

```
//블루투스 지원 유무 확인
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

//블루투스를 지원하지 않으면 null 을 리턴한다
if(mBluetoothAdapter == null){
    Toast.makeText(this, "블루투스를 지원하지 않는 단말기 입니다.",
    Toast.LENGTH_SHORT).show();
    finish();
    return;
}
```

블루투스를 지원하지 않는 기기일 수 있으므로 getDefaultAdapter();으로 블루투스 지원 유무를 판별해 만약 지원하지 않는 기기일 때 finish로 어플을 종료한다.

```
IntentFilter searchFilter = new IntentFilter();
searchFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
//BluetoothAdapter.ACTION_DISCOVERY_STARTED : 블루투스 검색 시작
searchFilter.addAction(BluetoothDevice.ACTION_FOUND); //BluetoothDevice.ACTION_FOUND :
블루투스 디바이스 찾음
searchFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
//BluetoothAdapter.ACTION_DISCOVERY_FINISHED : 블루투스 검색 종료
searchFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
searchFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
registerReceiver(mBluetoothSearchReceiver, searchFilter);
```

블루투스 검색 시작이나 종료 등 이벤트가 일어났을 시 broadcast 를 하여 각각 이벤트에 따른 액션을 취하도록 한다.

```
if(!mBluetoothAdapter.isEnabled()){
    Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(intent, BLUETOOTH_REQUEST_CODE);
}else{
    GetListPairedDevice();
}
```

블루투스가 비활성화 되어있다면 사용자에게 활성화를 요청한다.

```
void connectSelectedDevice(String address){
    mBluetoothDevice = mBluetoothAdapter.getRemoteDevice(address);

    try {
        //선택한 디바이스 페어링 요청
        // Method method = device.getClass().getMethod("createBond", (Class[]) null);
        // method.invoke(device, (Object[]) null);
        mBluetoothSocket =
        mBluetoothDevice.createInsecureRfcommSocketToServiceRecord(BT_UUID);
        mBluetoothSocket.connect();
        mThreadConnectedBluetooth = new ConnectedBluetoothThread(mBluetoothSocket);
        mThreadConnectedBluetooth.start();
        editor.putString("address",address);
        editor.putBoolean("IsAddress",true);
        editor.apply();
        btnShuffle.performClick();
        Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

```

        vibrator.vibrate(1000);
        Toast.makeText(getApplicationContext(),"연결되었습니다",Toast.LENGTH_LONG).show();

    } catch (Exception e) {
        e.printStackTrace();
        editor.putBoolean("IsAddress",false);
    }
}

```

연결하고자 하는 기기의 MAC 주소를 address 라는 매개변수로 받아 연결을 진행한다. 이때 sharedPreferences 를 이용해 연결한 기기의 주소를 “address” 라는 키로 editor 에 저장하여 다음 번 어플을 실행했을 시에도 그 정보를 활용할 수 있도록 한다. 또 “isAddress” 라는 키 또한 저장하여 이 주소가 페어링 된 적이 있다는 정보를 함께 저장한다. 이 정보를 이용해 아래에서 한번 연결이 된 이력이 있는 블루투스 기기들은 자동으로 페어링이 가능하도록 한다.

이어서 연결이 되면 shuffle 메소드를 실행시켜 키패드 배열이 한번 섞이는 이벤트와 send, mute 버튼의 위치를 전송하도록 한다. 마지막으로 vibrator 객체를 사용해 연결됐을 시 토스트 메시지와 진동이 울리도록 하여 사용자에게 페어링이 되었다는 신호를 알려준다.

```

private class ConnectedBluetoothThread extends Thread{
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedBluetoothThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
            Toast.makeText(getApplicationContext(), "소켓 연결 중 오류가 발생했습니다.",
            Toast.LENGTH_LONG).show();
        }
        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }

    public void run() {
        byte[] buffer = new byte[1024];
        int bytes;

        int checkBytes = 0;
        byte[] buffer_b = new byte[1024];

        while (true) {

            if (mBluetoothSocket == null){
                btnSearch.performClick();
                Toast.makeText(getApplicationContext(),"자동 검색을
                시작합니다",Toast.LENGTH_LONG).show();
            }
        }
    }
}

```

