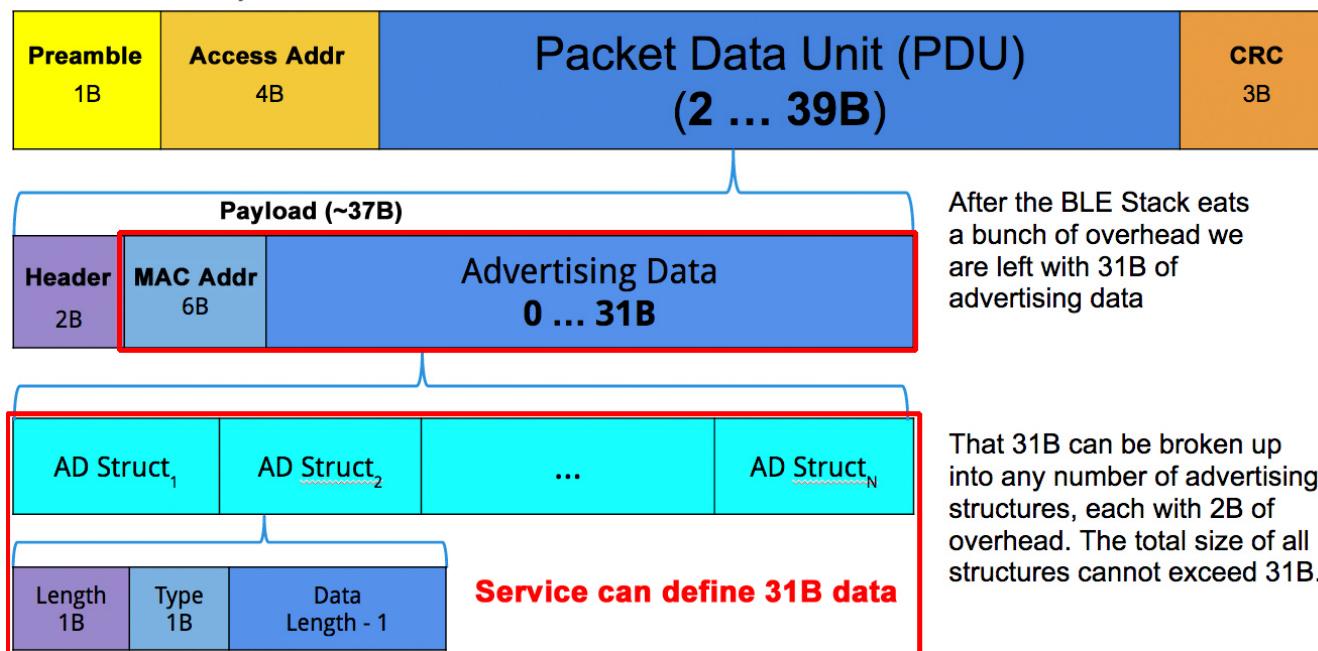
## Raspberry-Pi Communication

### ❖ Beacon

- Advertising 전용 장치
- iBeacon
  - 실내 측위 시스템을 위한 애플의 등록상표
  - PDU 내의 최대 31byte의 Payload



