<모바일응용 소프트웨어 설계>

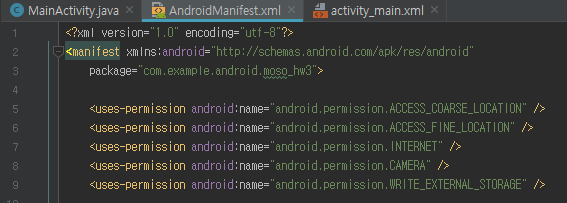
# 3주차 실습 과제

정보통신공학과

12161774 이 채 은

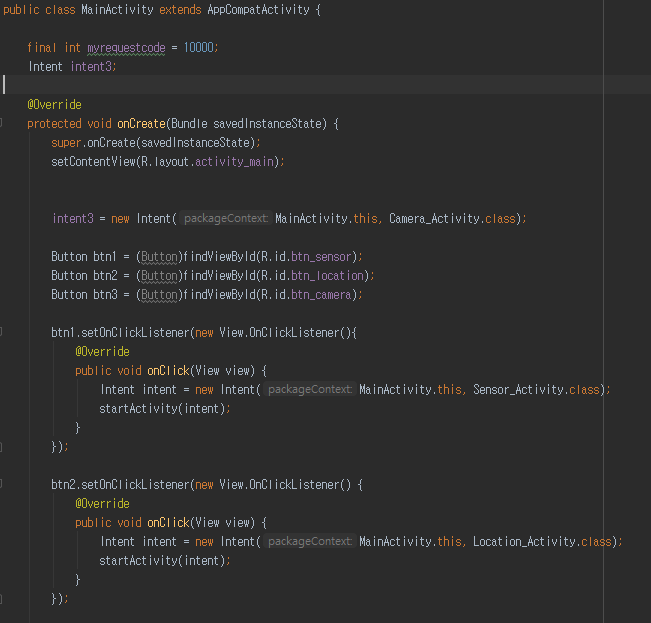
**[소스 코드]**

1. AndroidManifest.xml

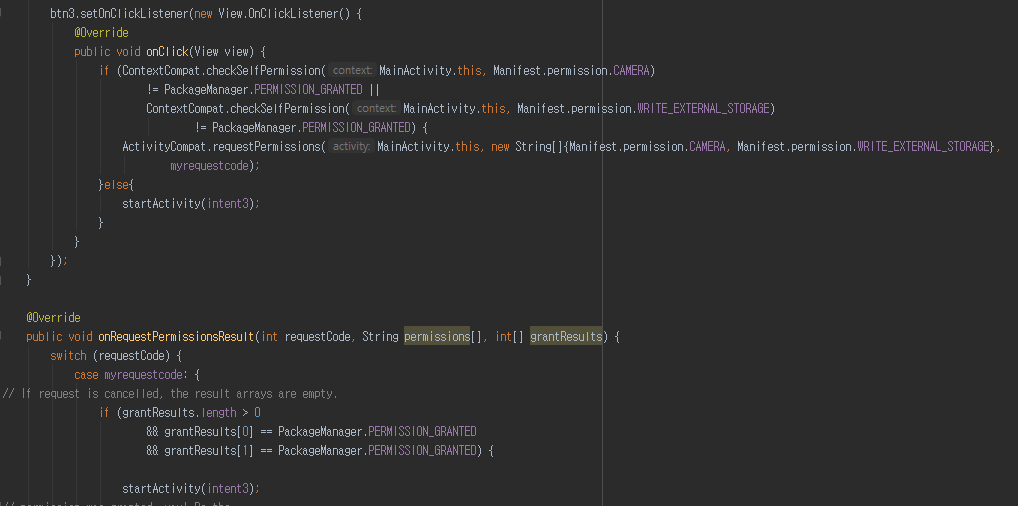


=> camera, location을 앱에서 이용하기 위해서 manifest 에 permission을 추가해준다.

