

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

T120B169 App Development for Smart Mobile Systems

**Report of labaratory work no.3**

**Dėstytojas: Rytis Maskeliūnas**

**Studentas: Kasparas Giniūnas IFF-6/4**

KAUNAS, 2018

Table of contents

[Tasks 3](#_Toc532810895)

[Implementation 4](#_Toc532810896)

[ANNEX 6](#_Toc532810897)

[ *MainActivity.java* file 6](#_Toc532810898)

[ *CameraActivity.java* file 9](#_Toc532810899)

[ *Activity\_main.xml* file 14](#_Toc532810900)

[ *Activity\_camera.xml* file 15](#_Toc532810901)

# Tasks

1. When testing the app you will notice that even without moving the device, the accelerometer data changes (because is not absolutely constant), so when you use data from the accelerometer, you should set a limit so that the slightest motion is not taken into account.
2. In the application, instead of the x, y, z values, show only the position of the smartphone over ground (orientation). For example, left side down, and up screen, etc
3. Create a compass and display the live compass onscreen.
4. Get the geo-position from the network (mobile operator & wireless network). On the phone screen this should be displayed next to the GPS coordinates for comparison
5. If the phone is oriented to the north, the application should run the Activity with the camera. It should automatically take a picture of the north (when compass shows north) and display it on screen.
6. \*When the smartphone is at 0 degrees, the brightness of the screen should be 0% (minimum value). If you change the position of the smartphone to 90 degrees (in a standing position), the brightness of the screen should increase to its maximum value. (9 points)
7. \*If the smartphone is oriented to the south at the 90-degree orientation position, the application should send an SOS signal using a camera flash (three short flashes, three long flashes, tree short flashes).(10 points)

# Implementation

1. When testing the app you will notice that even without moving the device, the accelerometer data changes (because is not absolutely constant), so when you use data from the accelerometer, you should set a limit so that the slightest motion is not taken into account.

Implemented high-pass filtering to filter out not stable accelerometer data

**final float** alpha = 0.97f;

**mGravity**[0] = alpha \* **mGravity**[0] + (1 - alpha) \* event.**values**[0];  
**mGravity**[1] = alpha \* **mGravity**[1] + (1 - alpha) \* event.**values**[1];  
**mGravity**[2] = alpha \* **mGravity**[2] + (1 - alpha) \* event.**values**[2];

1. In the application, instead of the x, y, z values, show only the position of the smartphone over ground (orientation). For example, left side down, and up screen, etc

Done by checking accelerometer values and orientation printed in *xValue* TextView element

**if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 10 && event.**values**[2] > 9){  
 **xValue**.setText(**"Screen up"**);  
}  
**if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < -9 && event.**values**[2] > -10){  
 **xValue**.setText(**"Screen down"**);  
  
 finish();  
}  
**if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 10 && event.**values**[1] > 9  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Vertical"**);  
}  
**if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < -9 && event.**values**[1] > -10  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Vertical (up side down)"**);  
}  
**if**(event.**values**[0] < -9 && event.**values**[0] > -10  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Right side down"**);  
}  
**if**(event.**values**[0] < 10 && event.**values**[0] > 9  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Left side down"**);  
}

1. Create a compass and display the live compass onscreen.

Compass implementation:

**if**(mySensor.getType() == Sensor.***TYPE\_MAGNETIC\_FIELD***) {  
  
  
 **mGeomagnetic**[0] = alpha \* **mGeomagnetic**[0] + (1 - alpha) \* event.**values**[0];  
 **mGeomagnetic**[1] = alpha \* **mGeomagnetic**[1] + (1 - alpha) \* event.**values**[1];  
 **mGeomagnetic**[2] = alpha \* **mGeomagnetic**[2] + (1 - alpha) \* event.**values**[2];  
  
