App Development for Smart Mobile Systems

Laboratory work 3, Elena Morelli

1. *When testing the app you will not 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.*

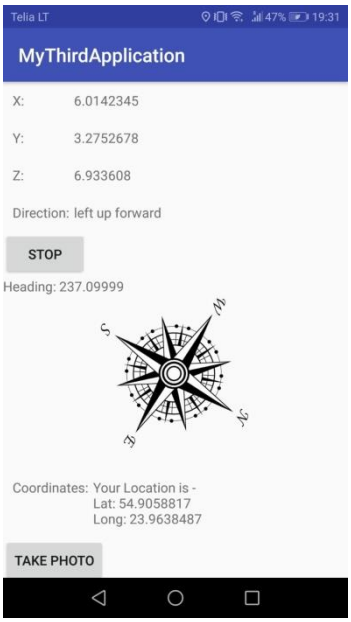
```
Sensor mySensor = sensorEvent.sensor;

String dir_x;
String dir_y;
String dir_z;



if(mySensor.getType() == Sensor.TYPE_ACCELEROMETER){
    if(sensorEvent.values[0] > prev_x +0.300 || sensorEvent.values[0] < prev_x -
0.300){
        xValue.setText(String.valueOf(sensorEvent.values[0]));
        prev_x = sensorEvent.values[0];
    }

    if(sensorEvent.values[1] > prev_y +0.300 || sensorEvent.values[1] < prev_y -
0.300){
        yValue.setText(String.valueOf(sensorEvent.values[1]));
        prev_y = sensorEvent.values[1];
    }

    if(sensorEvent.values[2] > prev_z +0.300 || sensorEvent.values[2] < prev_z -
0.300){
        zValue.setText(String.valueOf(sensorEvent.values[2]));
        prev_z = sensorEvent.values[2];
    }
```

A new parameter has been added to save the value that has been previously shown in the view, if the difference between the old parameter and the new mesure differs more than 0.300, the text view will be updated

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

```
if (sensorEvent.values[0] > 0){
    dir_x = "left";
} else if (sensorEvent.values[0] == 0.0){
    dir_x = "center";
} else {
    dir_x = "right";
}

if (sensorEvent.values[1] > 0){
    dir_y = "up";
} else if (sensorEvent.values[1] == 0.0) {
    dir_y = "center";
} else {
```

```
    dir_y = "down";
}

if (sensorEvent.values[2]> 0){
    dir_z = "forward";
} else if (sensorEvent.values[2] == 0.0) {
    dir_z = "center";
} else {
    dir_z = "back";
}

orientation_Y = sensorEvent.values[1];
direction.setText(dir_x + " " + dir_y + " " + dir_z);
```



3. *Create a compass and display the live compass on screen.*

```
senCompass = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
```

Created a new Sensor of orientation type for the compass

```
float degree = Math.round(sensorEvent.values[0]);

compass.setText("Heading: " + Float.toString(sensorEvent.values[0]));
compass_or = sensorEvent.values[0];

// create a rotation animation (reverse turn degree degrees)
RotateAnimation ra = new RotateAnimation(
        currentDegree,
        -degree,
        Animation.RELATIVE_TO_SELF, 0.5f,
        Animation.RELATIVE_TO_SELF,
        0.5f);

// how long the animation will take place
ra.setDuration(210);

// set the animation after the end of the reservation status
ra.setFillAfter(true);

// Start the animation
```

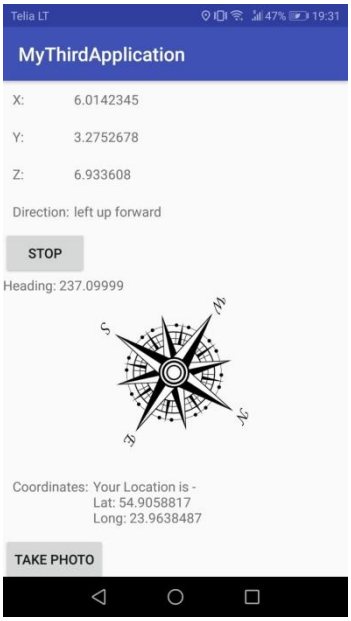
```
image.startAnimation(ra);
currentDegree = -degree;
```