2. MainActivity.java

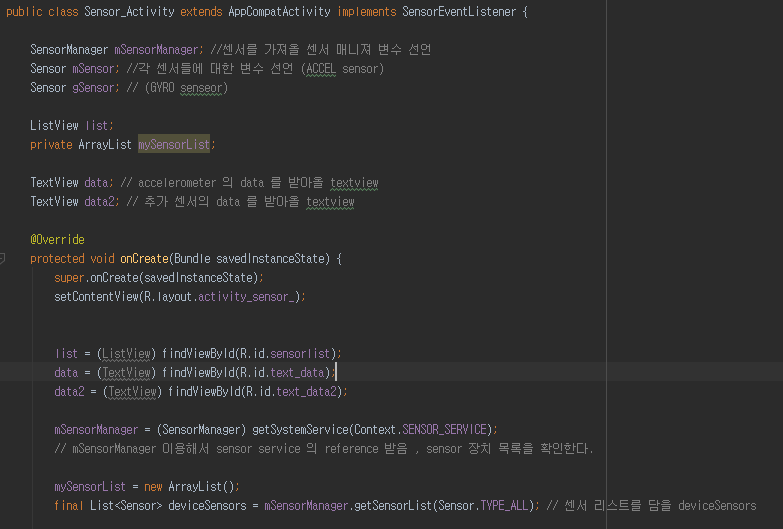


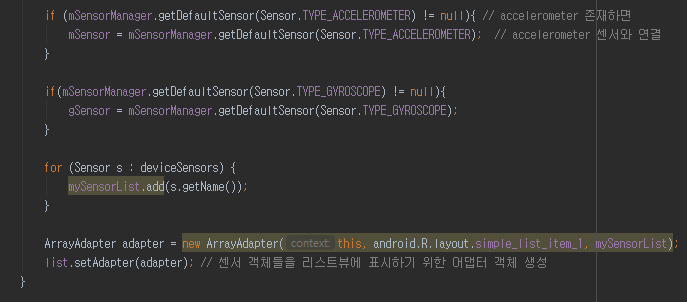
=> 버튼 세 개를 추가해서 각각 센서, 로케이션, 카메라와 관련된 Activity를 실행하도록 했다.

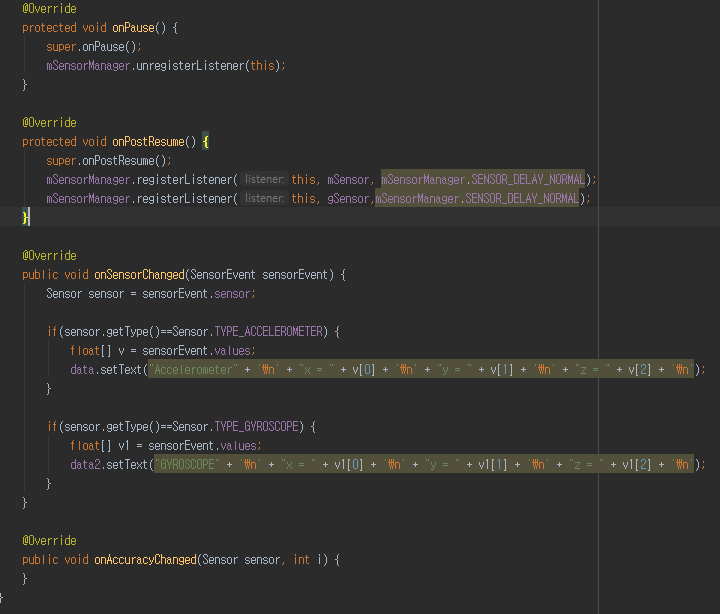


=> 카메라 버튼을 누르면 runtime 에 permission을 물어보고 실행하도록 한다. IF 문을 이용해서 WRITE\_EXTERNAL\_STORAGE, CAMERA 관련 퍼미션을 물어보고 모두 수락한 경우에 카메라 관련 activity가 실행되도록 한다. 그리고 onRequestPermissionResult를 이용해서 requestCode를 전달 받으면 requestCode와 grantResult 가 모두 맞는 경우 activity가 실행되도록 한다.