 **float** R[] = **new float**[9];  
 **float** I[] = **new float**[9];  
  
 **boolean** success = SensorManager.*getRotationMatrix*(R, I, **mGravity**, **mGeomagnetic**);  
  
 **if**(success){  
 **float** orientation[] = **new float**[3];  
 SensorManager.*getOrientation*(R, orientation);  
 **azimuth** = (**float**)Math.*toDegrees*(orientation[0]);  
 **azimuth** = (**azimuth**+360)%360;  
 Animation animation = **new** RotateAnimation(-**currentAzimuth**, -**azimuth**, Animation.***RELATIVE\_TO\_SELF***, 0.5f, Animation.***RELATIVE\_TO\_SELF***, 0.5f);  
 **currentAzimuth** = **azimuth**;  
  
 animation.setDuration(500);  
 animation.setRepeatCount(0);  
 animation.setFillAfter(**true**);  
  
 **imageView**.startAnimation(animation);  
  
 **if**(**azimuth** < 1 && **azimuth** > 359){  
 startActivity(**new** Intent(MainActivity.**this**, CameraActivity.**class**));  
 }  
 }  
}

# ANNEX

## *MainActivity.java* file

**package** edu.ktu.lab3;  
  
**import** android.Manifest;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.graphics.Camera;  
**import** android.graphics.ImageFormat;  
**import** android.graphics.SurfaceTexture;  
**import** android.hardware.Sensor;  
**import** android.hardware.SensorEvent;  
**import** android.hardware.SensorEventListener;  
**import** android.hardware.SensorManager;  
**import** android.hardware.camera2.CameraAccessException;  
**import** android.hardware.camera2.CameraCaptureSession;  
**import** android.hardware.camera2.CameraCharacteristics;  
**import** android.hardware.camera2.CameraDevice;  
**import** android.hardware.camera2.CameraManager;  
**import** android.hardware.camera2.CameraMetadata;  
**import** android.hardware.camera2.CaptureRequest;  
**import** android.hardware.camera2.TotalCaptureResult;  
**import** android.hardware.camera2.params.StreamConfigurationMap;  
**import** android.location.Location;  
**import** android.location.LocationListener;  
**import** android.location.LocationManager;  
**import** android.media.Image;  
**import** android.media.ImageReader;  
**import** android.os.Build;  
**import** android.os.Environment;  
**import** android.os.Handler;  
**import** android.os.HandlerThread;  
**import** android.support.annotation.NonNull;  
**import** android.support.annotation.RequiresApi;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.app.LoaderManager;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.util.Size;  
**import** android.util.SparseIntArray;  
**import** android.view.Surface;  
**import** android.view.TextureView;  
**import** android.view.View;  
**import** android.view.animation.Animation;  
**import** android.view.animation.RotateAnimation;  
**import** android.widget.Button;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileOutputStream;  
**import** java.io.IOException;  
**import** java.io.OutputStream;  
**import** java.nio.ByteBuffer;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.List;  
  
  
**public class** MainActivity **extends** AppCompatActivity **implements** SensorEventListener, LocationListener {  
  
 **private** SensorManager **senSensorManager**;  
 **private** Sensor **senAccelerometer**;  
 **private** Sensor **senCompass**;  
 **private** LocationManager **locationManager**;  
 **private** TextView **xValue**;  
 **private** TextView **networkCoord**;  
 **private** TextView **zValue**;  
 **private** TextView **coordinates**;  
 **private** Button **startAndStop**;  
 **private boolean InformationObtained**;  
  
 **private** Button **openCamera**;  
  
 **private** ImageView **imageView**;  
 **private** TextView **angleView**;  
 **private float azimuth** = 0f;  
 **private float currentAzimuth** = 0f;  
 **private float**[] **mGeomagnetic** = **new float**[3];  
 **private float**[] **mGravity** = **new float**[3];  
  
 **private** LocationManager **networkLocation**;  
  
 **private static final** SparseIntArray ***ORIENTATIONS*** = **new** SparseIntArray();  
 **static** {  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_0***, 90);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_90***, 0);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_180***, 270);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_270***, 180);  
 }  
  