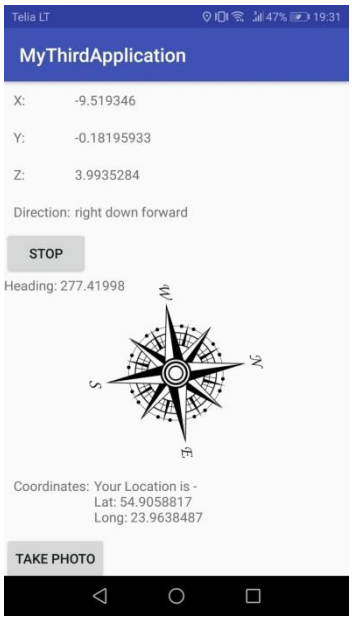
If the sensorEvent received is orientation type, then the compass animation will start

```
sensorManager.unregisterListener(MainActivity.this,senCompass);
```

```
sensorManager.registerListener(MainActivity.this,senCompass,SensorManager.SENSOR
_DELAY_GAME);
```

registration and un registration in the onPause and onResume method and when the button will start or stop the listener;





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.*

```
public class GPSTracker extends Service {

    private Context mContext;

    // Flag for GPS status
    boolean isGPSEnabled = false;

    // Flag for network status
    boolean isNetworkEnabled = false;

    // Flag for GPS status
    boolean canGetLocation = false;

    Location location; // Location
    double latitude; // Latitude
    double longitude; // Longitude

    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 1000; // 10
meters

    // The minimum time between updates in milliseconds
```

```
 private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

    Activity activity;

    public GPSTracker() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    public GPSTracker(Context context, Activity activity) {
        this.mContext = context;
        this.activity = activity;
        getLocation();
    }

    public Location getLocation() {
        try {


            locationManager = (LocationManager)
mContext.getSystemService(LOCATION_SERVICE);

            // Getting GPS status
            isGPSEnabled = locationManager
                    .isProviderEnabled(LocationManager.GPS_PROVIDER);

            // Getting network status
            isNetworkEnabled = locationManager
                    .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
                // No network provider is enabled
            } else {
                this.canGetLocation = true;
                if (isNetworkEnabled) {
                    int requestPermissionsCode = 50;
                    if (ContextCompat.checkSelfPermission(activity,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(activity,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                        ActivityCompat.requestPermissions(activity, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 50);
                    }
                    locationManager.requestLocationUpdates(
                            LocationManager.NETWORK_PROVIDER,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, mLocationListener);
                    Log.d("Network", "Network");
                    if (locationManager != null) {
                        location = locationManager

.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                        if (location != null) {
                            latitude = location.getLatitude();
                            longitude = location.getLongitude();
                        }
                    }
                }
            }
            // If GPS enabled, get latitude/longitude using GPS Services
            if (isGPSEnabled) {
```

```
                if (location == null) {
                    if (ContextCompat.checkSelfPermission(activity,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(activity,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                        ActivityCompat.requestPermissions(activity, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 50);

                    } else {
                        locationManager.requestLocationUpdates(
                                LocationManager.GPS_PROVIDER,
                                MIN_TIME_BW_UPDATES,
                                MIN_DISTANCE_CHANGE_FOR_UPDATES,
mLocationListener);
                        Log.d("GPS Enabled", "GPS Enabled");
                        if (locationManager != null) {

                            location = locationManager

.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if (location != null) {
                                latitude = location.getLatitude();
                                longitude = location.getLongitude();
                            }
                        }
                    }
                }

            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return location;
    }


    /**
     * Stop using GPS listener
     * Calling this function will stop using GPS in your app.
     * */
    public void stopUsingGPS() {

    }


    private final LocationListener mLocationListener = new LocationListener() {
        @Override
        public void onLocationChanged(final Location location) {

            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
            }
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras)
{

        }

        @Override
        public void onProviderEnabled(String provider) {

        }
```

```
        @Override
        public void onProviderDisabled(String provider) {

        }
    };

    /**
     * Function to get latitude
     * */
    public double getLatitude(){
        if(location != null){
            latitude = location.getLatitude();
        }

        // return latitude
        return latitude;
    }


    /**
     * Function to get longitude
     * */
    public double getLongitude(){
        if(location != null){
            longitude = location.getLongitude();
        }

        // return longitude
        return longitude;
    }

    /**
     * Function to check GPS/Wi-Fi enabled
     * @return boolean
     * */
    public boolean canGetLocation() {
        return this.canGetLocation;
    }


    /**
     * Function to show settings alert dialog.
     * On pressing the Settings button it will launch Settings Options.
     * */
    public void showSettingsAlert() {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

        // Setting Dialog Title
        alertDialog.setTitle("GPS is settings");

        // Setting Dialog Message
        alertDialog.setMessage("GPS is not enabled. Do you want to go to
settings menu?");

        // On pressing the Settings button.
        alertDialog.setPositiveButton("Settings", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                mContext.startActivity(intent);
            }
        });

        // On pressing the cancel button
        alertDialog.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
```

```
                dialog.cancel();
            }
        });

        // Showing Alert Message
        alertDialog.show();
    }
}
```

