SNMP Android Client

2015410012 김현섭

목차

1. 개발 환경

개발 환경
Android Studio 3.1.2
Build #AI-173.4720617, built on April 14, 2018
JRE: 1.8.0_152-release-1024-b01 x86_64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Mac OS X 10.13.4

실행 환경
SM-G965N
Android 8.0.0 (API 26)

프로젝트 환경
compileSDKVersion: 26
minSDKVersion: 19
targetSDKVersion: 26
Gradle Version: 3.1.2

2. SNMPv2 패킷 구조

<table>
<tr><td colspan="7">SNMPv2</td></tr>
<tr><td colspan="2">version</td><td colspan="5">community</td><td colspan="3">PDU</td></tr>
<tr><td colspan="2"></td><td>type</td><td>request ID</td><td>error status</td><td>error index</td><td colspan="3">variables (list)</td></tr>
<tr><td colspan="6"></td><td colspan="3">variable</td></tr>
<tr><td colspan="6"></td><td>oid</td><td>value</td></tr>
</table>

SNMP

version : SNMP version을 의미한다. SNMPv1의 값은 0이고 SNMPv2의 값은 1이다.
community : 라우터에 접근하기 위한 비밀번호 같은 것이다.
PDU : Protocol Data Unit으로 SNMP 패킷의 데이터를 나타낸다.

PDU

type : PDU의 타입이다. 자세한 값은 아래의 표로 대체한다.

Type	*Tag (Hex)*	*Type*	*Tag (Hex)*
GetRequest	**A0**	GetBulkRequest	**A5**
GetNextRequest	**A1**	InformRequest	**A6**
Response	**A2**	Trap (SNMPv2)	**A7**
SetRequest	**A3**	Report	**A8**

request ID : SNMP 패킷의 요청 번호이다.
error status : 패킷에 포함된 에러를 의미한다. 자세한 값은 아래의 표로 대체한다.

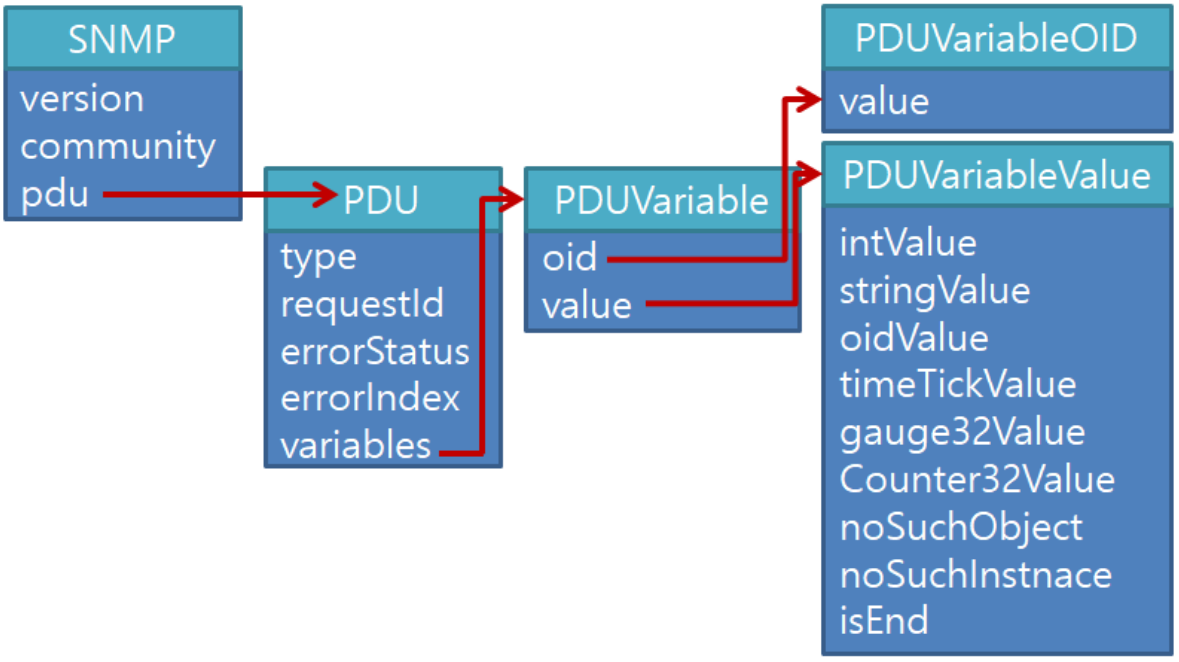
Status	*Name*	*Meaning*
0	noError	No error
1	tooBig	Response too big to fit in one message
2	noSuchName	Variable does not exist
3	badValue	The value to be stored is invalid
4	readOnly	The value cannot be modified
5	genErr	Other errors