 **private double**[] **previousValues** = {0,0,0};  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **senSensorManager** = (SensorManager) getSystemService(Context.***SENSOR\_SERVICE***);  
 **senAccelerometer** = **senSensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);  
 **senCompass** = **senSensorManager**.getDefaultSensor(Sensor.***TYPE\_MAGNETIC\_FIELD***);  
 **locationManager** = (LocationManager)getSystemService(Context.***LOCATION\_SERVICE***);  
 **networkLocation** = (LocationManager)getSystemService(Context.***LOCATION\_SERVICE***);  
  
 **startAndStop** = (Button)findViewById(R.id.***start\_and\_stop***);  
 **startAndStop**.setOnClickListener(**StartAndStopButtonListener**);  
  
 **xValue** = (TextView)findViewById(R.id.***x\_value***);  
 **networkCoord** = (TextView)findViewById(R.id.***network\_coordinates***);  
  
 **coordinates** = (TextView)findViewById(R.id.***coordinates***);  
  
 **senSensorManager** = (SensorManager) getSystemService(Context.***SENSOR\_SERVICE***);  
 **senAccelerometer** = **senSensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);  
  
 **openCamera** = (Button)findViewById(R.id.***open\_camera***);  
 **openCamera**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 startActivity(**new** Intent(MainActivity.**this**, CameraActivity.**class**));  
 }  
 });  
  
 **imageView** = (ImageView)findViewById(R.id.***compass***);  
  
 **senSensorManager**.registerListener(MainActivity.**this**, **senCompass**, SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
  
 View.OnClickListener **StartAndStopButtonListener** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if**(**senAccelerometer** == **null**){  
 Toast.*makeText*(MainActivity.**this**, getString(R.string.***no\_sensor***), Toast.***LENGTH\_LONG***).show();  
 }  
  
 **if**(**InformationObtained**){  
 **startAndStop**.setText(getString(R.string.***start***));  
 **senSensorManager**.unregisterListener(MainActivity.**this**, **senAccelerometer**);  
 **InformationObtained** = **false**;  
 } **else** {  
 **senSensorManager**.registerListener(MainActivity.**this**, **senAccelerometer**, SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 **startAndStop**.setText(getString(R.string.***stop***));  
 **InformationObtained** = **true**;  
 }  
 }  
 };  
  
  
 @Override  
 **public void** onSensorChanged(SensorEvent event){  
 Sensor mySensor = event.**sensor**;  
 **final float** alpha = 0.97f;  
  
 **synchronized** (**this**){  
 **if**(mySensor.getType() == Sensor.***TYPE\_ACCELEROMETER***){  
 **mGravity**[0] = alpha \* **mGravity**[0] + (1 - alpha) \* event.**values**[0];  
 **mGravity**[1] = alpha \* **mGravity**[1] + (1 - alpha) \* event.**values**[1];  
 **mGravity**[2] = alpha \* **mGravity**[2] + (1 - alpha) \* event.**values**[2];  
  
 **if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 10 && event.**values**[2] > 9){  
 **xValue**.setText(**"Screen up"**);  
 }  
 **if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < -9 && event.**values**[2] > -10){  
 **xValue**.setText(**"Screen down"**);  
  
 finish();  
 }  
 **if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < 10 && event.**values**[1] > 9  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Vertical"**);  
 }  
 **if**(event.**values**[0] < 1 && event.**values**[0] > -1  
 && event.**values**[1] < -9 && event.**values**[1] > -10  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Vertical (up side down)"**);  
 }  
 **if**(event.**values**[0] < -9 && event.**values**[0] > -10  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Right side down"**);  
 }  
 **if**(event.**values**[0] < 10 && event.**values**[0] > 9  
 && event.**values**[1] < 1 && event.**values**[1] > -1  
 && event.**values**[2] < 1 && event.**values**[2] > -1){  
 **xValue**.setText(**"Left side down"**);  
 }  
 }  
  
