컴퓨터 네트워크
-5주차-

학번: 201404377
이름: 진승언

과제목표(도출해야할결과)

지금까지의 과제는 도커 컨테이너와 파이썬 기반 서버를 두고 네트워크 통신을 했다면 이번에는 안드로이드 폰을 이용해서 자바 기반으로 소켓 통신을 하는 것이 과제목표였다. 따라서 폰 하나는 서버 다른 폰 하나는 클라이언트를 맡고 소켓 통신을 하는 결과를 구현했고 프로젝트는 서버와 클라이언트 두 개로 나눠서 만들었다.

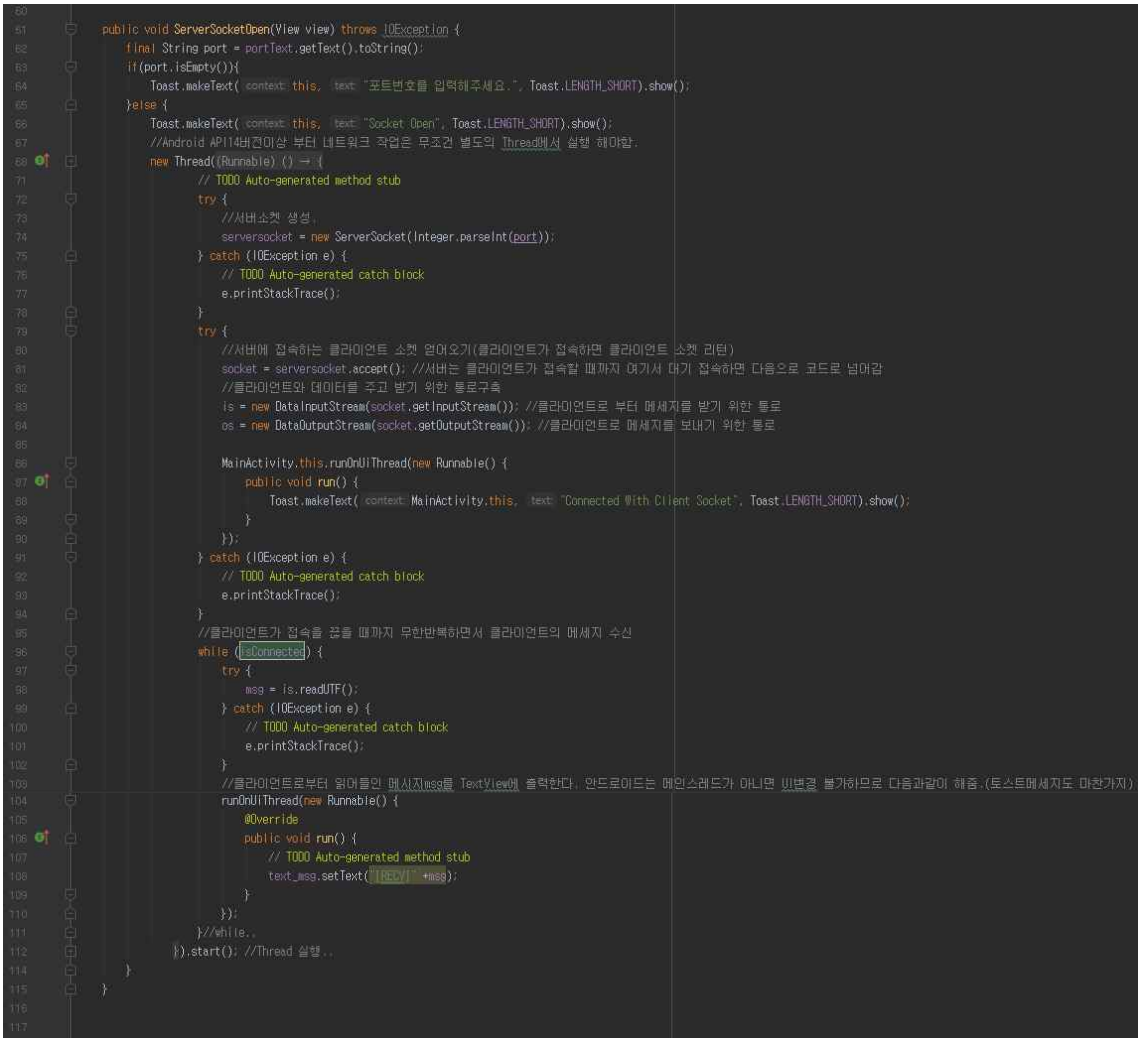
코드설명과과제해결방법

[서버]

```
    //LH ip 얻기
    private String getLocalIpAddress() throws UnknownHostException {
        WifiManager wifiManager = (WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE);
        assert wifiManager != null;
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        int ipInt = wifiInfo.getIpAddress();
        return InetAddress.getByAddress(ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(ipInt).array()).getHostAddress();
    }
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ipText = findViewById(R.id.ip);
    portText = findViewById(R.id.port);
    text_msg = findViewById(R.id.chatting);
    edit_msg = findViewById(R.id.msg);

    //내 아이피 확인 및 세팅
    try {
        ipText.setText(getLocalIpAddress());
    } catch (UnknownHostException e) {
        e.printStackTrace();
    }
}
```

위의 getLoacalAddress() 메소드를 통해서 현재 내 휴대폰의 ip를 알 수 있고 TextView에 set해준다. 클라이언트에서 서버 ip를 알아야 접속이 가능하므로 필요하다.

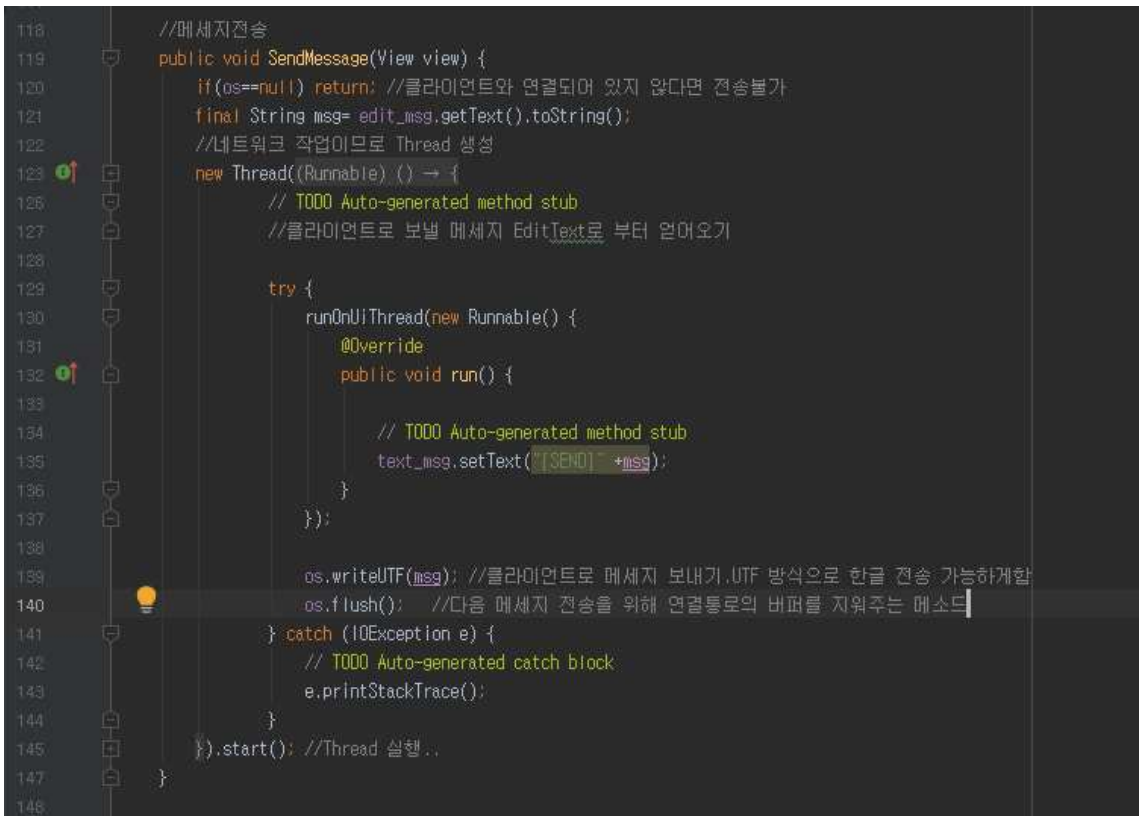


```
60
61      public void ServerSocketOpen(View view) throws IOException {
62          final String port = portText.getText().toString();
63          if(port.isEmpty()){
64              Toast.makeText( context: this,  text: "포트번호를 입력해주세요.", Toast.LENGTH_SHORT).show();
65          }else {
66              Toast.makeText( context: this,  text: "Socket Open", Toast.LENGTH_SHORT).show();
67              //Android API14버전이상 부터 네트워크 작업은 무조건 별도의 Thread에서 실행 해야함.
68              new Thread((Runnable) () → {
71                      // TODO Auto-generated method stub
72                      try {
73                          //서버소켓 생성.
74                          serversocket = new ServerSocket(Integer.parseInt(port));
75                      } catch (IOException e) {
76                          // TODO Auto-generated catch block