Over the air 47 Bytes are transmitted

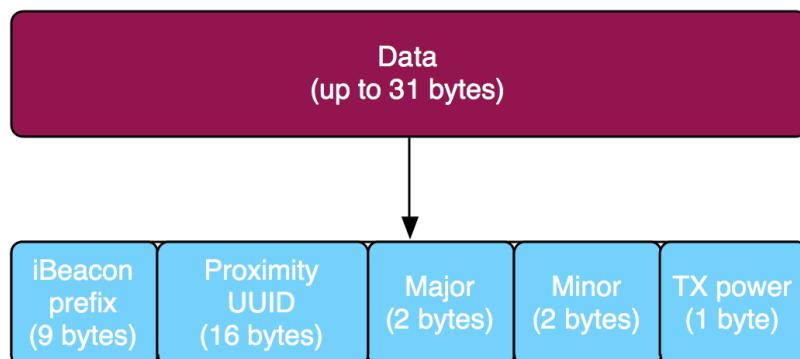


# Bluetooth

## Raspberry-Pi Communication

### ❖ iBeacon Data Frame

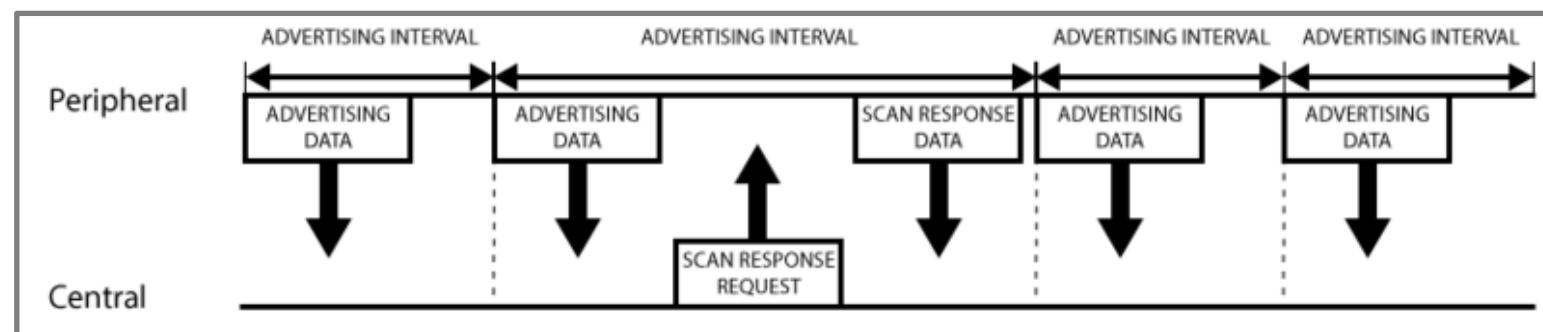
- Advertising Data
- PDU 내 최대 31Byte Payload
  - Major/Minor Number
    - 비콘 장치 식별
  - Tx Power
    - 장치 1m 거리에서 측정했을 때의 신호 강도
    - 거리 산출에 활용



Item	Byte	Desc
AD Flags	3	length, FlagType, FlagValue
AD Headers	2	length, 제조사 AD Type
Company ID	2	Apple : 0x4C00
Type ID	1	iBeacon : 0x02
Data Length	1	나머지 length
UUID	16	128bit 장치 식별 ID
Major Number	2	주 번호
Minor Number	2	부 번호
TX Power	1	신호 세기

### ❖ Advertising / Connection

- Advertising / Scanning
  - Advertising / Scan Response
  - Advertising Interval
  - Interval에 따른 전력 소모 차이
- Connection / Pairing
  - Advertising 종료
  - Scan 불가
  - GATT/Service/Characteristic 을 이용한 통신



- ❖ Raspberry-Pi 를 iBeacon 으로 만들기

- sudo hciconfig hci0 noscan
  - 스캔 중지
- sudo hciconfig hci0 leadv3
  - None Connectable LE Advertising, leadv0 : Connectable
- sudo hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 E2 0A 39 F4 73 F5 4B C4 A1 2F 17 D1 AD 07 A9 61 00 01 00 02 C8 00
  - Advertising 전송 명령
  - 0x08 : OGF, 블루투스 커맨드 그룹
  - 0x0008 : OCF, 비콘 전송 명령
  - 1E 02 01 : AD Flag(3Byte), 총길이 30byte, General Discoverable Mode
  - 1A 1A : AD Headers(2Bytes), 나머지 길이 26, 제조사 AD Type
  - FF 4C 00 02: 제조사 스펙 데이터, 애플 ID(4C 00), Typeld(02:iBeacon)
  - 15 : 나머지 길이 : 21
  - E2 0A 39 F4 73 F5 4B C4 A1 2F 17 D1 AD 07 A9 61 : UUID
  - 00 01 : Major Number
  - 00 02 : Minor Number
  - C8 : Tx Power
  - 00 : Null EOF
- sudo hciconfig hci0 noleadv
  - Advertising 전송 중지

