비콘을 이용한 통신방법 보고서

작성자 조은희
cho108@krri.re.kr

목 차

비콘을 이용한 통신 방법 보고서

I. 교통카드 데이터 (KS X 6924)

(교통카드 데이터의 통신정보는 KS 규격 (KS X 6924) 에 따라 작성하였다.)

본 보고서에서는 교통카드의 지불 절차로 선불 IC카드의 지불 절차를 따른다. 선불 IC 카드는 지불을 거래시점에 지불하는데 사용되는 카드를 의미한다.

1-1. 교통카드 데이터 구조

거래명령어는 다음과 같다.

1. Initialize Card
2. Purchase Card

Initialize Card (단말기에서 선불 IC 카드로 가는 명령어)

a) 기능 : 거래초기화를 수행한다.

b) 명령어 구조

해당 명령어는 단말기에서 선불IC카드로 수신되는 명령어로 총 12 bytes 이다.

항목	설정값	크기(byte(s))	의미
CLA	0x90	1	class
INS	0x02	1	Instruction
P1	0x00	1	선불카드 거래
	0x01		선불카드 재거래
	0x02		선불카드 직전 거래 취소
P2	0x00	1	
Lc(Length of Command)	0x04	1	
Data	Mpda	4	거래금액
Le(Length of Expected data)	0x17	1	p1=0x00 일 때
	0x27		p2=0x01 일 때
	0x27		p3=0x02 일 때

해당 명령어를 받은 선불 IC카드는 지불 SAM으로 다음과 같은 응답을 보낸다.

응답구조는 P1의 설정값에 따라 다음과 같이 세가지 경우로 나눈다.

P1:0x00 : 지불거래시 (25 byte)

Data	ALG	1	사용암호 알고리즘 식별자
	VK	1	키 버전
	BAL	4	선불카드 잔액
	ID(center)	1	교통시스템 운영사 또는 카드발행사 식별자
	ID	8	카드 식별자
	NT	4	카드 거래카운터
	Sign1	4	카드 인증서명
	SW	2	Status word

P1:0x01 : 재거래시 (41 byte)

Data	ALG	1	사용암호 알고리즘 식별자
	VK	1	키 버전
	BAL	4	선불카드 잔액
	ID(center)	1	교통시스템 운영사 또는 카드발행사 식별자
	ID	8	카드 식별자
	NT	4	카드 거래카운터
	ID(SAM)	8	직전 거래의 SAM 식별자
	M(PDA)	4	직전 거래의 거래 금액
	NT'	4	직전 거래의 거래 카운트
	Sign1	4	카드의 인증서명
	SW	2	status word

P1:0x02 : 지불 거래 취소시 (41 byte)

Data	ALG	1	사용암호 알고리즘 식별자
	VK	1	키 버전
	BAL	4	선불카드 잔액
	ID(center)	1	교통시스템 운영사 또는 카드발행사 식별자
	ID	8	카드 식별자
	NT	4	카드 거래카운터
	ID(SAM)	8	직전 거래의 SAM 식별자
	M(PDA)	4	직전 거래의 거래 금액
	NT'	4	직전 거래의 거래 카운트
	Sign1	4	카드의 인증서명
	SW	2	status word

카드를 초기화하고 응답을 한 이후 지불을 하기위한 명령어는 다음과 같다.

Purchase card

a) 기능 : 거래를 수행한다.

b) 명령어구조

해당 명령어는 부가 데이터가 없으면 모두 P1= 0x00, 0x01 일 때 24 bytes로 동일하며, P1=0x02 즉, 거래 취소시에는 23 bytes 이다.

항목	설정값	크기(byte(s))	의미
CLA	0x90	1	class
INS	0x04	1	Instruction
P1	0x00	1	선불카드 거래
	0x01		선불카드 재거래
	0x02		선불카드 직전 거래 취소
P2	0x00	1	부가 데이터 없음
	0x##		부가 데이터 있음
Lc(Length of Command)	0x##	1	데이터의 길이
Data	##	##	데이터(P1에 의해 정의된 데이터+부가 데이터)
Le(Length of Expected data)	0x04	1	S3(P1=0x00,0x01일 경우)
	null	null	P1=0x02 일 때

명령어에 포함되어 있는 Data 구조 (부가 데이터가 없으면 모두 18bytes로 동일)

P1:0x00

Data	ID	8	SAM의 식별자
	NT	4	SAM의 거래 카운터
	SC	2	SAM 의 상태코드 = “거래 중”
	Sign2	4	SAM의 인증 서명
	부가데이터	xx	P2 값에 의해 정의된 데이터

P1:0x01

Data	ID	8	SAM의 식별자
	NT	4	SAM의 거래 카운터
	SC	2	SAM 의 상태코드 = “거래 중”
	Sign2	4	SAM의 인증 서명
	부가데이터	xx	P2 값에 의해 정의된 데이터