error index : 에러가 있는 값의 index를 의미한다.
variables : 요청이나 요청한 값의 데이터 리스트이다.

Variable

oid : Agent에 존재하는 Object의 ID이다.
value : Object ID에 해당하는 Object의 값이다.

3. 자료구조



전체적으로 위와 같은 자료 구조로 데이터들이 관리 되고 있다.

```
public class SNMP implements BERSerializable {
    public int version;
    public String community;
    public PDU pdu;
```

SNMP 패킷은 위와 같이 3가지 멤버 변수를 가지고 있다.
version은 int, community는 String으로 관리하고 pdu는 PDU 클래스로 관리한다.

```
public class PDU implements BERSerializable {
    public PDUType type;
    public int requestId;
    public int errorStatus = 0;
    public int errorIndex = 0;
    public ArrayList<PDUVariable> variables;
```

PDU는 위와 같이 5가지 멤버 변수를 가지고 있다.
RequestID, ErrorStatus, ErrorIndex는 int로 관리한다.
PDU 타입은 PDUType이라는 Enum 클래스로 관리한다.

```
public enum PDUType implements BERSerializable {
    GET_REQUEST,
    GET_NEXT_REQUEST,
    GET_RESPONSE,
    SET_REQUEST;
```

PDUType은 GET_REQUEST, GET_NEXT_REQUEST, GET_RESPONSE,
SET_REQUEST로 총 4가지의 값으로 되어 있다.

```
public class PDUVariable implements BERSerializable {
    public PDUVariableOID oid;
    public PDUVariableValue value;
```

PDUVariable은 위와 같이 2가지 멤버 변수를 가지고 있다.
oid는 PDUVariableOID라는 클래스로 value는 PDUVariableVaue라는 클래스로 관리한다.

```
public class PDUVariableOID implements BERSerializable {
    public int[] value;
```

PDUVariableOID는 위와 같이 1가지의 멤버 변수를 가지고 있다.
ObjectID는 인코딩 디코딩 편리성을 위하여 String 보다 int array로 관리한다.

```
public class PDUVariableValue implements BERSerializable {
    public Integer intValue = null;
    public String stringValue = null;
    public PDUVariableOID oidValue = null;
    public Long timeTickValue = null;
    public Long gauge32Value = null;
    public Long counter32Value = null;

    public Boolean noSuchObject = null;
    public Boolean noSuchInstance = null;
    public Boolean isEnd = null;
```

PDUVariableValue는 위와 같이 9가지 멤버 변수를 가지고 있다.
어떤 값이 null이 아니냐에 따라 PDUVariableValue의 값이 결정이 된다.

4. SNMP 패킷 BER Encoding

```
public void encodeBER(OutputStream os) throws IOException {
    int payloadLength = getBERPayloadLength();

    BER.encodeHeader(os, BER.SEQUENCE, payloadLength);
    BER.encodeInteger(os, BER.INTEGER, version);
    BER.encodeString(os, BER.OCTETSTRING, community.getBytes());
    pdu.encodeBER(os);
}
```

위의 코드는 SNMP 클래스의 encodeBER 메서드 이다.