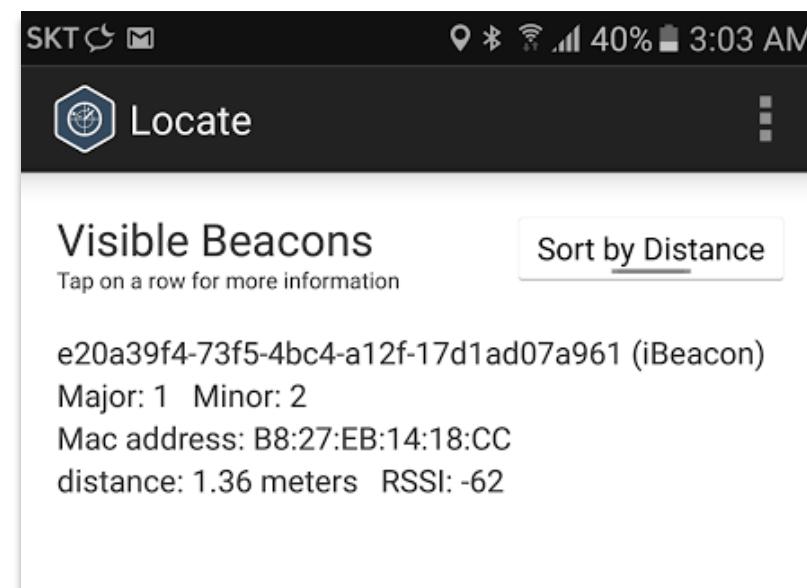
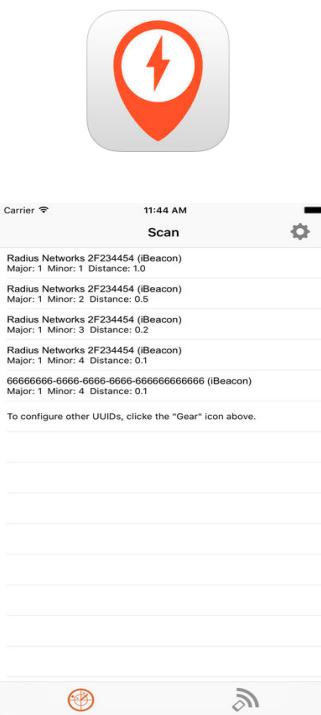
```
[pi@raspberrypi:~ $ sudo hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02
15 E2 0A 39 F4 73 F5 4B C4 A1 2F 17 D1 AD 07 A9 61 00 00 00 00 C8 00
< HCI Command: ogf 0x08, ocf 0x0008, plen 32
  1E 02 01 1A 1A FF 4C 00 02 15 E2 0A 39 F4 73 F5 4B C4 A1 2F
  17 D1 AD 07 A9 61 00 00 00 00 C8 00
> HCI Event: 0x0e plen 4
  01 08 20 00
```

# Bluetooth

## Raspberry-Pi Communication

### ❖ Mobile App으로 확인

- Locate Beacon App , Radius Networks
- Android
  - <https://play.google.com/store/apps/details?id=com.radiusnetworks.locate&hl=ko>
- iPhone
  - <https://itunes.apple.com/kr/app/locate-beacon/id738709014?mt=8>