77                          e.printStackTrace();
78                      }
79                      try {
80                          //서버에 접속하는 클라이언트 소켓 얻어오기(클라이언트가 접속하면 클라이언트 소켓 리턴)
81                          socket = serversocket.accept(); //서버는 클라이언트가 접속할 때까지 여기서 대기 접속하면 다음으로 코드로 넘어감
82                          //클라이언트와 데이터를 주고 받기 위한 통로구축
83                          is = new DataInputStream(socket.getInputStream()); //클라이언트로 부터 메세지를 받기 위한 통로
84                          os = new DataOutputStream(socket.getOutputStream()); //클라이언트로 메세지를 보내기 위한 통로
85
86                          MainActivity.this.runOnUiThread(new Runnable() {
87                              public void run() {
88                                  Toast.makeText( context: MainActivity.this,  text: "Connected With Client Socket", Toast.LENGTH_SHORT).show();
89                              }
90                          });
91                      } catch (IOException e) {
92                          // TODO Auto-generated catch block
93                          e.printStackTrace();
94                      }
95                      //클라이언트가 접속을 끊을 때까지 무한반복하면서 클라이언트의 메세지 수신
96                      while (isConnected) {
97                          try {
98                              msg = is.readUTF();
99                          } catch (IOException e) {
100                             // TODO Auto-generated catch block
101                             e.printStackTrace();
102                         }
103                         //클라이언트로부터 읽어들인 메시지msg를 TextView에 출력한다. 안드로이드는 메인스레드가 아니면 UI변경 불가하므로 다음과같이 해줌.(토스트메세지도 마찬가지)
104                         runOnUiThread(new Runnable() {
105                             @Override
106                             public void run() {
107                                 // TODO Auto-generated method stub
108                                 text_msg.setText("RECV" +msg);
109                             }
110                         });
111                     }//while..
112                 }).start(); //Thread 실행..
114         }
115     }
116
117
```