1. SNMP 패킷의 총 길이를 구한 뒤 이 것을 이용해서 Header를 인코딩
2. SNMP 버전 인코딩
3. Community String 인코딩
4. PDU 인코딩은 PDU의 멤버 메서드를 호출해서 수행

```
public void encodeBER(OutputStream os) throws IOException {
    BER.encodeHeader(os, type.getValue(), getBERPayloadLength());
    BER.encodeInteger(os, BER.INTEGER, requestId);
    BER.encodeInteger(os, BER.INTEGER, errorStatus);
    BER.encodeInteger(os, BER.INTEGER, errorIndex);

    int variablesLength = 0;
    for (PDUVariable v : variables)
        variablesLength += v.getBERLength();

    BER.encodeHeader(os, BER.SEQUENCE, variablesLength);
    for (PDUVariable v : variables)
        v.encodeBER(os);
}
```

위의 코드는 PDU 클래스의 encodeBER 메서드 이다.

1. PDU의 총 길이를 구하고 PDU type을 이용해서 Header를 인코딩
2. requestId, errorStatus, errorIndex를 인코딩
3. variable들의 길이를 더해서 variable들의 총 길이를 구한 뒤 인코딩
4. 각각의 variable 인코딩은 각각의 멤버 메서드를 호출해서 수행

```
public void encodeBER(OutputStream os) throws IOException {
    BER.encodeHeader(os, BER.SEQUENCE, getBERPayloadLength());
    oid.encodeBER(os);
    value.encodeBER(os);
}
```

위 코드는 PDUVariable 클래스의 encodeBER 메서드 이다.

1. PDUVarialbe의 총 길이를 구한 뒤 인코딩
2. PDUVariableOID 인코딩은 멤버 메서드를 호출해서 수행

3. PDUVariableValue 인코딩은 멤버 메서드를 호출해서 수행

```
public void encodeBER(OutputStream os) throws IOException {
    BER.encodeOID(os, BER.OID, value);
}
```

위 코드는 PDUVariableOID 클래스의 encodeBER 메서드 이다.

1. 단순히 int array로 되어 있는 ObjectID를 인코딩

```
public void encodeBER(OutputStream os) throws IOException {
    if (intValue != null)
        BER.encodeInteger(os, BER.INTEGER, intValue);
    else if (stringValue != null)
        BER.encodeString(os, BER.OCTETSTRING, stringValue.getBytes());
    else if (oidValue != null)
        oidValue.encodeBER(os);
    else if (timeTickValue != null)
        BER.encodeUnsignedInteger(os, BER.TIMETICKS, timeTickValue);
    else if (gauge32Value != null)
        BER.encodeUnsignedInteger(os, BER.TIMETICKS, gauge32Value);
    else if (counter32Value != null)
        BER.encodeUnsignedInteger(os, BER.TIMETICKS, counter32Value);
    else if (noSuchObject != null)
        BER.encodeHeader(os, BER.NOSUCHOBJECT, getBERPayloadLength());
    else if (noSuchInstance != null)
        BER.encodeHeader(os, BER.NOSUCHINSTANCE, getBERPayloadLength());
    else if (isEnd != null)
        BER.encodeHeader(os, BER.ENDOFMIBVIEW, getBERPayloadLength());
    else
        BER.encodeHeader(os, BER.ASN_NULL, getBERPayloadLength());
}
```

위 코드는 PDUVariableValue 클래스의 encodeBER 메서드 이다.

1. 각각의 값들을 차례로 검사하면서 null이 아니면 그 값에 대해 인코딩

5. SNMP 패킷 BER Decoding

```
public void decodeBER(BERInputStream is) throws IOException {
    BER.decodeHeader(is, new BER.MutableByte());

    version = BER.decodeInteger(is, new BER.MutableByte());

    community = new String(BER.decodeString(is, new BER.MutableByte()));

    pdu = new PDU();
    pdu.decodeBER(is);
}
```

위 코드는 SNMP 클래스의 decodeBER 메서드 이다.