P1:0x02

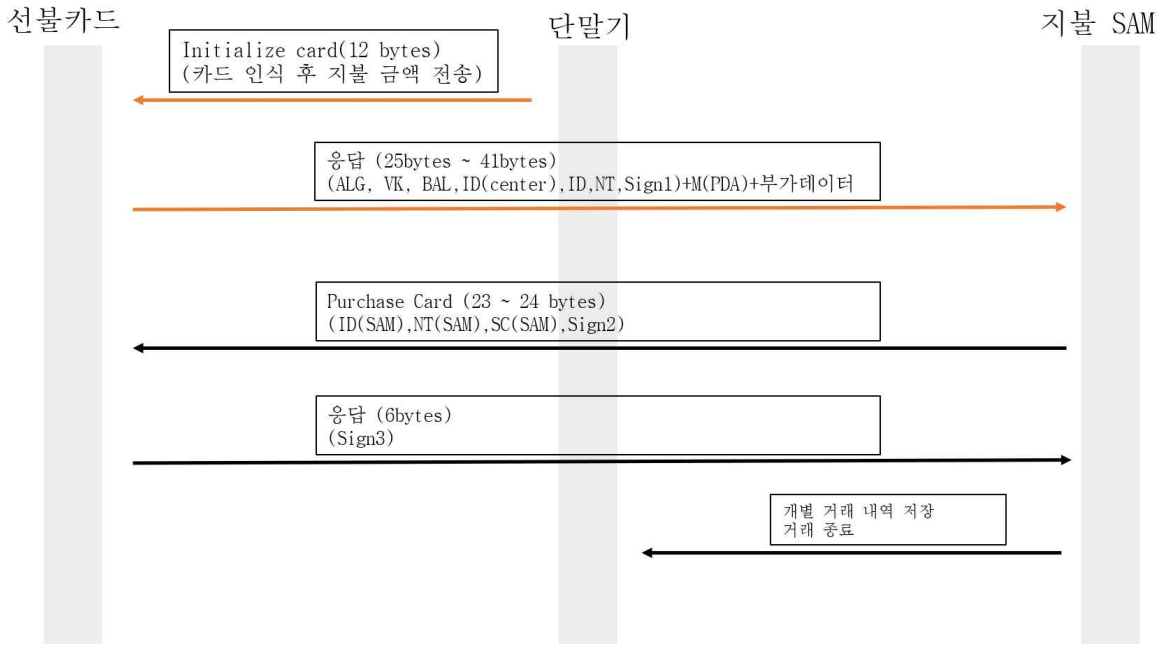
Data	ID	8	SAM의 식별자
	NT	4	SAM의 거래 카운터
	SC	2	SAM 의 상태코드 = “취소 진행”
	Sign2	4	SAM의 인증 서명
	부가데이터	xx	P2 값에 의해 정의된 데이터

해당 명령어를 받은 선불 IC카드는 SAM 에 다음과 같은 응답구조로 응답을 한다. (6bytes)

Data	Sign3	4	서명3(P1=0x02 일 경우 null)
SW	0xXX XX	2	status word

1-2. 교통카드 데이터 흐름

선불IC 카드의 거래 프로토콜은 다음과 같다.



[그림 1 – 선불 IC 카드의 지불 거래 프로토콜]

II. Beacon

2-1. Beacon 종류

a) iBeacon

애플이 2013년에 발표한 iBeacon은 BLE 4.0 의 advertising packet 전송 표준을 활용하여, IOS 기기에 적용한 것이다. 안드로이드의 경우 4.3 젤리빈 버전(2013년 4월 출시)부터 지원한다.

b) no-iBeacon

1. AltBeacon: Radius Newroks 사에서 오픈한 규격으로 애플의 iBeacon 규격을 보다 유연성있게 재구성한 비콘이다.
2. UriBeacon: Google의 Physical Web 프로젝트와 연결되어 특정 URI 가 포함된 Advertising Data를 브라우저를 통해 바로 정보를 보여주는 방식이다.
3. Placedge: 삼성이 자사의 스마트폰인 갤럭시폰에 관련 기능을 지원하면서 해당 서비스 App이 설치되어 있지않아도 이를 바로 연결해 주도록 지원하고 쿠폰, 콘텐츠 등을 전달하는 플랫폼을 별도로 개발하지 않아도 가능하도록 서버 플랫폼을 함께 제공하고 있다.
4. Eddystone(에디스톤): 2015sus 구글에서 발표했으며 오픈소스 프로젝트이다. 차이점은 에디스톤의 전송패킷은 Eddystone-UID, Eddystone-URL, Eddystone-TML 세가지 타입이 있다. 기존 규격을 지키면서 유연한 구조로 IOS와 Andoird 모두에게 잘 동작하는 것을 목표로 두고 있다.