 **if**(mySensor.getType() == Sensor.***TYPE\_MAGNETIC\_FIELD***) {  
  
  
 **mGeomagnetic**[0] = alpha \* **mGeomagnetic**[0] + (1 - alpha) \* event.**values**[0];  
 **mGeomagnetic**[1] = alpha \* **mGeomagnetic**[1] + (1 - alpha) \* event.**values**[1];  
 **mGeomagnetic**[2] = alpha \* **mGeomagnetic**[2] + (1 - alpha) \* event.**values**[2];  
  
 **float** R[] = **new float**[9];  
 **float** I[] = **new float**[9];  
  
 **boolean** success = SensorManager.*getRotationMatrix*(R, I, **mGravity**, **mGeomagnetic**);  
  
 **if**(success){  
 **float** orientation[] = **new float**[3];  
 SensorManager.*getOrientation*(R, orientation);  
 **azimuth** = (**float**)Math.*toDegrees*(orientation[0]);  
 **azimuth** = (**azimuth**+360)%360;  
 Animation animation = **new** RotateAnimation(-**currentAzimuth**, -**azimuth**, Animation.***RELATIVE\_TO\_SELF***, 0.5f, Animation.***RELATIVE\_TO\_SELF***, 0.5f);  
 **currentAzimuth** = **azimuth**;  
  
 animation.setDuration(500);  
 animation.setRepeatCount(0);  
 animation.setFillAfter(**true**);  
  
 **imageView**.startAnimation(animation);  
  
 **if**(**azimuth** < 1 && **azimuth** > 359){  
 startActivity(**new** Intent(MainActivity.**this**, CameraActivity.**class**));  
 }  
 }  
 }  
 }  
  
 }  
  
 @Override  
 **public void** onAccuracyChanged(Sensor sensor, **int** accuracy){  
  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***JELLY\_BEAN\_MR2***)  
 @Override  
 **protected void** onPause(){  
 **super**.onPause();  
 **if**(**senAccelerometer** != **null**){  
 **senSensorManager**.unregisterListener(MainActivity.**this**, **senAccelerometer**);  
 }  
  
 **if**(**senCompass** != **null**){  
 **senSensorManager**.unregisterListener(MainActivity.**this**, **senCompass**);  
 }  
  
 **if**(ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** &&  
 ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** ){  
 **return**;  
 }  
  
 **this**.**locationManager**.removeUpdates(**this**);  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **protected void** onResume(){  
 **super**.onResume();  
 **if**(**senAccelerometer** != **null** && **InformationObtained**){  
 **senSensorManager**.registerListener(MainActivity.**this**, **senAccelerometer**, SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
  
 **if**(**senCompass** != **null**){  
 **senSensorManager**.registerListener(MainActivity.**this**, **senCompass**, SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
  
 **if**(ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** &&  
 ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** ){  
 **return**;  
 }  
  
 **this**.**locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***, 400, 1, **this**);  
  
 Location net\_loc;  
  
 **if**(**networkLocation**.isProviderEnabled(LocationManager.***NETWORK\_PROVIDER***)){  
 net\_loc = **networkLocation**.getLastKnownLocation(LocationManager.***NETWORK\_PROVIDER***);  
 }  
 }  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
 **if**(location != **null**){  
 **coordinates**.setText(getString(R.string.***Latitude\_text***) + **" "** + location.getLatitude()+ **" \n"** + getString(R.string.***Longitude\_text***) + **" "** + location.getLongitude());  
 }  
 }  
  
 @Override  
 **public void** onStatusChanged(String provider, **int** status, Bundle extras) {  
  