3. Sensor\_Activity.java

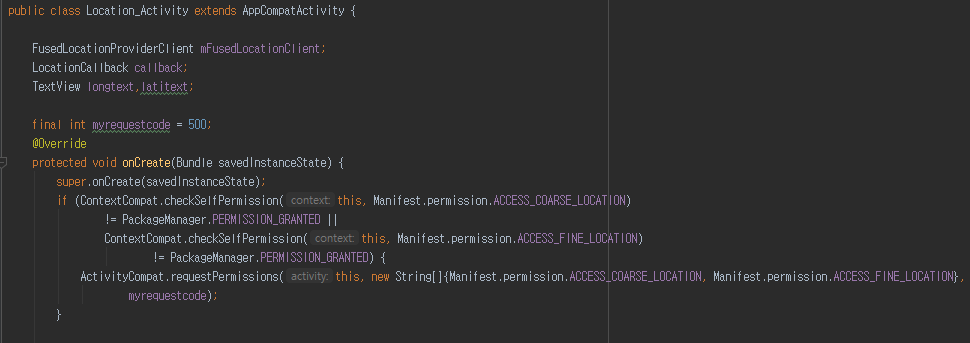






=> sensor 인스턴스를 선언하고, sensormanager 변수를 선언해서 각 sensor를 가져올 수 있도록 한다. sensor목록을 받아올 listview에 adapter, sensormanager를 이용해서 getname으로 센서 이름의 목록을 받아온다. sensorEvent로 센서의 값 중 x, y, z value를 받아온다. Value array의 0,1,2번째 자리에서 순서대로 가지고오는것이다. 이 값은 setText로 data 라는 ID를 가진 textview에 보여진다. ACCELEROMETER 와 GYROSCOPE 센서 두개의 값을 가지고 온다.

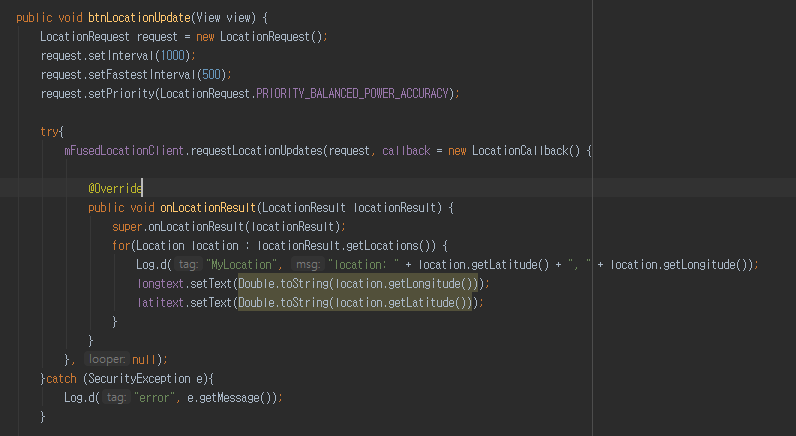
4. Location\_Activity.java



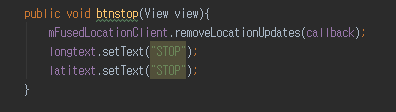
=> onCreate에서 퍼미션을 추가해서 런타임에 권한을 물어본다. ACCESS\_COARSE\_LOCATION, ACCESS\_FINE\_LOCATION 두 개의 퍼미션을 받아온다. 또한 FusedLocationProviderClient 인스턴스를 생성한다.



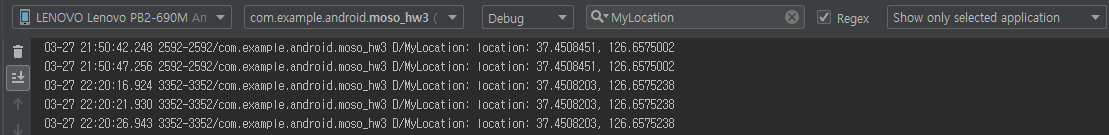
=> last 버튼을 눌렀을 때, last location을 받아오게 하는 함수이다. FusedLocationProviderClient 인스턴스를 이용해서 getLastLocation 메서드로 last location을 안다. 이렇게 받아온 값은 toString을 이용해서 string 값으로 바꿔서 setText로 longitude, latitude 값을 각 textview에 추가해준다.