2-2. iBeacon의 데이터 구조

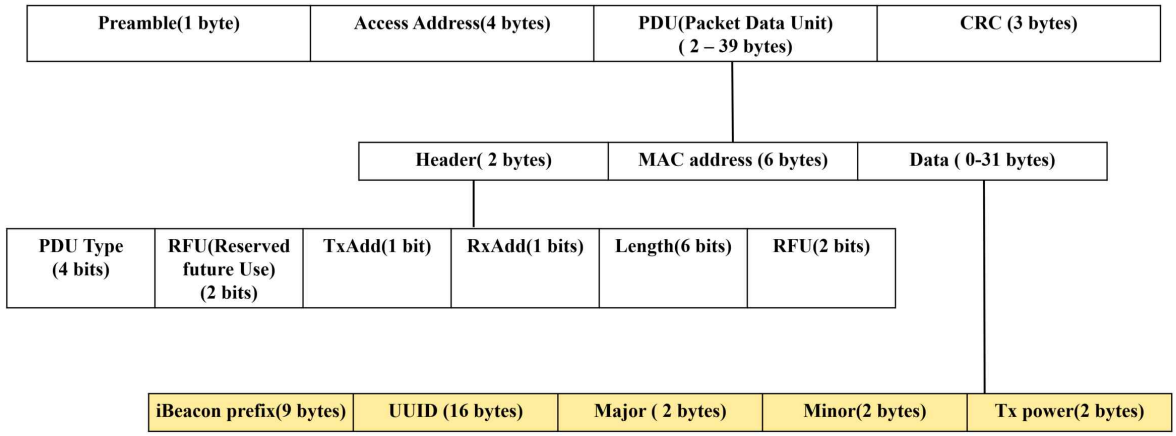
본 실험에서는 블루투스 패킷으로 iBeacon을 사용하였다.

iBeacon의 Advertising Data의 총 사이즈는 47 bytes로, 이 중 PDU(Protocal Data Unit) 중 iBeacon에서 정의한 Data 규격은 총 31 bytes 이다.

9 bytes 는 iBeacon prefix로 iBeacon의 규격을 따른다면 수정이 불가능하다. 이후 UUID, major,

minor 값은 각각 16 bytes, 2 bytes, 2 bytes 로 모두 수정이 가능한 값이며, 마지막에 있는 2bytes는 Tx power 이다.

따라서 총 20 bytes가 수정 가능한 데이터이다. 앞에서 설명했던 교통카드 데이터의 응답크기가 최소 6 bytes에서 최대 41 bytes 이기 때문에 차후 패킷의 갯수를 늘리거나 데이터의 크기를 조절해야 할 것으로 보인다.



[그림 2 - iBeacon의 패킷 구조]

```
uuid: OK+DISC:18010215:BB00C127101122223333F4911BA9FFA6:0005000AC5:43D73996347B:-060
This is a BTAG
```

[그림 3 - iBeacon 예시]

아두이노에서 수신한 iBeacon의 데이터는 [그림 3]과 같다. 순서대로 데이터를 나열하여 분석하면 다음과 같다.