ServerSocketOpen()은 Open Server 버튼을 누르면 동작할 온클릭 메소드이다.
[62~78라인]: 소켓을 생성할려면 포트번호가 필요하다. 그러므로 포트번호를 editText에서 입력했는지 검사 후 서버 소켓을 생성한다. 안드로이드에서는 소켓 통신을(네트워크 통신) 할 때 스레드가 필수적으로 사용된다.

[81~94라인]: accept()로 서버에 접속하는 클라이언트의 소켓을 얻어온다. (클라이언트가 접속할때까지 대기한다.) 그 후 클라이언트가 접속하면 데이터를 주고받기 위한 통로를 DataInputStream과 DataOutputStream으로 구성한다. 또한 클라이언트와 연결되었다고 토스트메세지를 띄워주는데 안드로이드에서는 메인스레드 외에 스레드에서는 UI를 변경하거나 띄워줄수 없기 때문에 runOnUiThread()를 사용해줘야 한다.

[96~112라인]: 클라이언트가 접속을 끊을 때까지 while문으로 돌리면서 클라이언트의 메시지를 계속 수신할 수 있게 한다. 메시지는 앞서 선언한 DataInputStream (통로)의 readUTF()를 통해 받을 수 있다. UTF는 한글을 수신할 수 있기 위해 필요하다. 그리고 수신받은 메시지를 TextView에 띄워주도록한다. 이것도 마찬가지로 메인스레드가 아니므로 runOnUiThread()를 사용하여 UI를 변경하도록 한다.