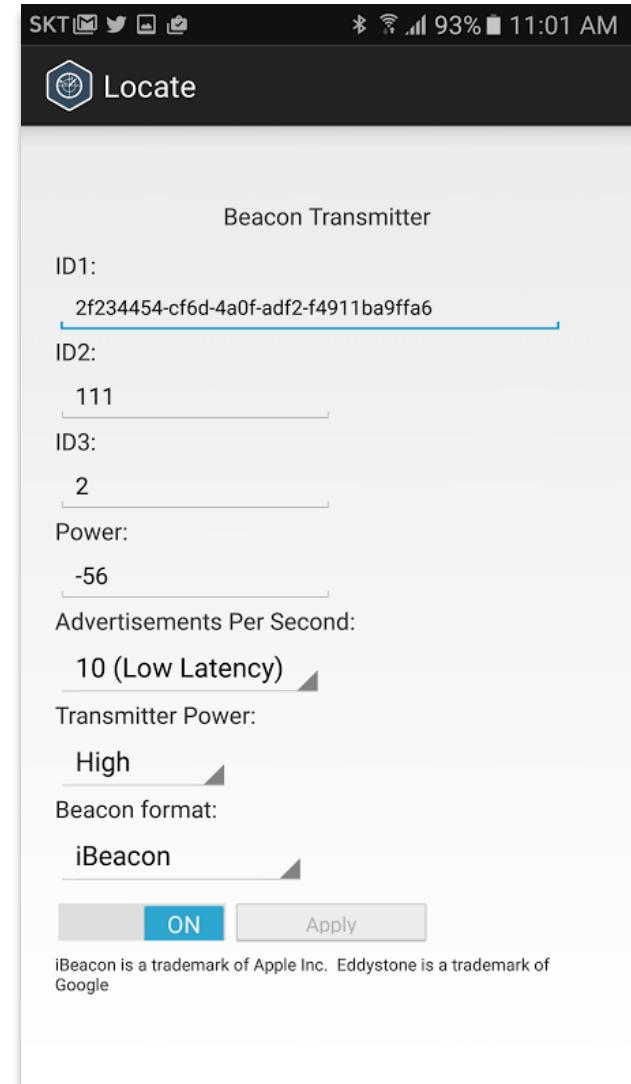


# Bluetooth

## Raspberry-Pi Communication

### ❖ RaspberryPi Beacon Scanner

- iBeacon Device
  - Locate App, Beacon Transmitter
    - Android 5.0+
  - Estimote
    - <http://estimote.com/>
  - SensorTag
    - <http://www.ti.com/sensortag>
  - HM-10
    - <http://www.jnhuamao.cn/>
    - Ti CC2540/2541 SoC 기반



# Bluetooth

## Raspberry-Pi Communication

### ❖ RaspberryPi ¶ Beacon Scanner

- hcidump 설치
  - sudo apt-get install hcidump
- le scan
  - sudo hcitool lescan --duplicate
    - 동일한 장치 필터링 제외
  - sudo hcidump -raw
    - lescan 정보를 자세히 출력
    - UUDI, Major/Minor, Tx 분석

```
pi@raspberrypi:~ $ sudo hcitool lescan --duplicate
LE Scan ...
44:D7:C7:9A:81:0B (unknown)
```

```
pi@raspberrypi:~ $ sudo hcidump --raw
HCI sniffer - Bluetooth packet analyzer ver 5.23
device: hci0 snap_len: 1500 filter: 0xffffffff
< 01 0B 20 07 01 10 00 10 00 00 00
> 04 0E 04 01 0B 20 00
< 01 0C 20 02 01 00
> 04 0E 04 01 0C 20 00
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 BA
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B8
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B5
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B9
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B7
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B8
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
2F 23 44 54 CF 6D 4A 0F AD F2 F4 91 1B A9 FF A6 00 6F 00 02
C8 B8
> 04 3E 27 02 01 03 01 4B 27 2F 89 84 6B 1B 1A FF 4C 00 02 15
```

### ❖ RaspberryPi ¶ Beacon Scanner in Python

- iBeacon Scanner
  - <https://github.com/switchdoclabs/iBeacon-Scanner->
  - <http://www.switchdoc.com/> BeaconAir Project의 일부
- 소스 다운로드 및 실행
  - git clone <https://github.com/switchdoclabs/iBeacon-Scanner->
  - cd iBeacon-Scanner-
  - sudo python testblescan.py
    - MAC addr, UUID, Major, Minor, Tx Power, RSSI

```
pi@raspberrypi:~/iBeacon-Scanner- $ sudo python testblescan.py
ble thread started
-----
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-70
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-73
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-70
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-73
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-69
b8:27:eb:14:18:cc,e20a39f473f54bc4a12f17d1ad07a961,1,2,-56,-50
b8:27:eb:14:18:cc,00cc1814eb27b80d0c09726173706265,29298,31088,105,-47
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-73
7c:7b:3e:05:d4:4d,2f234454cf6d4a0fadf2f4911ba9ffa6,111,2,-56,-69
-----
```