=> request를 눌렀을 때 일정 주기에 따라서 location 값을 업데이트 해주는 btnLocationUpdate 함수를 구현했다. setInterval, setFastest Interval, setPriority를 이용해서 주기를 밀리세크 단위로 설정하고, 최고 업데이트 주기와 우선 순위를 정해준다. 그리고 location의 결과를 getLongitude, getLatitude로 받아와 마찬가지로 textview에 나타낸다.

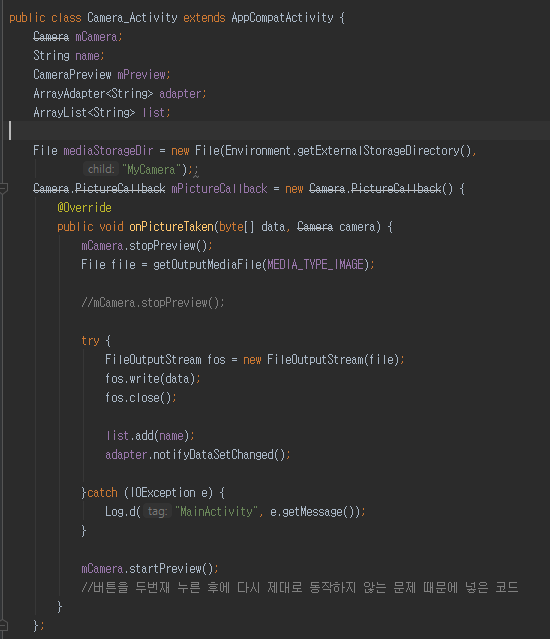


=> stop 버튼을 눌렀을 때 removeLocationUpdates를 이용해서 위치 업데이트를 멈춰준다. 텍스트에는 stop이 출력되고, logcat을 통해 업데이트 되고 있던 위치 값이 멈추는 것을 확인할 수 있었다.



=> log.d 로 출력한 결과 request 를 누르면 위치가 뜨고, stop 을 누르면 주기적으로 업데이트 되고있던 위치가 멈추는 것을 확인할 수 있다.