 }  
  
 @Override  
 **public void** onProviderEnabled(String provider) {  
  
 }  
  
 @Override  
 **public void** onProviderDisabled(String provider) {  
  
 }  
}

## *CameraActivity.java* file

**package** edu.ktu.lab3;  
  
**import** android.Manifest;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.graphics.ImageFormat;  
**import** android.graphics.SurfaceTexture;  
**import** android.hardware.SensorManager;  
**import** android.hardware.camera2.CameraAccessException;  
**import** android.hardware.camera2.CameraCaptureSession;  
**import** android.hardware.camera2.CameraCharacteristics;  
**import** android.hardware.camera2.CameraDevice;  
**import** android.hardware.camera2.CameraManager;  
**import** android.hardware.camera2.CameraMetadata;  
**import** android.hardware.camera2.CaptureRequest;  
**import** android.hardware.camera2.TotalCaptureResult;  
**import** android.hardware.camera2.params.StreamConfigurationMap;  
**import** android.location.LocationManager;  
**import** android.media.Image;  
**import** android.media.ImageReader;  
**import** android.os.Build;  
**import** android.os.Environment;  
**import** android.os.Handler;  
**import** android.os.HandlerThread;  
**import** android.support.annotation.NonNull;  
**import** android.support.annotation.RequiresApi;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.util.Size;  
**import** android.util.SparseIntArray;  
**import** android.view.Surface;  
**import** android.view.TextureView;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** java.io.File;  
**import** java.io.FileOutputStream;  
**import** java.io.IOException;  
**import** java.io.OutputStream;  
**import** java.nio.ByteBuffer;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.List;  
  
**public class** CameraActivity **extends** AppCompatActivity {  
  
 **private** Button **takePictureButton**;  
 **private** TextureView **textureView**;  
 **private static final** SparseIntArray ***ORIENTATIONS*** = **new** SparseIntArray();  
 **static** {  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_0***, 90);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_90***, 0);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_180***, 270);  
 ***ORIENTATIONS***.append(Surface.***ROTATION\_270***, 180);  
 }  
 **private** String **cameraId**;  
 **protected** CameraDevice **cameraDevice**;  
 **protected** CameraCaptureSession **cameraCaptureSessions**;  
 **protected** CaptureRequest.Builder **captureRequestBuilder**;  
 **private** Size **imageDimention**;  
 **private** ImageReader **imageReader**;  
 **private** File **file**;  
 **private static final int *REQUEST\_CAMERA\_PERMISSION*** = 200;  
 **private** Handler **mBackgroundHandler**;  
 **private** HandlerThread **mBackgroundThread**;  
 **private static final** String ***TAG*** = **"AndroidCameraApi"**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_camera***);  
  
 **textureView** = (TextureView)findViewById(R.id.***textureView***);  
 **assert textureView** != **null**;  
 **textureView**.setSurfaceTextureListener(**textureListener**);  
  
 **takePictureButton** = (Button)findViewById(R.id.***take\_photo***);  
 **assert takePictureButton** != **null**;  
 **takePictureButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 takePicture();  
 }  
 });  
 }  
  
 @Override  
 **protected void** onResume(){  
 **super**.onResume();  
  
 startBackgoundThread();  
  
 **if**(**textureView**.isAvailable()){  
 openCamera();  
 } **else** {  
 **textureView**.setSurfaceTextureListener(**textureListener**);  
 }  
 }  
  
 @Override  
 **protected void** onPause(){  
 **super**.onPause();  
  
  
 stopBackgroundThread();  
 }  
  
 TextureView.SurfaceTextureListener **textureListener** = **new** TextureView.SurfaceTextureListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **public void** onSurfaceTextureAvailable(SurfaceTexture surface, **int** width, **int** height) {  
 openCamera();  
 }  
  
 @Override  
 **public void** onSurfaceTextureSizeChanged(SurfaceTexture surface, **int** width, **int** height) {  
  
 }  
  
 @Override  
 **public boolean** onSurfaceTextureDestroyed(SurfaceTexture surface) {  
 **return false**;  
 }  
  