```

    }

    try {
        // Read from the InputStream
        bytes = mmInStream.read(buffer);

        if(bytes == 1){ //통신 시 한자리만 오는 것을 예외처리
            System.arraycopy(buffer, 0,buffer_b,0,1);
            checkBytes = 1;
        } else {
            if(checkBytes == 1){
                System.arraycopy(buffer,0,buffer_b,1,bytes);
                checkBytes = 0;
            }
        }

        // Send the obtained bytes to the UI Activity
        //mHandler.obtainMessage(BebopActivity.MESSAGE_READ, bytes, -1,
buffer).sendToTarget();
        GetReceivedData(bytes+1, buffer_b);
    } catch (IOException e) {
        Log.e("MainActivity", "connected(run) - message=" +
e.getLocalizedMessage());
        //connectionLost();
        break;
    }
}

}

public void GetReceivedData(int bytes, byte[] buffer){
    String RealMessage;
    byte[] readBuf = buffer;
    if(bytes>=3) { //앞뒤로 < >가 들어가므로 최소 3 자리
        String readMessage = new String(readBuf, 0, bytes);
        Log.e("MainActivity", "SearchStartEnd(run) - message=" + readMessage);
        RealMessage = readMessage.substring(1, bytes-1); // RealMessage : 앞뒤 < >를
잘라낸 데이터
        Log.e("MainActivity","realMessage = " + RealMessage);
        /*arrMessage = RealMessage.toCharArray(); //arrMessage : 받아온 문자열 ->
char[]로 변환한 데이터*/

        newPW = savedData.getString("newPW","00");
        char[] charPW = new char[newPW.length()];
        for(int i = 0; i<charPW.length;i++) { //newPW 의 길이만큼 실행
            charPW[i] = (newPW.charAt(i)); //newPW 를 char[]로 자르기
        }

        int[] tmp = new int[charPW.length];
        Log.d("main", "charPW = " + charPW[0]);
    }
}

```

```

        for (int i = 0; i < charPW.length; i++) {
            char k = charPW[i];
            tmp[i] = rndArray[k-48]; //charPW(값)이 있는 위치값을 tmp에 저장한다
        }
        String str =
Arrays.toString(tmp).replace("10","A").replace("11","B").replace(",","").replace("[","").replace("]", "");
        Log.d("main", "str = "+str);

        if (RealMessage.equals(str)){
            mThreadConnectedBluetooth.write1((byte)0x41);
        }
        else {
            mThreadConnectedBluetooth.write1((byte)0x42);
            Shuffle();
        }
    }
}

public void write1(byte str) {
    byte bytes = str;
    try {
        mmOutputStream.write(bytes);
    } catch (IOException e) {
        Toast.makeText(getApplicationContext(), "데이터 전송 중 오류가 발생했습니다.",
Toast.LENGTH_LONG).show();
    }
}