5. Camera\_Activity.java

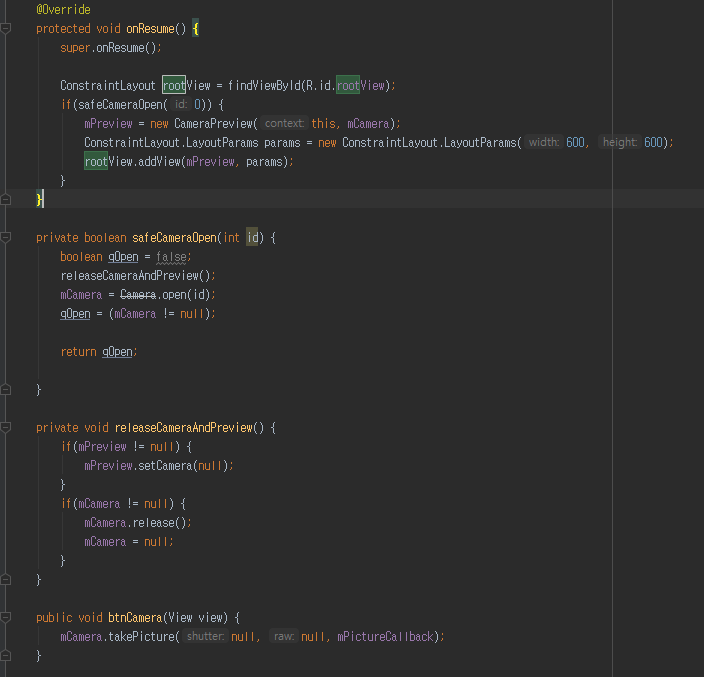


=> File 형태로 mediaStorageDir 를 선언하고, 매니페스트에서 제공되는 environment의 getExternalStorageDirectory를 이용해서 디렉토리 파일을 핸들할 수 있도록 한다. 외장 메모리의 path 상태를 알 수 있다. 그 아래에 MyCamera 라는 파일을 추가해주는 역할을 한다. try 문에서는 FileOutputstream 을 fos로 이름 짓고, fos.write 와 fos.close로 사진을 찍으면 바로 data에 write하고, close 할 수 있도록 했다. 이 때 함수의 처음에 stopPreview를 먼저 해주고, 함수의 마지막에 다시 startPreview를 해주느데, 버튼을 두번 누른 후 다시 제대로 동작하지 않는 문제를 해결하기 위해 코드를 추가했다.



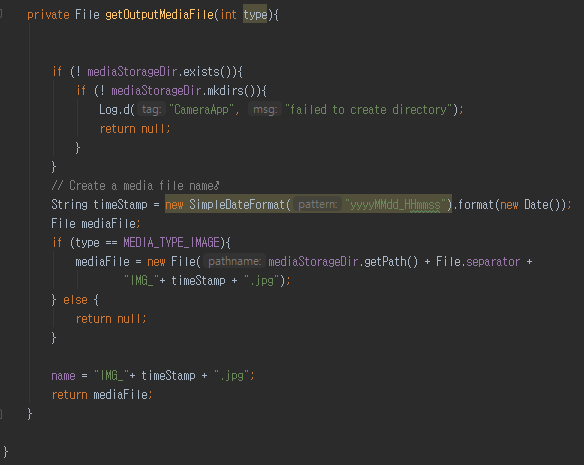


=> OnCreate 에서는 카메라로 찍은 사진 파일의 이름을 리스트뷰로 나타내기 위해서 ArrayList 객체를 생성한다. 그리고 File 형태로 myfile을 선언해서 mediaStorageDir의 path와 seperator를 이용해서 슬래쉬를 추가한다. 그리고 myfile을 리스트로 추가한다. 이 리스트는 for문을 통해서 ArrayList에 add 된다. ListView에 추가된 각 파일을 누르면 파일의 형태가 imageview 자리에 나타나야 하는데, 이는 onclicklistener를 등록해서 onclick 시에 실행되도록 했다. 클릭한 파일명을 string 형태로 가져와서 bitmap 으로 다시 가져오고, setimageBitmap 을 이용해서 imageview 자리에 나타날 수 있도록 한다.



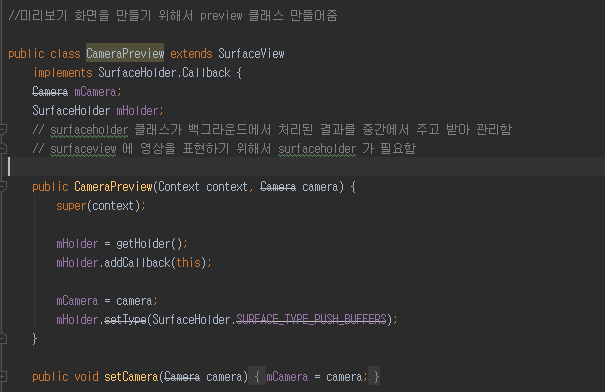
=> constraintlayout 형태로 rootView를 가져온다. if문을 통해서 safeCameraOpen 의 id 가 0으로 제대로 return 됐을 때 width와 height 가 600인 camerapreview 위치를 만들어준다.

safeCameraOpen 함수는 카메라 클래스의 open 함수를 이용해서 카메라를 제어할 수 있도록 open 한다.



=> 저장되는 파일의 이름을 설정해주는 함수이다. Getpath를 이용해서 path를 받아오고, separator로 슬래쉬를 추가하고, IMG 와 timestamp(string 으로 선언하고 simpledateformat을 이용해서 날짜를 구현함), jpg를 순서대로 추가해준다.