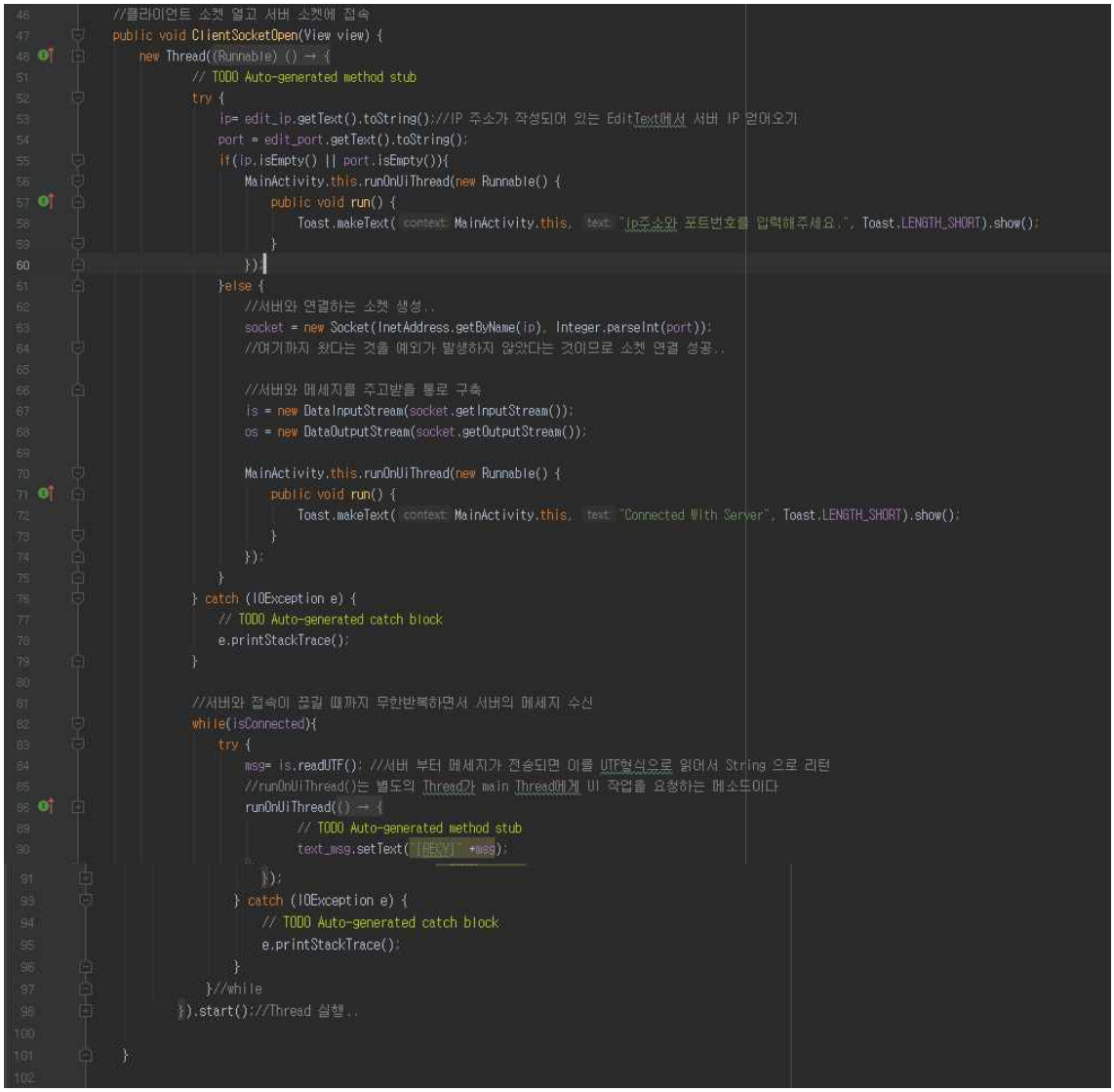


```
118     //메세지전송
119     public void SendMessage(View view) {
120         if(os==null) return; //클라이언트와 연결되어 있지 않다면 전송불가
121         final String msg= edit_msg.getText().toString();
122         //네트워크 작업이므로 Thread 생성
123         new Thread((Runnable) () → {
126                 // TODO Auto-generated method stub
127                 //클라이언트로 보낼 메세지 EditText로 부터 얻어오기
128
129                 try {
130                     runOnUiThread(new Runnable() {
131                         @Override
132                         public void run() {
133
134                             // TODO Auto-generated method stub
135                             text_msg.setText("[SEND] " +msg);
136                         }
137                     });
138
139                     os.writeUTF(msg); //클라이언트로 메세지 보내기,UTF 방식으로 한글 전송 가능하게함
140                     os.flush();   //다음 메세지 전송을 위해 연결통로의 버퍼를 지워주는 메소드
141                 } catch (IOException e) {
142                     // TODO Auto-generated catch block
143                     e.printStackTrace();
144                 }
145         }).start(); //Thread 실행..
147     }
148
```

메시지 전송버튼을 눌렀을 때 동작하는 온클릭 메소드이다.
[120라인]: DataOutputStream (os) 가 null이면 앞선 코드에서 살펴볼 수 있듯이 클라이언트가 접속되지 않은 것을 의미하므로 메시지를 보낼 수 없다.