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Toast.makeText(getApplicationContext(), "소켓 해제 중 오류가 발생했습니다.",
Toast.LENGTH_LONG).show();
    }
}
}
}

```

Thread 하나를 생성하여 socket 통신과 관련된 이벤트들은 이 스레드를 통해 이루어 지도록 한다.

run 함수를 통해 스레드가 실행될 동안 while 문을 통해 socket 이 생성되지 않았을 때, 즉 연결된 기기가 없을 때 검색버튼을 누르는 함수를 발생시켜 자동 검색을 통한 페어링을 하도록 한다. 그리고 지속적으로 inputStream에 들어오는 데이터를 받아들여 buffer에 저장하고 bytes에 그 길이를 저장한다. 실제로 통신을 하면서 한번에 데이터가 들어오는 것이 아닌 한 글자, 나머지 글자 형식으로 두 번에 걸쳐 데이터가 들어오는 버그를 발견해 만약 들어온 글자수가 한 글자일 때 buffer를 buffer_b 배열에 복사한 후 나머지 데이터를 그 뒤에 이어서 받는 알고리즘을 통해 받은 데이터 전체를 buffer_b에 저장한다.

그런 후 GetReceivedData로 그 buufer_b(데이터 내용)와 bytes + 1(글자수)을 인자로 전송해 실행시킨다.

GetReceivedData 메소드에서는 받아들이는 데이터를 가공하는 역할을 수행한다.

데이터를 전송받는 과정에서 노이즈가 낄 수 있기 때문에 데이터의 앞뒤로 < >를 추가해 받는다. 따라서 글자수는 3 미만일 수 없으므로 예외처리를 해주고 3 자리 이상의 데이터가 들어왔을 경우 char 형 배열로 받아들이는 데이터를 readMessage에 string 형식으로 저장한다. 그 데이터의 맨 앞과 마지막 글자를 제거하고 RealMessage에 저장한다.

Sharedpreferences에 저장되어 있는 비밀번호를 newPW에 저장하여 char 배열형식으로 자른 후 int 형 배열로 변환시킨다. 그리고 첫 자리부터 마지막 자리까지 k에 담은 후 rndArray(무작위로 배열된 키패드 배열)에서 k-48(아스키코드 표를 참조하여 데이터 형식을 맞춰줌)에 해당하는 위치 값을 tmp에 저장한다. 이로써 tmp 배열에는 저장된 비밀번호가 현재 섞여있는 무작위 숫자 배열에서 몇 번째에 위치해 있는지를 저장하고 있다.

이렇게 해서 만들어진 RealMessage(전송받은 위치 값)과 str(비밀번호가 위치한 index)을 비교해 같다면 기기로 0x41이라는 데이터를 보내주고 다르다면 0x42라는 데이터를 보내고 shuffle을 실행하여 비밀번호가 틀렸을 때 키패드 배열이 섞이도록 해준다.

Write 메소드는 outputStream을 통해 전송하고자 하는 데이터(byte 형)을 보내도록 해준다.

cancel 메소드는 연결된 소켓을 해제하는 이벤트를 수행한다.

```
//블루투스 검색결과 BroadcastReceiver
BroadcastReceiver mBluetoothSearchReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        String savedAddress = savedInstanceState.getString("address", null);

        switch(action){

            //블루투스 디바이스 검색 종료
            case BluetoothAdapter.ACTION_DISCOVERY_STARTED:
                dataDevice.clear();
                bluetoothDevices.clear();
                Toast.makeText(MainActivity.this, "블루투스 검색 시작",
                    Toast.LENGTH_SHORT).show();
                alertDialog.show();
                break;

            //블루투스 디바이스 검색 종료
            case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:
                btnSearch.setEnabled(true);
                pairedDevices = mBluetoothAdapter.getBondedDevices();
                if (savedData.getBoolean("IsAddress", false) && (pairedDevices.size() > 0)) {
                    for (BluetoothDevice tempDevice : bluetoothDevices) {
                        String address = savedInstanceState.getString("address", "nope");
                        if (tempDevice.getAddress().equals(address)) {
                            connectSelectedDevice(address);
                            break;
                        }
                    }
                }
            }
        }
    }
}
```

```
    }  
    break;  
  
    //블루투스 디바이스 찾음  
    case BluetoothDevice.ACTION_FOUND:  
        //검색한 블루투스 디바이스의 객체를 구한다  
        BluetoothDevice device =  
intent .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);  
  
        //데이터 저장  
        Map map = new HashMap();  
        map.put("name", device.getName()); //device.getName() : 블루투스  
디바이스의 이름  
        map.put("address", device.getAddress()); //device.getAddress() : 블루투스  
디바이스의 MAC 주소  
        dataDevice.add(map);  
        //리스트 목록갱신  
        adapterDevice.notifyDataSetChanged();  
        //블루투스 디바이스 저장  
        bluetoothDevices.add(device);  
        pairedDevices = mBluetoothAdapter.getBondedDevices();  
        if (device.getAddress().equals(savedData.getString("address","nope"))&&  
pairedDevices.size()>0){  
            mBluetoothAdapter.cancelDiscovery();  
        }  
        break;  
  
        //블루투스 디바이스 페어링 상태 변화  
        case BluetoothDevice.ACTION_BOND_STATE_CHANGED:  
            BluetoothDevice paired =  
intent .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);  
            if(paired.getBondState()==BluetoothDevice.BOND_BONDED){  
                //데이터 저장  
                Map map2 = new HashMap();  
                map2.put("name", paired.getName()); //device.getName() : 블루투스  
디바이스의 이름  
                map2.put("address", paired.getAddress()); //device.getAddress() :  
블루투스 디바이스의 MAC 주소  
                dataPaired.add(map2);  
                //리스트 목록갱신  
                adapterPaired.notifyDataSetChanged();  
  