6. CameraPreview.java

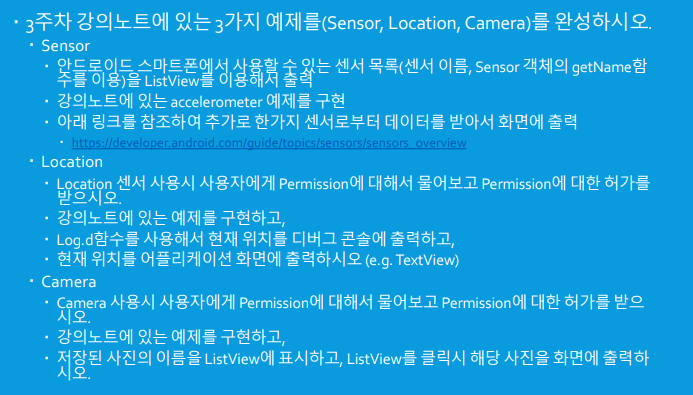
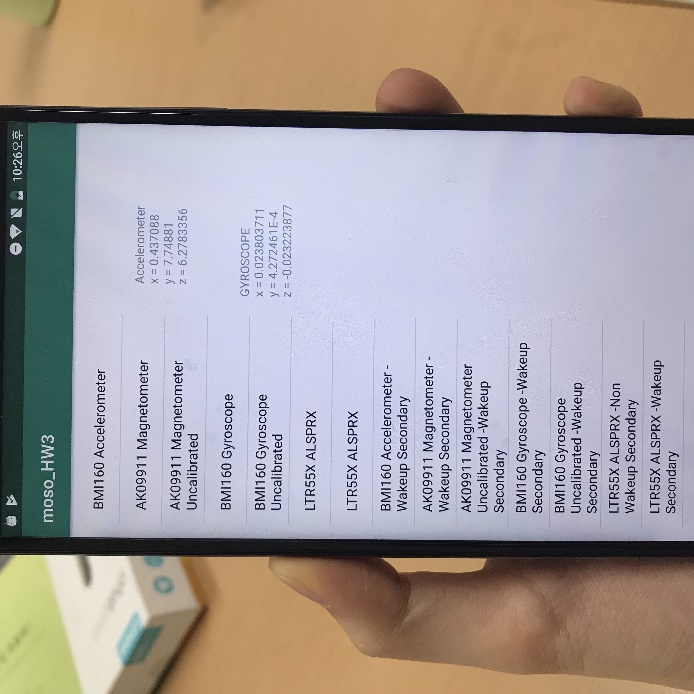
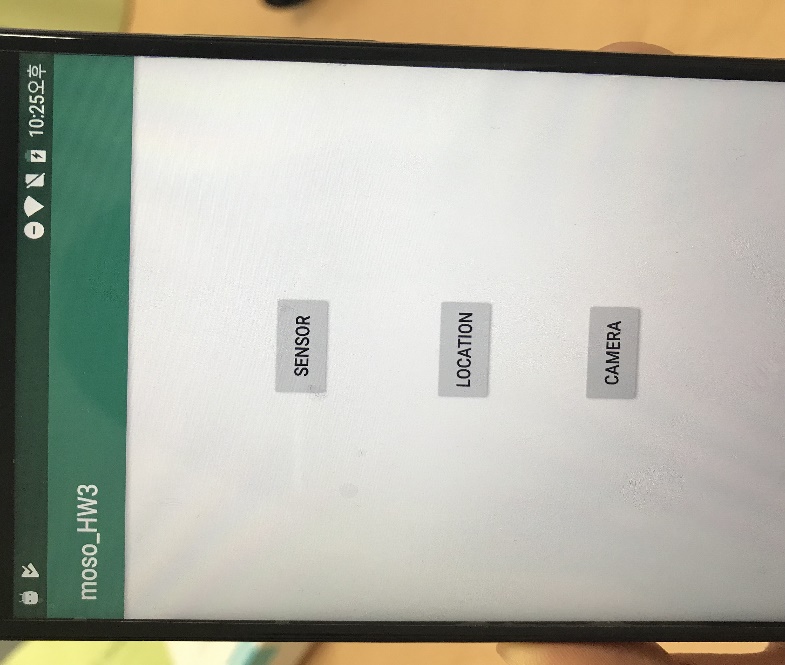


=> surfaceview에 찍은 사진을 보이기 위해서는 surfaceholder가 필요하다. 따라서 camera 와 surfaceholder 변수를 선언한다.