BB00C127101122223333F4911BA9FFA6 : iBeacon 의 UUID

0005: major 값

000A: minor 값

C5 : 신호 세기

43D73996347B : MAC 주소

-060: RSSI 값

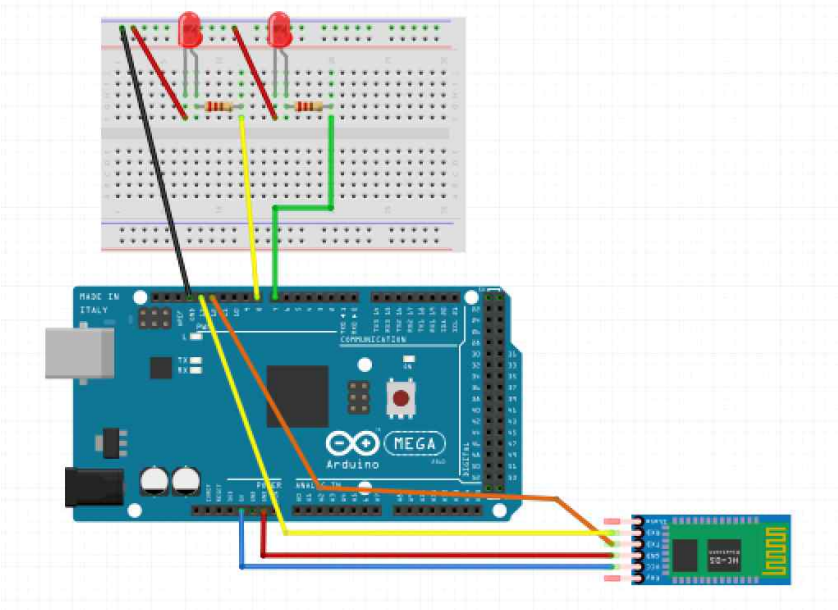
3. 실험 보고

3-1. 실험환경

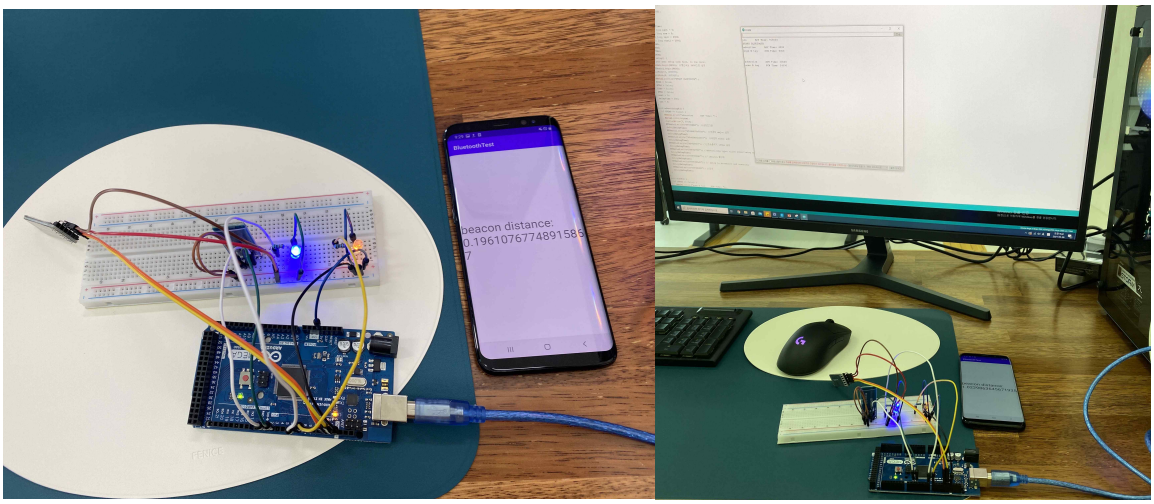
분류	종류	비고
블루투스	HM-10	
아두이노	아두이노 2560 MEGA	
스마트폰	삼성 갤럭시 S8	안드로이드 버전: 9

[표1 - 실험환경]

아두이노에서 HM-10 모듈을 사용하여 회로를 구성하였으며, 비콘의 송수신을 확인하기 위해 안드로이드 어플리케이션을 간단히 제작하였다.



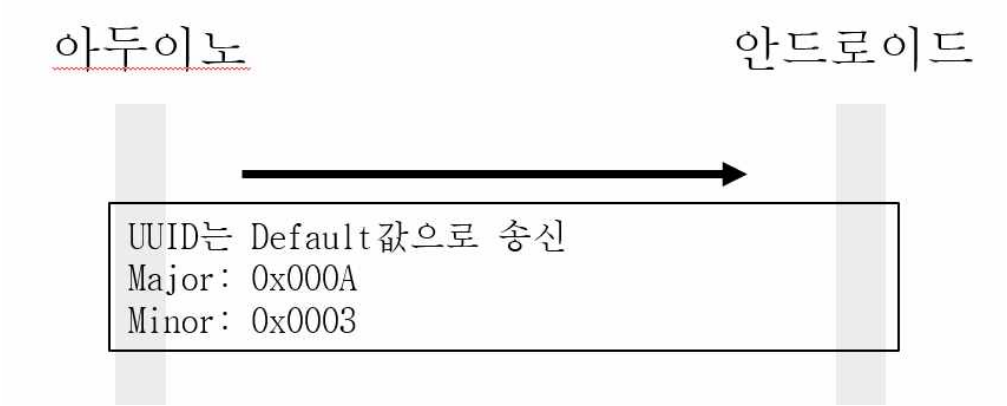
[그림 4 - HM-10을 사용하여 구성한 회로]



[그림 5 - 실제 구성한 아두이노 / 안드로이드 어플리케이션]

3-2. 실험과정

1. 비콘 송신



[그림 6 - 아두이노 송신/안드로이드 수신 구성도]

A 태그를 major 값에 붙인 상태로 advertising packet을 송신하면, 안드로이드 어플리케이션에서 비콘신호를 수신하여, 해당 비콘 데이터의 major 값이 A이면 비콘과의 거리를 출력하도록 하였다.

```
void advertisingA() {
  if (BTAG == false) {
    Serial.print("advertise      ADV Time ");
    Serial.println(now);
    digitalWrite(7, HIGH);
    BTSerial.write("AT+RENEW"); //공장초기화
    delay(delayTime);
    BTSerial.write("AT+MARJ0x000A"); //비콘의 major 설정
    delay(delayTime);
    BTSerial.write("AT+MINO0x0003"); //비콘의 minor 설정
    delay(delayTime);
    BTSerial.write("AT+ADVI0"); //신호송출주기 100ms 설정
    delay(delayTime);
    BTSerial.write("AT+ADTY2"); //advertising type: allow advertising and scan response
    delay(delayTime);
    BTSerial.write("AT+IBEA1"); // iBeacon 활성화
    delay(delayTime);
    BTSerial.write("AT+DELO1"); // allow to broadcast and scanning
    delay(delayTime);
    BTSerial.write("AT+RESET"); //리셋
    delay(delayTime);
  }
}
```