 @Override  
 **public void** onSurfaceTextureUpdated(SurfaceTexture surface) {  
  
 }  
 };  
  
 **private final** CameraDevice.StateCallback **stateCallback** = **new** CameraDevice.StateCallback() {  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **public void** onOpened(@NonNull CameraDevice camera) {  
 Log.*e*(***TAG***, **"onOpened"**);  
 **cameraDevice** = camera;  
 createCameraPreview();  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **public void** onDisconnected(@NonNull CameraDevice camera) {  
 **cameraDevice**.close();  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **public void** onError(@NonNull CameraDevice camera, **int** error) {  
 **cameraDevice**.close();  
 **cameraDevice** = **null**;  
 }  
 };  
  
 **final** CameraCaptureSession.CaptureCallback **captureCallbackListener** = **new** CameraCaptureSession.CaptureCallback() {  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 @Override  
 **public void** onCaptureCompleted(@NonNull CameraCaptureSession session, @NonNull CaptureRequest request, @NonNull TotalCaptureResult result) {  
 **super**.onCaptureCompleted(session, request, result);  
 Toast.*makeText*(CameraActivity.**this**, **"Saved:"** + **file**, Toast.***LENGTH\_SHORT***).show();  
 createCameraPreview();  
 }  
 };  
  
 **protected void** startBackgoundThread(){  
 **mBackgroundThread** = **new** HandlerThread(**"Camera background"**);  
 **mBackgroundThread**.start();  
 **mBackgroundHandler** = **new** Handler(**mBackgroundThread**.getLooper());  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***JELLY\_BEAN\_MR2***)  
 **protected void** stopBackgroundThread(){  
 **mBackgroundThread**.quitSafely();  
 **try**{  
 **mBackgroundThread**.join();  
 **mBackgroundThread** = **null**;  
 **mBackgroundHandler** = **null**;  
 } **catch**(InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 **protected void** takePicture() {  
  
 **if** (**null** == **cameraDevice**) {  
 Log.*e*(***TAG***, **"Camera device is null"**);  
 }  
  
 CameraManager manager = (CameraManager) getSystemService(Context.***CAMERA\_SERVICE***);  
 **try** {  
 CameraCharacteristics characteristics = manager.getCameraCharacteristics(**cameraDevice**.getId());  
 Size[] jpegSizes = **null**;  
  
 **if** (characteristics != **null**) {  
 jpegSizes = characteristics.get(CameraCharacteristics.***SCALER\_STREAM\_CONFIGURATION\_MAP***).getOutputSizes(ImageFormat.***JPEG***);  
 }  
 **int** width = 640;  
 **int** height = 480;  
 **if** (jpegSizes != **null** && 0 < jpegSizes.**length**) {  
 width = jpegSizes[0].getWidth();  
 height = jpegSizes[0].getHeight();  
 }  
 ImageReader reader = ImageReader.*newInstance*(width, height, ImageFormat.***JPEG***, 1);  
 List<Surface> outputSurfaces = **new** ArrayList<Surface>(2);  
 outputSurfaces.add(reader.getSurface());  
 outputSurfaces.add(**new** Surface(**textureView**.getSurfaceTexture()));  
  
 **final** CaptureRequest.Builder captureBuilder = **cameraDevice**.createCaptureRequest(CameraDevice.***TEMPLATE\_STILL\_CAPTURE***);  
 captureBuilder.addTarget(reader.getSurface());  
  
 captureBuilder.set(CaptureRequest.***CONTROL\_MODE***, CameraMetadata.***CONTROL\_MODE\_AUTO***);  
  
 **int** rotation = getWindowManager().getDefaultDisplay().getRotation();  
 captureBuilder.set(CaptureRequest.***JPEG\_ORIENTATION***, ***ORIENTATIONS***.get(rotation));  
  