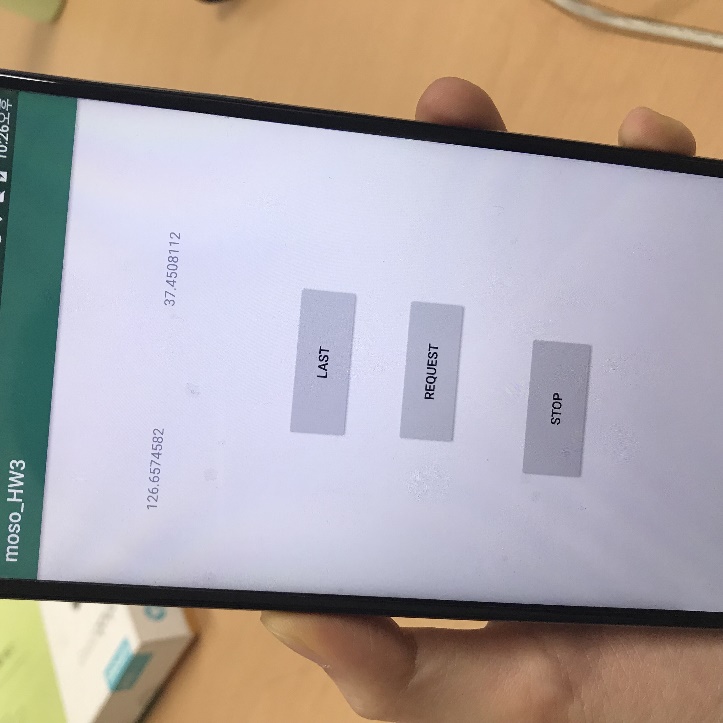
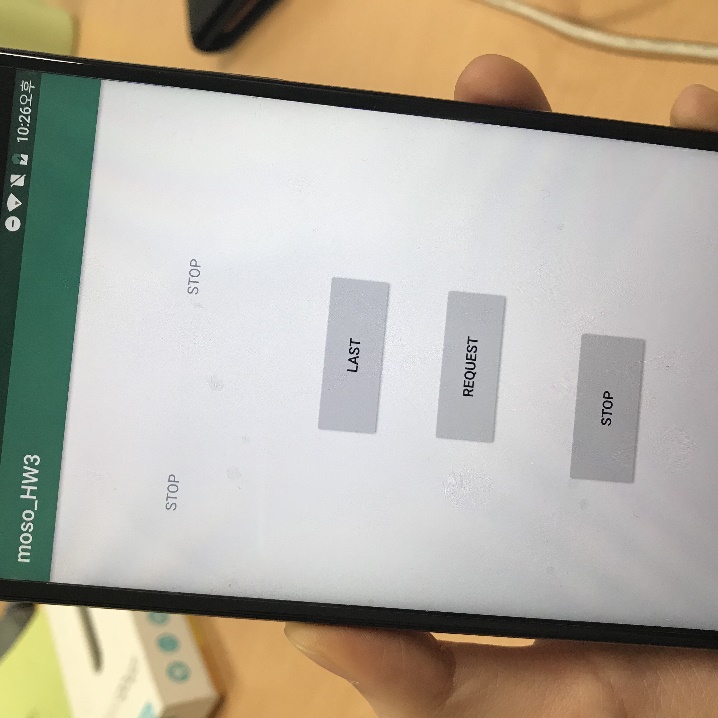
=> surfaceCreated 함수는 surfaceview가 생성될 때 발생하며, 카메라와 surfaceholder를 연결하고 카메라 preview를 시작한다. 이 때 setDisplayOrientation을 통해서 카메라가 정 방향으로 찍힐 수 있도록 추가해준다. 또한 surfaceChanged 함수는 상태가 변경될 때 마다 실행되는 함수로 카메라 preview를 재설정한다. 마지막으로 surfaceDestroyed 함수는 surfaceview 객체가 사라질 때 호출되는 함수로 카메라 리소스를 반환한다.