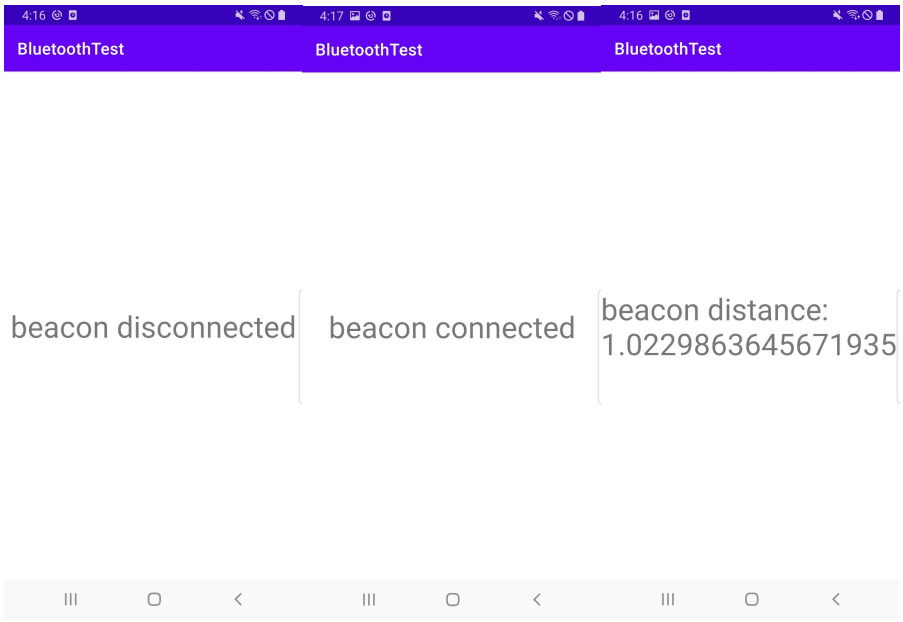
[그림 7 - 송신모드 변경시 사용되는 AT 명령어]

안드로이드 어플리케이션 단에서는 비콘이 인식되지 않으면 “beacon disconnected”, 비콘이 인식되면 “beacon connected”, 인식된 비콘의 major 값이 A 이면 비콘과의 거리(m)를 출력하게끔 구현하였다.

```
beaconManager = BeaconManager.getInstanceForApplication(this);//비콘매니저 초기화
beaconManager.getBeaconParsers().add(new BeaconParser().setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));
beaconManager.bind( consumer: this); // 비콘 탐지 시작

beaconManager.addRangeNotifier(new RangeNotifier() {
    @Override
    public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {
        List<Beacon> list = (List<Beacon>) beacons;
        if (beacons.size() > 0) {
            handler.sendEmptyMessage( what: 0);
           // textView.setText("beacon connected");

            Log.d(TAG2,  msg: ":::::This ::UUID:: of beacon  :  " + beacons.iterator().next().getId1().toString() + ":::::");
            Log.d(TAG2,  msg: ":::::This ::DISTANCE:: of beacon  :  " + beacons.iterator().next().getDistance() + ":::::");

            int MAJOR=beacons.iterator().next().getId2().toInt();
            int MINOR=beacons.iterator().next().getId3().toInt();
            distance=beacons.iterator().next().getDistance();

            if(MAJOR==0x000A && MINOR==3){
                handler.sendEmptyMessage( what: 2);
            }else{
                handler.sendEmptyMessage( what: 0);
            }
        }
        else if(beacons.size()==0){
            handler.sendEmptyMessage( what: 1);
        }
    }
});
```

[그림 8 - 비콘 수신 안드로이드 코드]

비콘 탐지가 시작되고 비콘의 범위 안으로 인식되면 didRangeBeaconsInRegion 함수가 실행된다.

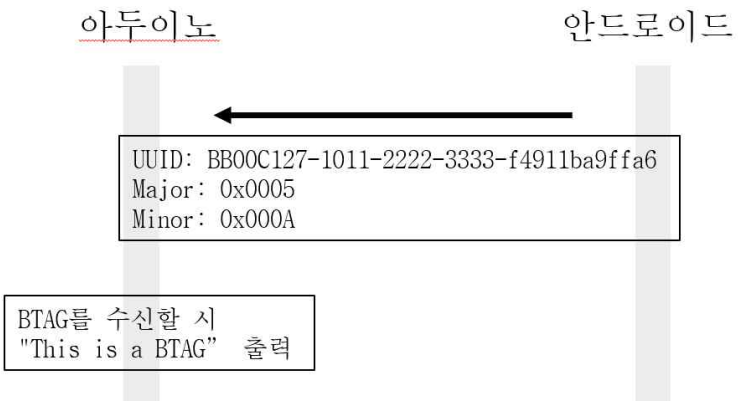


[그림 9 - 비콘 통신 안드로이드 테스트 (비콘이 인식되지 않을 때/ 비콘이 인식될 때 /인식된 비콘의 Major 값이 A 일 때)]