[123~145라인]: 전송할 메시지를 갖고온 후 소켓 통신 즉 네트워크 통신을 해야 하므로 스레드가 사용된다. DataOutputStream 객체의 writeUTF를 통해 메시지를 보내준다. 그리고 다음 메시지 전송을 위해 DataOutputStream 연결통로의 버퍼를 flush()로 지워준다. 송신한 메시지를 자신의 폰에서 볼 수 있게 setText() 해준다. 이것도 UI를 건드는 작업이므로 runOnUiThread()를 이용한다.

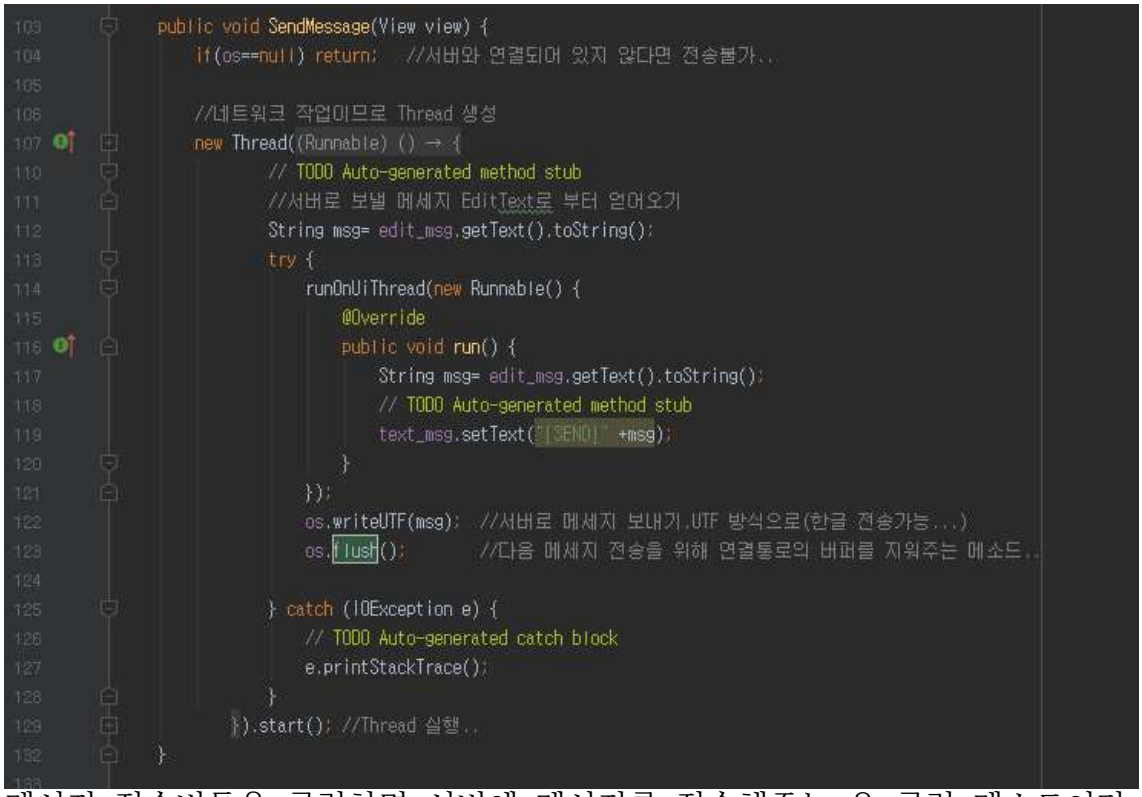
[클라이언트]



```
//클라이언트 소켓 열고 서버 소켓에 접속
public void ClientSocketOpen(View view) {
    new Thread((Runnable) () → {
            // TODO Auto-generated method stub
            try {
                ip= edit_ip.getText().toString();//IP 주소가 작성되어 있는 EditText에서 서버 IP 얻어오기
                port = edit_port.getText().toString();
                if(ip.isEmpty() || port.isEmpty()){
                    MainActivity.this.runOnUiThread(new Runnable() {
                        public void run() {
                            Toast.makeText( context: MainActivity.this,  text: "ip주소와 포트번호를 입력해주세요.", Toast.LENGTH_SHORT).show();
                        }
                    });
                }else {
                    //서버와 연결하는 소켓 생성..
                    socket = new Socket(InetAddress.getByName(ip), Integer.parseInt(port));
                    //여기까지 왔다는 것을 예외가 발생하지 않았다는 것이므로 소켓 연결 성공..

                    //서버와 메세지를 주고받을 통로 구축
                    is = new DataInputStream(socket.getInputStream());
                    os = new DataOutputStream(socket.getOutputStream());

                    MainActivity.this.runOnUiThread(new Runnable() {
                        public void run() {
                            Toast.makeText( context: MainActivity.this,  text: "Connected With Server", Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            //서버와 접속이 끊길 때까지 무한반복하면서 서버의 메세지 수신
            while(isConnected){
                try {
                    msg= is.readUTF(); //서버 부터 메세지가 전송되면 이를 UTF형식으로 읽어서 String 으로 리턴
                    //runOnUiThread()는 별도의 Thread가 main Thread에게 UI 작업을 요청하는 메소드이다
                    runOnUiThread(() → {
                            // TODO Auto-generated method stub
                            text_msg.setText("RECV" +msg);
                        });
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }//while
            }).start();//Thread 실행..

    }
```