                //검색된 목록  
                if(selectDevice != -1){  
                    bluetoothDevices.remove(selectDevice);  
                    dataDevice.remove(selectDevice);  
                    adapterDevice.notifyDataSetChanged();  
                    selectDevice = -1;  
                }  
            }  
        }  
        break;
```

```

        case BluetoothDevice.ACTION_ACL_CONNECTED:

            break;

        }

    }

};

```

broadcast 에서는 위에서 블루투스 이벤트에 따른 broadcast 를 받아들여 이벤트를 처리한다.

블루투스 검색을 시작하면 먼저 저장되어 있던 정보들을 초기화 시킨 후 Dialog 를 출력한다.

검색이 종료되면 검색된 기기와 저장된 주소가 같을 경우 자동으로 페어링이 되도록 한다.

디바이스를 찾았을 때 dialog 창에 검색된 기기들의 정보를 출력하고 저장되어 있는 주소와 검색된 기기의 주소가 같을 때 검색을 종료하여 리소스 낭비가 없도록 한다.

블루투스 상태가 변하면 저장된 기기들의 정보를 가져온다.

```

public void Shuffle()
{
    boolean flag[] = new boolean[12];

    for(int i = 1; i < 13; i++)
    {
        while(true) {
            int rnd = (int)(Math.random() * 12 ); //rnd 는 0~11 까지 랜덤 수
            if (!flag[rnd]) {
                if(mBluetoothSocket!=null) {
                    if (rnd == 11) {
                        mThreadConnectedBluetooth.write1((byte) ((byte) (0x60) + (byte) (i
- 1))); //Send 자리 전송
                        SystemClock.sleep(30);
                    } else if (rnd == 10) {
                        mThreadConnectedBluetooth.write1((byte) ((byte) (0x70) + (byte) (i
- 1))); //Mute 자리 전송
                        SystemClock.sleep(30);
                    }
                }
            }

            int posId = getResources().getIdentifier("pos" + i, "id",
getPackageName()); // @+id/pos1, @+id/pos2...를 posId 에 넣는다.
            int drawableId = getResources().getIdentifier("line_" + rnd, "drawable",
getPackageName()); // drawable/line_0, drawable/line_2 를 drawableId 에 넣는다

            iv_pos[i - 1] = (ImageView) findViewById(posId); //id 할당
            iv_pos[i - 1].setImageResource(drawableId); // 할당된 iv_pos[]에 그림
그리기

            flag[rnd] = !flag[rnd];

            rndArray[rnd] = i-1;

            break;
        }
    }
}

```

```
}
}
}
```

shuffle 메소드는 키패드를 무작위로 shuffle 하는 역할을 수행한다.

Rnd 에 12 까지의 무작위 수를 넣은 후 첫번째부터 12 번째 자리에 순차적으로 랜덤한 키패드를 배치한다. 이때 배열된 숫자가 10(MUTE)일 경우와 11(Send)일 때 그 위치를 기기에 전송해준다.

```
public void ChangePW(View view)
{
    final EditText et = new EditText(this);
    final AlertDialog.Builder builder = new AlertDialog.Builder(this,R.style.ChangeMyPW);
    builder.setTitle("비밀번호 바꾸기").setMessage("변경할 비밀번호를 입력하세요");;

    builder.setView(et);

    builder.setPositiveButton("저장", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            newPW = et.getText().toString();
            editor.putString("newPW",newPW);
            editor.apply();
            Toast.makeText(getApplicationContext(),"비밀번호가 " + newPW + "로
변경되었습니다",Toast.LENGTH_LONG).show();
        }
    });
    builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
}
```

changepw 에서는 dialog 를 출력하여 사용자가 그곳에 string 형식으로 새로운 비밀번호를 입력하면 그 데이터를 sharepreferences 에 저장하여 비밀번호를 잊지 않고 다음번에도 그 번호를 사용할 수 있도록 한다.

61

ICT