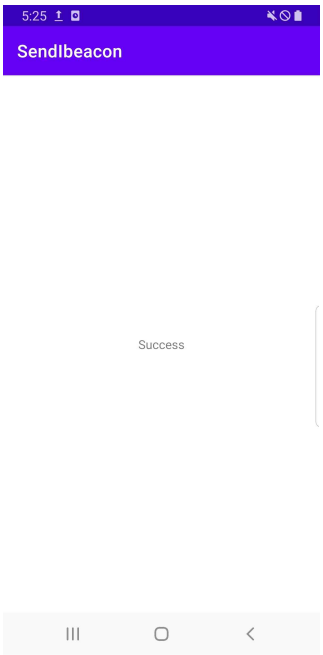
2. 비콘 수신

안드로이드 어플리케이션에서는 원하는 UUID, major, minor 값을 지정하여 비콘을 송신할 수 있다. 본 실험에서는 major 값은 5, minor 값은 10, UUID는 임의로 지정하여 첫 번째 자리에 B 태그를 붙여 송신하였다. 송신이 제대로 작동하면 화면에 “Success”를 출력한다.

아두이노에서는 UUID의 첫 번째 자리에 B 태그를 송신받을시 “This is a BTAG” 라는 값을 출력하도록 구현하였다.



[그림 10 - 안드로이드 송신/아두이노 수신 구성도]




```
Beacon beacon=new Beacon.Builder()
        .setId1("BB00C127-1011-2222-3333-f4911ba9ffa6")
        .setId2("5")
        .setId3("10")
        .setManufacturer(0x0118)
        .setTxPower(-59)
        .setDataFields(Arrays.asList(new Long[] {01}))
        .build();
```

[그림 11 - 안드로이드의 비콘 송신 어플 코드 - Id1: UUID, Id2: Major, Id3: Minor]

```
void ScanB() {
  if (BTAG == false) {
    Serial.print("scan B tag     SCN Time: ");
    Serial.println(now);
    digitalWrite(8, HIGH);
    BTSerial.print("AT+ROLE1");
    delay(delayTime);
    BTSerial.print("AT+IMME1");
    delay(delayTime);
    BTSerial.print("AT+START");
    delay(delayTime);
    BTSerial.print("AT+RESET");
    delay(1000);
    BTSerial.print("AT+DISI?");
    // Serial.println("AT+DISI?");
    BTAG = true;
  }
```

[그림 12 - 아두이노 수신 모드 변경시 사용되는 AT 명령어 총 5가지]

AT+DISI? 명령어 이후 BTAG를 true 값으로 변경한다. 응답으로 오는 문자열 중, “OK+DISC:” 즉, 비콘을 출력하는 응답 시작 문자열의 위치를 찾아 해당 위치에서 바이트 수를 계산하여 UUID의 가장 첫자리가 B 태그인지 is_Start 함수내에서 지속적으로 확인한다. 이후 “OK+DISCE” 즉, 모든 비콘을 둘러보고 마지막으로 응답하는 문자열이 나올 때까지 계속한다.

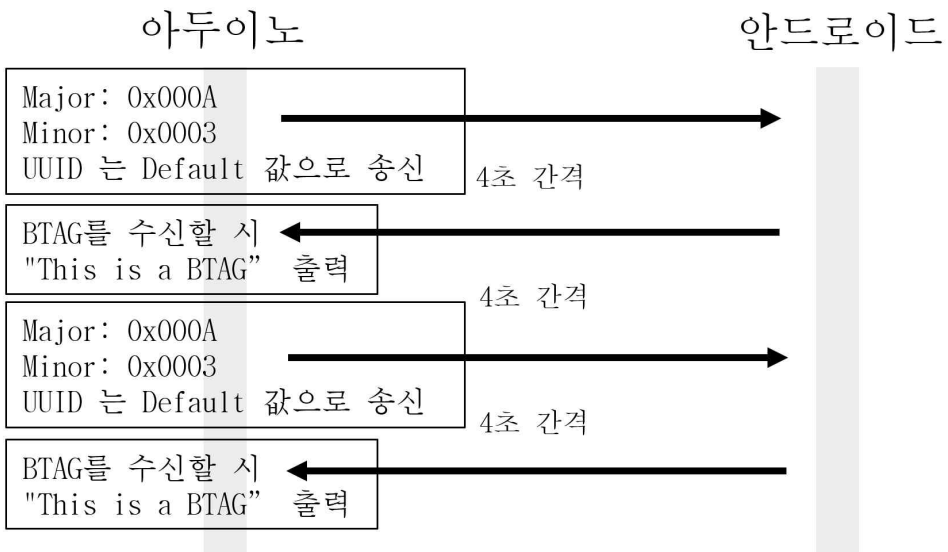
```
else if (BTAG == true) {
  String str_toString = String((char*)str);
  int buffer_address = (int)str;
  int positions = strstr((char*)str, "OK+DISC:");
  int positions2 = strstr((char*)str, "OK+DISCE");
  //String po = strstr((char*)str, "OK+DISC:");
  //String po2 = strstr((char*)str, "OK+DISCE");

  while (positions < positions2) {
    is_Start(positions - buffer_address, str_toString);
    positions += 78;
  }
  BTAG = false;
  memset(str, '\0', cnt-1);
  cnt = 0;
}
```