접속 버튼을 누르면 클라이언트의 소켓을 생성하고 서버 소켓에 연결하는 온클릭 메소드이다.
[48~79라인]: ip와 port를 EditText를 통해 받는다. 이때 ip는 서버의 ip와 열어논 포트번호를 입력해야 한다. 그리고 서버에서 소켓을 생성했던 것처럼 소켓을 생성한다. (다른점은 ip를 입력해야 한다.) 통로도 서버에서 했던 것처럼 생성해준다. 서버와 연결됐다면 토스트메세지를 띄워주도록 한ㄷ.

[82~100라인]: 서버와 접속이 끊킬 때 까지 무한 반복하면서 서버의 메시지를 수신하고 UI에 setText() 해준다.. 이것은 서버에서의 코드와 똑같다.



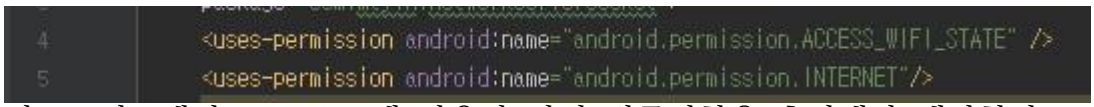
```
public void SendMessage(View view) {
    if(os==null) return;   //서버와 연결되어 있지 않다면 전송불가..

    //네트워크 작업이므로 Thread 생성
    new Thread((Runnable) () → {
            // TODO Auto-generated method stub
            //서버로 보낼 메세지 EditText로 부터 얻어오기
            String msg= edit_msg.getText().toString();
            try {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        String msg= edit_msg.getText().toString();
                        // TODO Auto-generated method stub
                        text_msg.setText("[SEND]" +msg);
                    }
                });
                os.writeUTF(msg);  //서버로 메세지 보내기,UTF 방식으로(한글 전송가능...)
                os.flush();        //다음 메세지 전송을 위해 연결통로의 버퍼를 지워주는 메소드..

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }).start(); //Thread 실행..
}
```

메시지 전송버튼을 클릭하면 서버에 메시지를 전송해주는 온 클릭 메소드이다. 서버에서의 메시지 전송과 코드가 동일하다. 스레드를 이용해서 네트워크 통신을 하고 stream 통로를 통해 메시지를 전송하고 flush()로 버퍼를 비워준다.

```
@Override
protected void onStop() {
    super.onStop();
    try {
        socket.close(); //소켓을 닫는다.
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

안드로이드 생명주기인 onStop() 시 소켓연결을 close()하도록 해주었다.

[P.S]



```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET"/>
```

안드로이드에서 manifest에 다음과 같이 접근권한을 추가해야 해야한다.

실행결과 youtube 링크

https://www.youtube.com/watch?v=OvRQpf1oZ5M&feature=youtu.be

과제후기

다양한 환경으로 소켓통신을 구현해 볼 수 있어서 좋았습니다.