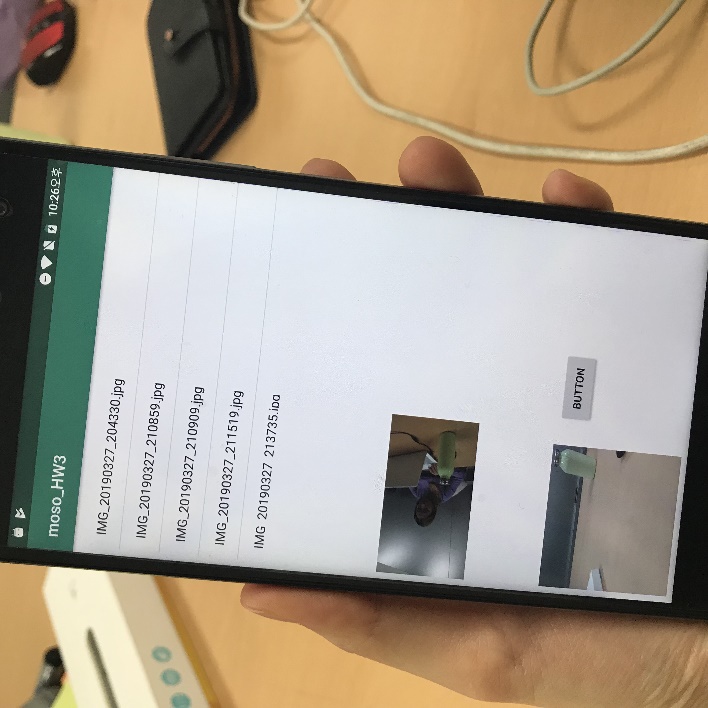
**[실습 내용 및 고찰]**



=> 앱을 처음 들어가면 첫번째 사진과 같이 세 개의 버튼이 나온다.

=> SENSOR 버튼을 눌렀을 때 왼쪽 리스트뷰에 받아온 센서 목록이 뜨고, 오른쪽 텍스트뷰에 ACCEL sensor와 GYRO sensor 의 x,y,z 축 값이 뜬다.



=> LOCATION 버튼을 눌렀을 때 LOCATION\_ACTIVITY에서 구현한대로 LAST, REQUEST, STOP 버튼이 나오고 맨 위에 textview 두개가 뜬다. LAST를 누르면 마지막 위치가 나오고, REQUEST를 누르면 위치를 설정한 주기마다 새롭게 알려준다. 그 후 STOP을 누르면 textview에 STOP 이 뜨면서 위치 업데이트가 멈춘다.

=> 마지막으로 CAMERA 버튼을 눌렀을 때 CAMERA\_ACTIVITY에서 구현한 화면이 나오는 것이다. 맨 위에는 내가 찍은 사진의 이름이 listview 형태로 순서대로 저장되어있다. 각 사진의 이름을 누르면 listview 밑에 위치한 imageview에 그 사진이 보여진다. 그리고 맨 아래는 카메라 화면을 보여주는 camerapreview가 구현되어 있다. 옆에 있는 Button을 누르면 사진이 찍히고, 바로 위의 listview에 업데이트 되는 것을 확인할 수 있었다.

=> 이번 과제에서는 앱에서 센서, 위치, 카메라 정보를 받아오는 것을 실습할 수 있었다. 매니페스트에 추가한 뒤에도 런타임 시 권한을 물어서 접근해야 하는 경우를 배우고, 직접 런타임 퍼미션을 구현해볼 수 있었다.