[그림 13 - UUID 추출]

```
uuid: OK+DISC:18010215:BB00C127101122223333F4911BA9FFA6:0005000AC5:43D73996347B:-060
This is a BTAG
```

[그림 14 - 아두이노의 비콘 스캔 테스트]

3. 비콘 송수신(4초 간격)



[그림 15 - 아두이노 송수신]

아두이노에서 비콘의 송수신을 약 4초간격으로 설정하였다. 이는 각각 송신과 수신모드로 변경시에 AT+Command 로 각각의 모드를 설정해야하는 것이 필수적이기 때문이다. 한 명령어에 대해 블루투스 모듈(HM-10)의 설정시간이 약 400ms 로 측정되기 때문에 송신모드의 경우 모드 변경시까지 약 3.2초가 필요하다. 따라서 송수신간의 간격을 4초로 설정하였다. 이보다 짧아질 경우 비콘의 송신, 수신모드로의 변경이 제대로 되지않아 어플리케이션에서 비콘신호가 제대로 수신되지 않

는 것을 확인할 수 있다.

비콘의 송수신 모드 변경시 각각의 실행시간을 출력해보았다. 4초간의 간격으로 송신모드에서 수신모드로 변경될 때는 4초마다 제대로 변경되는 것을 볼 수 있지만, 수신모드일 때는 스캔을 하는 작업(AT+DISI?)에서 시간을 소요시키는 것을 볼 수 있다. 수신모드로 변경 후, 약 12초 후에 송신모드로 변경된다.

[그림 16 - 송수신 모드 변경시 걸리는 시간 출력]

3-3. 실험 결과

아두이노에서 비콘의 송신모드와 수신모드를 4초마다 번갈아 가게 하였을 때, 수신모드일 때는 주변의 비콘을 스캔하는 명령어로 인하여 4초가 아닌 12초 후에 송신모드로 변경되는 것을 볼 수 있다. 지연시간이 너무 오래 걸리는 문제점을 해결하기 위해 블루투스 모듈 2개를 아두이노에 장착하여 송수신 모드를 변경하지 않은 상태로 비콘을 조절할 계획이다.

또한, 안드로이드 어플리케이션의 비콘 송수신과정의 코드가 통합되지 않아 각각의 어플리케이션을 사용하기 때문에 이를 통합하는 과정이 필요하다.

AT-COMMAND

아두이노 단에서 블루투스 모듈을 설정하기 위한 명령어 모음

명령어	의미
AT+ADDR?	블루투스 모듈 맥주소 확인
AT+AD[para1]?? AT+AD[para1][para2]	화이트리스트(whitelist) 맥주소 확인 화이트리스트(whitelist) 맥주소 설정 [Para1]:1,2,3 [Para2]:맥주소
AT+FILT? AT+FILT[Para]	블루투스 기기 필터링 검색 확인/설정 [Para]:0 : 모든 BLE 모듈을 찾는다 [Para]:1 : HM 모듈만을 찾는다 Default:1
AT+NAME? AT+NAME[Para]	블루투스 모듈 이름 확인/설정 Default: HMSoft
AT+PASS? AT+PASS[Para]	비밀번호(pin code) 확인/설정 Dafault: 000000
AT+POWE? AT+POWE[Para]	모듈 신호세기 확인/설정 [Para]:0~3 0: -23dbm 1:-6dbm 2: 0dbm 3: 6dbm Default:2
AT+SLEEP	sleep 모드 설정(peripheral:advertising mode 역할에서 사용)
AT+PWRM? AT+PWRM[Para]	모듈 sleep time 확인/설정(peripheral:advertising mode 역할에서 사용)
AT+RESET	모듈 재시작
AT+RENEW	공장초기화 상태로 모듈값 복구
AT+PIO1? AT+PIO1[Para]	System LED 점등 관련 확인/설정 Para1:0,1 0: Unconnected Output 500ms High 500ms Low, Connected output High. 1: Unconnected output Low, Connected output High. Default:0
AT+BATC? AT+BATC[para]	배터리 측정 스위치 확인/설정 [Para]:0~1 0: Off 1: On Default: 0
AT+BATT?	배터리 정보 확인 100% = 3V, 0% = 2V
AT+TEMP?	모듈 온도 확인