1. Header를 디코딩
2. SNMP Version을 디코딩
3. SNMP Community String을 디코딩
4. PDU 객체 생성 후 멤버 메서드를 이용해 디코딩

```
public void decodeBER(BERInputStream is) throws IOException {
    BER.MutableByte pduType = new BER.MutableByte();
    BER.decodeHeader(is, pduType);
    type = PDUType.parse(pduType.getValue());

    requestId = BER.decodeInteger(is, new BER.MutableByte());
    errorStatus = BER.decodeInteger(is, new BER.MutableByte());
    errorIndex = BER.decodeInteger(is, new BER.MutableByte());

    int variableLength = BER.decodeHeader(is, new BER.MutableByte());

    int startPos = (int) is.getPosition();
    variables = new ArrayList<>();
    while (is.getPosition() - startPos < variableLength) {
        PDUVariable vb = new PDUVariable();
        vb.decodeBER(is);
        variables.add(vb);
    }
}
```

위 코드는 PDU 클래스의 decodeBER 메서드 이다.

1. PDU Header 디코딩해서 나온 값을 이용하여 PDU Type을 알아냄
2. RequestID, ErrorStatus, ErrorIndex를 디코딩
3. Variable List Header를 디코딩 해서 list의 총 길이를 구함
4. 구한 총 길이 만큼 PDUVariable을 생성하고 각각의 멤버 메서드를 이용하여 디코딩

```java
public void decodeBER(BERInputStream is) throws IOException {
    BER.decodeHeader(is, new BER.MutableByte());

    oid = new PDUVariableOID();
    oid.decodeBER(is);

    value = new PDUVariableValue();
    value.decodeBER(is);
}
```

위 코드는 PDUVariable 클래스의 deocdeBER 메서드 이다.

1. PDUVariable Header를 디코딩
2. PDUVariableOID 객체를 생성 후 멤버 메서드를 이용하여 디코딩
3. PDUVariableValue 객체를 생성 후 멤버 메서드를 이용하여 디코딩

```java
public void decodeBER(BERInputStream is) throws IOException {
    value = BER.decodeOID(is, new BER.MutableByte());
}
```

위 코드는 PDUVariableOID 클래스의 decodeBER 메서드 이다.

1. 디코딩 하여 int array로 되어 있는 ObjectID를 얻음

```java
public void decodeBER(BERInputStream is) throws IOException {
    byte type = is.getBuffer().array()[(int) is.getPosition()];
    if (type == BER.INTEGER) {
        intValue = BER.decodeInteger(is, new BER.MutableByte());
    } else if (type == BER.OCTETSTRING) {
        stringValue = new String(BER.decodeString(is, new BER.MutableByte()));
    } else if (type == BER.OID) {
        oidValue = new PDUVariableOID();
        oidValue.decodeBER(is);
    } else if (type == BER.TIMETICKS) {
        timeTickValue = BER.decodeUnsignedInteger(is, new BER.MutableByte());
    } else if (type == BER.GAUGE32) {
        gauge32Value = BER.decodeUnsignedInteger(is, new BER.MutableByte());
    } else if (type == BER.COUNTER32) {
        counter32Value = BER.decodeUnsignedInteger(is, new BER.MutableByte());
    } else if (type == BER.NULL) {
        BER.decodeHeader(is, new BER.MutableByte());
    } else if (type == (byte) BER.NOSUCHOBJECT) {
        BER.decodeHeader(is, new BER.MutableByte());
        noSuchObject = true;
    } else if (type == (byte) BER.NOSUCHINSTANCE) {
        BER.decodeHeader(is, new BER.MutableByte());
        noSuchInstance = true;
    } else if (type == (byte) BER.ENDOFMIBVIEW) {
        BER.decodeHeader(is, new BER.MutableByte());
        isEnd = true;
    }
}
```

위 코드는 PDUVariableValue 클래스의 decodeBER 메서드 이다.

1. 디코딩을 하지 않고 type을 얻어낸 뒤 그 type에 따라 디코딩을 한다.