 **final** File file = **new** File(Environment.*getExternalStorageDirectory*() + **"/pic.jpg"**);  
  
  
 ImageReader.OnImageAvailableListener readerListener = **new** ImageReader.OnImageAvailableListener() {  
 @RequiresApi(api = Build.VERSION\_CODES.***KITKAT***)  
 @Override  
 **public void** onImageAvailable(ImageReader reader) {  
 **try** (Image image = reader.acquireLatestImage()) {  
 ByteBuffer buffer = image.getPlanes()[0].getBuffer();  
 **byte**[] bytes = **new byte**[buffer.capacity()];  
 buffer.get(bytes);  
 save(bytes);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private void** save(**byte**[] bytes) **throws** IOException {  
 **try** (OutputStream output = **new** FileOutputStream(file)) {  
 output.write(bytes);  
 }  
 }  
 };  
  
 reader.setOnImageAvailableListener(readerListener, **mBackgroundHandler**);  
  
 **final** CameraCaptureSession.CaptureCallback captureListener = **new** CameraCaptureSession.CaptureCallback() {  
 @Override  
 **public void** onCaptureCompleted(@NonNull CameraCaptureSession session, @NonNull CaptureRequest request, @NonNull TotalCaptureResult result) {  
 **super**.onCaptureCompleted(session, request, result);  
 Toast.*makeText*(CameraActivity.**this**, **"Saved: "** + file, Toast.***LENGTH\_SHORT***).show();  
 createCameraPreview();  
 }  
 };  
  
  
  
 **cameraDevice**.createCaptureSession(outputSurfaces, **new** CameraCaptureSession.StateCallback() {  
 @Override  
 **public void** onConfigured(@NonNull CameraCaptureSession session) {  
 **try** {  
 session.capture(captureBuilder.build(), captureListener, **mBackgroundHandler**);  
 } **catch** (CameraAccessException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 **public void** onConfigureFailed(@NonNull CameraCaptureSession session) {  
  
 }  
 }, **mBackgroundHandler**);  
  
 } **catch**(CameraAccessException e){  
 e.printStackTrace();  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.***LOLLIPOP***)  
 **protected void** createCameraPreview(){  
 **try**{  
 SurfaceTexture texture = **textureView**.getSurfaceTexture();  
 **assert** texture != **null**;  
 texture.setDefaultBufferSize(**imageDimention**.getWidth(), **imageDimention**.getHeight());  
 Surface surface = **new** Surface(texture);  
 **captureRequestBuilder** = **cameraDevice**.createCaptureRequest(CameraDevice.***TEMPLATE\_PREVIEW***);  
 **captureRequestBuilder**.addTarget(surface);  
 **cameraDevice**.createCaptureSession(Arrays.*asList*(surface), **new** CameraCaptureSession.StateCallback() {  
 @Override  
 **public void** onConfigured(@NonNull CameraCaptureSession session) {  
 **if**(**null** == **cameraDevice**){  
 **return**;  
 }  
  
 **cameraCaptureSessions** = session;  
 updatePreview();  
 }  
  
 @Override  
 **public void** onConfigureFailed(@NonNull CameraCaptureSession session) {  
 Toast.*makeText*(CameraActivity.**this**, **"Configuration change"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }, **null**);  
 } **catch** (CameraAccessException e){  
 e.printStackTrace();  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)  
 **private void** openCamera(){  
 CameraManager manager = (CameraManager)getSystemService(Context.CAMERA\_SERVICE);  
 Log.e(TAG, **"is camera open"**);  
  
 **try**{  
 cameraId = manager.getCameraIdList()[0];  
 CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);  
 StreamConfigurationMap map = characteristics.get(CameraCharacteristics.SCALER\_STREAM\_CONFIGURATION\_MAP);  
 **assert** map != **null**;  
 imageDimention = map.getOutputSizes(SurfaceTexture.**class**)[0];  
  