AT+MODE?	모듈 동작모드 설정 0: transmission Mode 1: PIO collection Mode +Mode 0 2: Remote Control Mode +Mode 0 Default: 0
AT+ROLE?	모듈 master(Central : scan), Slave(Peripheral : advertise) 역할 확인/설정 0: Peripheral 1: Central Default: 0
AT+ADTY? AT+ADTY[Para]	Advertising 유형 확인/설정 [Para]: 0~3 0: Advertising ScanResponse, Connectable (블루투스 페어링 모드) 1: Only allow last device connect in 1.28 seconds (블루투스 페어링 모드) 2: Only allow advertising and scanResponse (비콘 모드+ 스캔 응답) 3: Only allow advertising (비콘 모드) Default: 0
AT+BAUD? AT+BAUD[Para]	Baud Rate 확인/ 설정 [Para]: Baud rate No. 0 ------- 9600 1 ------- 19200 2 ------- 38400 3 ------- 57600 4 ------- 115200 5 ------- 4800 6 ------- 2400 7 ------- 1200 8 ------- 230400 Default: 0(9600)
AT+SHOW? AT+SHOW[Para]	장치이름 스캔여부 확인/설정 (AT+FILT0 실행 후 사용) [Para]: 0,1 0: Don't show name 1: show name Default: 0 (AT+SHOW1 설정시 AT+DISC? 명령어를 사용하면 이름에 대한 정보가 보여진다)

AT+ADVI? AT+ADVI[Para]	'advertising' 간격 확인/설정 [Para]:0~F 0: 100ms 1: 152.5ms 2: 211.25ms 3: 318.75ms 4: 417.5ms 5: 546.25ms 6: 760ms 7: 852.5ms 8: 1022.5ms 9: 1285ms A: 2000ms B: 3000ms C: 4000ms D: 5000ms E: 6000ms F: 7000ms Default: 0
AT+IMME? AT+IMME[Para]	모듈동작 타입 설정(Central:Scan 역할에서 사용) [Para]: 0,1 1: when module is powered on, only respond the AT command, don't do anything until AT+START is received or can use AT+CON, AT+CONNL 0: when power on, work immediately Default: 0
AT+START	동작 명령(AT+IMME1 이 설정되었을 때)
AT+RSSI?	모듈의 RSSI 값 확인(장치와 연결된 후)
AT+UUID? AT+UUID[Para]	모듈 UUID 확인/설정 [Para]:0x0001 ~ 0xFFFE Default: 0xFFE0
AT+TYPE? AT+TYPE[Para]	모듈 Bond 모드 확인/설정 [Para]:0~2 0: not need PINcode 1: pair not need PIN 2: pair with PIN 3: Pair and bond Default: 0

AT+IBEA? AT+IBEA[Para]	블루투스 모듈 iBeacon 모드 스위치(peripheral: advertising 역할에서 사용) [Para]: 0,1 0: Turn off iBeacon 1: Turn on iBeacon Default: 0 Default iBeacon UUID : 742788DA-B644-4520-8F0C-720EAF059935
AT+DELO[Para]	iBeacon 전개 모드 설정 [Para]: 1,2 1: Allowed to broadcast and scanning 2: Only allow broadcast OK+DELO[Para] 응답 이후 500ms 가 지나면 reset 된다.
AT+DISI?	iBeacon 장치 검색 시작(AT+IMM1 와 AT+ROLE1 명령어 실행 후) *응답구조 OK+DISC:[para1]:[para2]:[para3]:[para4]:[para5] [para1]: factory ID [para2]: iBeacon ID [para3]: Major 값, minor 값, 측정세기 [para4]: MAC [para5]: RSSI
AT+IBE0? AT+IBE1? AT+IBE2? AT+IBE3? AT+IBE0[Para1] AT+IBE1[Para2] AT+IBE2[Para3] AT+IBE3[Para4]	iBeacon UUID 값 확인/설정 [Para1]: 00000001 ~ FFFFFFFE Default: 74278BDA [Para2]: 00000001 ~ FFFFFFFE Default: B6444520 [Para2]: 00000001 ~ FFFFFFFE Default: 8F0C720E [Para2]: 00000001 ~ FFFFFFFE Default: AF059935
AT+MARJ? AT+MARJ[Para]	iBeacon major 값 확인/설정 [Para]: 0x0001~ 0xFFFE Default: 0xFFE0
AT+MINO? AT+MINO[Para]	iBeacon minor 값 확인/설정 [Para]: 0x0001~ 0xFFFE Default: 0xFFE1
AT+MEA?? AT+MEA[Para]	iBeacon 측정 세기 확인/설정 [Para]: 0x00 ~ 0xFF Default: 0xC5

참고문헌

- KS X 6924
- https://m.blog.naver.com/xisaturn/220712028679 (For AT Command)
- https://tech-people.github.io/2019/11/13/beacon-ble/ (For beacon information)