6. 네트워크 관련 코드

```
public class NetworkClient
```

모든 네트워크 통신은 NetworkClient라는 클래스에서 담당한다.

```
public static SNMP sendSNMP(DatagramSocket socket, String host, int port, SNMP packet) throws IOException {
    BEROutputStream os = new BEROutputStream(ByteBuffer.allocate(packet.getBERLength()));
    packet.encodeBER(os);
    byte[] bytes = os.getBuffer().array();

    byte[] receivedBytes = NetworkClient.sendUDP(socket, host, port, bytes);

    BERInputStream is = new BERInputStream(ByteBuffer.wrap(receivedBytes));
    SNMP receivedPacket = new SNMP();
    receivedPacket.decodeBER(is);

    return receivedPacket;
}
```

UDPSocket, host, port, SNMPPacket을 파라미터로 받고 이를 이용해 패킷을 보낸후 받을 패킷을 반환하는 메서드이다.

1. 먼저 받은 패킷을 byte array로 변환
2. 그 다음 이 byte array를 UDPSocket을 통해 host에 전송
3. 받은 byte array를 SNMP 패킷으로 만들어서 반환

```
private static byte[] sendUDP(DatagramSocket socket, String host, int port, byte[] data) throws IOException {
    socket.setSoTimeout(DEFAULT_TIMEOUT);

    InetAddress ia = InetAddress.getByName(host);
    DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);

    byte[] receiveData = new byte[DEFAULT_RECEIVE_BUFFER_SIZE];
    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

    while (true) {
        socket.send(dp);

        try {
            socket.receive(receivePacket);
            return receivePacket.getData();
        } catch (SocketTimeoutException ex) {
            ex.printStackTrace();
        }
    }
}
```

위의 sendSNMP 메서드에서 사용하는 byte array를 UDPSocket으로 보낼 때 쓰는 메서드이다.

1. 먼저 Socket에 Timeout을 설정, Timeout은 1초로 설정
2. InetAddress.getByName을 통해 Domain을 IP주소로 변환
3. 파라미터로 받은 byte array로 DatagramPacket 생성 후 전송
4. Socket을 통해 byte array를 받으면 반환, SocketTimeOutExeption이 발생하면 재전송

7. 메인 플로우

7.1. SNMPGET

```
private void onGetBtnClicked() {
    String oid = oidField.getText().toString();

    new SNMPGetAsyncTask(oid)
            .setCallback(new SNMPPacketCallback() {
                @Override
                public void onSNMPPacketSent(SNMP packet) {
                    resultTextView.setText(packet.toString());
                }

                @Override
                public void onSNMPPacketReceived(SNMP packet) {
                    String resultString = resultTextView.getText().toString();
                    resultString += "\n" + packet.toString();
                    resultTextView.setText(resultString);
                }
            })
            .execute();
}
```

SNMPGET 버튼이 눌리면 onGetBtnClicked라는 메서드가 호출

1. EditText에 입력된 값을 이용하여 ObjectID 값을 얻음
2. SNMPGetAsyncTask를 생성해 ObjectID를 넘기고 실행
3. SNMPPacketCallback을 등록해서 SNMPPacket을 보내거나 받을 때 그 packet을 resultTextView에 출력

```
public SNMPGetAsyncTask(String oid) {
    int requestId = new Random().nextInt( bound: 0x7FFFFFFF);

    packet = SNMPPacketBuilder.create(
            MainActivity.COMMUNITY_READ,
            PDUType.GET_REQUEST,
            requestId,
            oid
    );
}

@Override
protected SNMP doInBackground(Void... voids) {
    try {
        return NetworkClient.sendSNMP(MainActivity.HOST, MainActivity.PORT, packet);
    } catch (IOException ex) {
        return null;
    }
}
```

SNMPGetAsnycTask가 하는 역할도 간단

1. 랜덤으로 RequestID를 만든후 생성자로 받은 ObjectID를 활용해서 SNMP패킷을 만듬
2. 만든 패킷을 전송하기 전에 등록했었던 callback 메서드를 이용하여 처리 (위 사진에 없음)
3. Background Thread에서 그 패킷을 전송
4. 패킷을 받으면 등록했었던 callback 메서드를 이용하여 처리 (위 사진에 없음)