 **if**(ActivityCompat.checkSelfPermission(**this**, Manifest.permission.CAMERA) != PackageManager.PERMISSION\_GRANTED  
 && ActivityCompat.checkSelfPermission(**this**, Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED){  
 ActivityCompat.requestPermissions(CameraActivity.**this**, **new** String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE\_EXTERNAL\_STORAGE},  
 REQUEST\_CAMERA\_PERMISSION);  
 **return**;  
 }  
 manager.openCamera(cameraId, stateCallback, **null**);  
 } **catch** (CameraAccessException e){  
 e.printStackTrace();  
 }  
 Log.e(TAG, **"openCamera X"**);  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)  
 **protected void** updatePreview(){  
 **if**(**null** == cameraDevice){  
 Log.e(TAG, **"updatePreview error, return"**);  
 }  
  
 captureRequestBuilder.set(CaptureRequest.CONTROL\_MODE, CameraMetadata.CONTROL\_MODE\_AUTO);  
 **try**{  
 cameraCaptureSessions.setRepeatingRequest(captureRequestBuilder.build(), **null**, mBackgroundHandler);  
 } **catch** (CameraAccessException e){  
 e.printStackTrace();  
 }  
 }  
  
 @RequiresApi(api = Build.VERSION\_CODES.LOLLIPOP)  
 **private void** closeCamera(){  
 **if**(**null** != cameraDevice){  
 cameraDevice.close();  
 cameraDevice = **null**;  
 }  
 **if**(**null** != imageReader){  
 imageReader.close();  
 imageReader = **null**;  
 }  
 }  
  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults){  
 **if**(requestCode == REQUEST\_CAMERA\_PERMISSION){  
 **if**(grantResults[0] == PackageManager.PERMISSION\_DENIED){  
 Toast.makeText(CameraActivity.**this**, **"Sorry, you can't use this app without grantin permission"**, Toast.LENGTH\_SHORT).show();  
 *//finish();* }  
 }  
 }  
}

## *Activity\_main.xml* file

*<?***xml version="1.0" encoding="utf-8"***?>*<**ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:app="http://schemas.android.com/apk/res-auto"**>  
  
<**LinearLayout  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
  
 <**TableLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**TableRow android:padding="10dp"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:paddingEnd="5dp"  
 android:paddingRight="5dp"  
 android:text="Orietation: "** />  
  
 <**TextView  
 android:id="@+id/x\_value"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="-"** />  
  
 </**TableRow**>  
  
 <**TableRow android:padding="10dp"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:paddingEnd="5dp"  
 android:paddingRight="5dp"  
 android:text="Coordinates"** />  
  
 <**TextView  
 android:id="@+id/coordinates"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="-"** />  
  
 </**TableRow**>  
  
 <**TableRow android:padding="10dp"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:paddingEnd="5dp"  
 android:paddingRight="5dp"  
 android:text="Coordinates(network)"** />  
  
 <**TextView  
 android:id="@+id/network\_coordinates"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="-"** />  
  
 </**TableRow**>  
  
 </**TableLayout**>  
  
 <**Button  
 android:id="@+id/start\_and\_stop"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="start"** />  
  
  
  
  
 <**ImageView  
 android:id="@+id/compass"  
 android:layout\_width="250dp"  
 android:layout\_height="250dp"  
  
 android:src="@drawable/compass"  
 android:layout\_gravity="center\_horizontal"**/>  
  
 <**Button  
 android:id="@+id/open\_camera"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="OPEN CAMERA"** />  
  
  
  
</**LinearLayout**>  
</**ScrollView**>

## *Activity\_camera.xml* file

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".CameraActivity"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**Button  
 android:id="@+id/take\_photo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Take photo"** />  
  
 <**FrameLayout  
 android:id="@+id/layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextureView  
 android:id="@+id/textureView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"** />  
  
 </**FrameLayout**>  
 </**LinearLayout**>  
  
  
  
</**android.support.constraint.ConstraintLayout**>