7.2. SNMPWALK

```
private void onWalkBtnClicked() {
    new SNMPWalkAsyncTask()
            .setCallback(new SNMPPacketCallback() {
                @Override
                public void onSNMPPacketSent(SNMP packet) { resultTextView.setText(""); }

                @Override
                public void onSNMPPacketReceived(final SNMP packet) {
                    String tmp = resultTextView.getText().toString();
                    tmp += packet.toSimpleString() + "\n";
                    resultTextView.setText(tmp);
                    scrollView.fullScroll(View.FOCUS_DOWN);
                }
            })
            .execute();
}
```

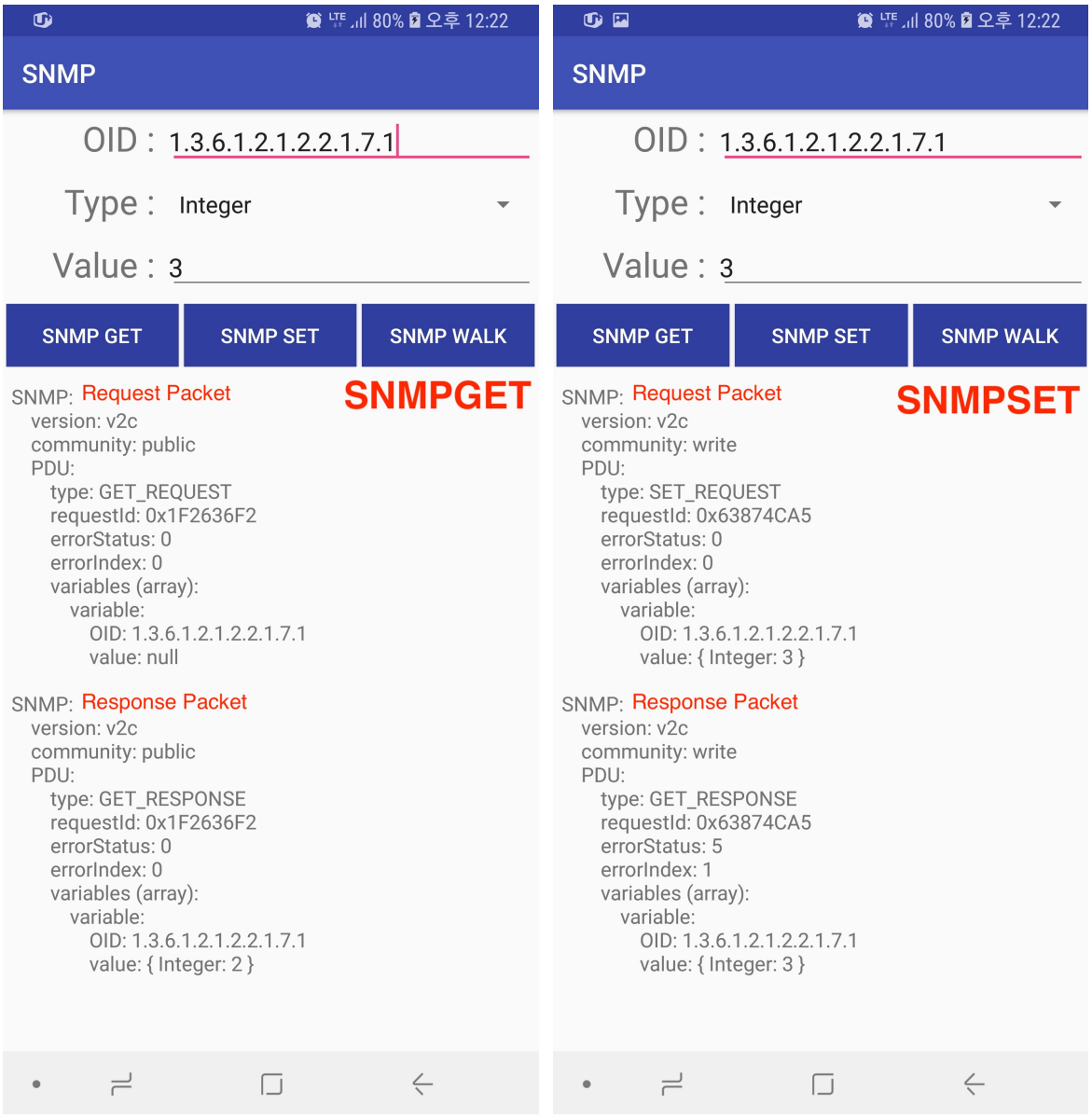
SNMPWALK 버튼이 눌리면 onWalkBtnClicked라는 메서드가 호출

1. SNMPWalkAsyncTask를 생성하고 실행
2. SNMPPacketCallback을 등록해서 SNMPPacket을 보내거나 받을 때 그 packet을 resultTextView에 출력

```
public SNMPWalkAsyncTask() {
    int requestId = new Random().nextInt( bound: 0x7FFFFFFF);

    packet = SNMPPacketBuilder.create(
            MainActivity.COMMUNITY_READ,
            PDUType.GET_NEXT_REQUEST,
            requestId,
             oidStr: "1.3.6.1.2.1"
    );
}

@Override
protected SNMP doInBackground(Void... voids) {
    try {
        DatagramSocket socket = new DatagramSocket();

        while (true) {
            SNMP received = NetworkClient.sendSNMP(socket, MainActivity.HOST, MainActivity.PORT, packet);

            publishProgress(received);

            if (received.pdu.variables.get(0).value.isEnd != null) {
                return null;
            }

            packet.pdu.requestId += 1;
            packet.pdu.variables.get(0).oid = new PDUVariableOID(received.pdu.variables.get(0).oid.toString());
        }
    } catch (IOException ex) {
        return null;
    }
}
```

1. SNMPWalkAsyncTask가 생성되었을 때 GetNextRequest Type인 패킷을 만듬
2. 만든 패킷을 전송하기 전에 등록했었던 callback 메서드를 이용하여 처리 (위 사진에 없음)
3. 1) 만든 패킷을 전송
   2) 받은 패킷을 publishProgress 메서드를 통하여 MainThread에서 등록했었던 callback 메서드를 호출하여 처리 (위 사진에 없음)
   3) 만약 받은 패킷이 SNMP walk의 마지막 패킷이라면 쓰레드 종료
   4) 아니라면 보냈던 패킷의 RequestID와 ObjectID를 수정해서 다시 보낸다.

8. 앱 스크린샷



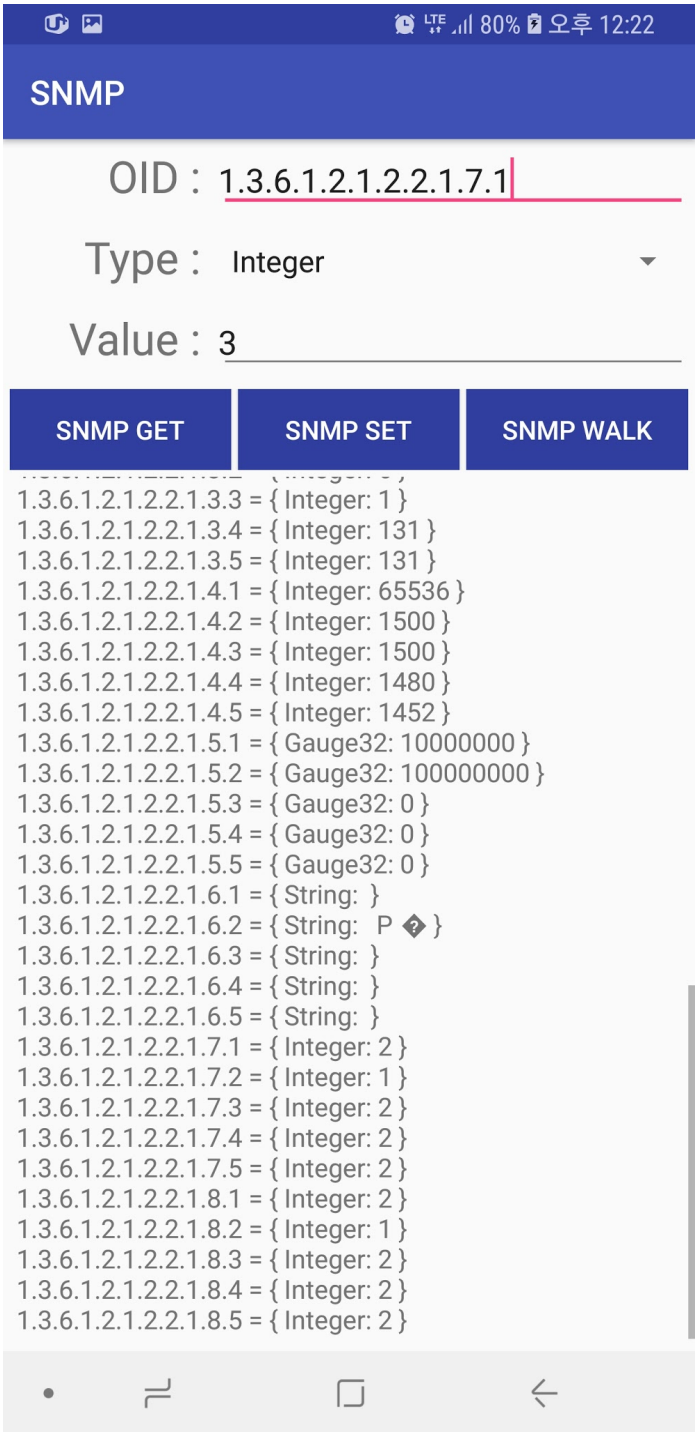
(빨간색 텍스트는 스크린 샷 찍은 후 포토샵으로 추가한 것임)

8.1. SNMPGET

1.3.6.1.2.1.2.2.1.7.1에 해당하는 객체의 값은 Integer이고 2인 것을 알 수 있다.

8.2. SNMPSET

1.3.6.1.2.1.2.2.1.7.1에 해당하는 객체에 적절하지 않은 값(3)을 보내니 받은 SNMP 패킷에 에러가 있는 것을 확인할 수 있다. errorStatus = 5, errorIndex = 1



8.3. SNMPWALK

다양한 값이 들어있는 것을 확인할 수 있다.

9. 겪은 문제점과 해결 방법

BER.java에 INTEGER와 UNSINGED INTEGER를 인코딩하는 메서드는 있었으나 이 둘의 길이를 재는 메서드는 존재 하지 않았다. 각종 데이터들을 인코딩하기 전에는 그 데이터의 길이를 먼저 헤더에 인코딩을 해야 하기 때문에 둘의 길이를 재는 메서드는 반드시 필요하였다.

그래서 BER.java에서 INTEGER와 UNSIGNED INTEGER를 인코딩 하는 부분에서 길이만 재는 부분을 추출해서 따로 메서드를 만듬으로써 해결하였다.

아래 사진은 그 결과이다. (BERLengthUtil 클래스를 만듬)

```java
public class BERLengthUtil {
    // Get BER Length of given integer
    public static int getLengthOfInteger(int integer) {
        int mask;
        int intsize = 4;

        /*
         * Truncate "unnecessary" bytes off of the most significant en
         * 2's complement integer.  There should be no sequence of 9
         * consecutive 1's or 0's at the most significant end of the
         * integer.
         */
        mask = 0x1FF << ((8 * 3) - 1);
        /* mask is 0xFF800000 on a big-endian machine */
        while((((integer & mask) == 0) || ((integer & mask) == mask))
                && intsize > 1){
            intsize--;
            integer <<= 8;
        }
        return intsize;
    }

    // Get BER Length of given unsigned integer
    public static int getLengthOfUnsignedInteger(long value) {
        int LENMASK = 0x0ff;

        // figure out the len
        int len = 1;
        if ((( value >> 24) & LENMASK) != 0) {
            len = 4;
        }
        else if ((( value >> 16) & LENMASK) !=0) {
            len = 3;
        }
        else if ((( value >> 8) & LENMASK) !=0) {
            len = 2;
        }
        // check for 5 byte len where first byte will be
        // a null
        if ((( value >> (8 * (len -1))) & 0x080) !=0)   {
            len++;
        }

        return len;
    }
}
```