Created a GPSTracker service because in the phone it wasn't working the normal listener.

```
public void gpsLocation(){
    if (ContextCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);

    } else {
        Toast.makeText(mContext,"You need have granted
permission",Toast.LENGTH_SHORT).show();
        gps = new GPSTracker(mContext, MainActivity.this);

        // Check if GPS enabled
        if (gps.canGetLocation()) {

            double latitude = gps.getLatitude();
            double longitude = gps.getLongitude();

            // \n is for new line
            coordinates.setText("Your Location is - \nLat: " + latitude +
"\nLong: " + longitude);
            // Toast.makeText(getApplicationContext(), "Your Location is -
\nLat: " + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
        } else {
            // Can't get location.
            // GPS or network is not enabled.
            // Ask user to enable GPS/network in settings.
            gps.showSettingsAlert();
        }
    }
}
```

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*

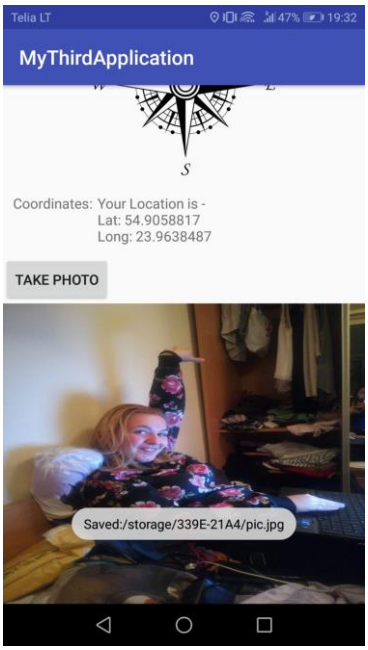
```
    if (sensorEvent.values[0] < 0.2 ){
        isNorth = true;
        reset();
        takePicture();

    }

}
```

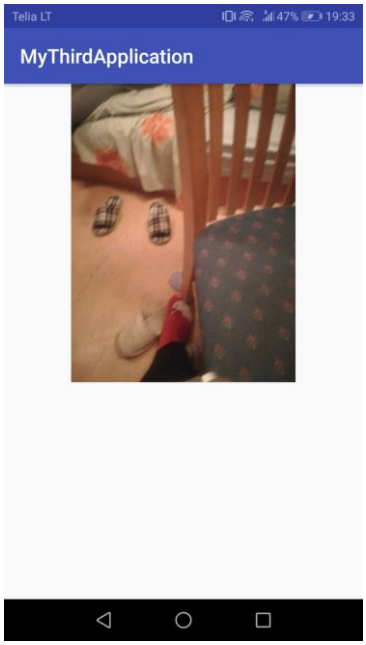
If the sensorEvent is type orientation and it is north oriented that the phone will take a picture.

```
} finally {
    if(null != output){
        output.close();
        if(isNorth){
            isNorth = false;
            changeActivity();
        }
    }
}
```

```
private void changeActivity(){
    Intent intent = new Intent(this, CameraActivity.class);
    startActivity(intent);
}
```

After the picture will be saved in the phone and the boolean parameter is true a the picture will be shown in the next activity.





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.*

```
if (sensorEvent.values[1] < 1){
    Settings.System.putInt(
            this.getContentResolver(),
            Settings.System.SCREEN_BRIGHTNESS,
            0
    );

} else if(sensorEvent.values[1] > 8){
    Settings.System.putInt(
            this.getContentResolver(),
            Settings.System.SCREEN_BRIGHTNESS,
            255
    );
}
```

To change the brighness the permission to change the phone setting was added to the manifest

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 short flashes).*

```
<permission android:name="android.permission.FLASHLIGHT"/>
```

```
if(compass_or > 179.5 && compass_or < 180.2 ){
    if( orientation_Y > 9.4 && orientation_Y < 9.8){
        closeCamera();
        sosLight();
        openCamera();
    }

}
```

The most important thing is to close and open the camera, otherwise there are going to be problem regarding mutual exclusion.

```
private void turnFlashlightOn() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        try {
            manager = (CameraManager)
mContext.getSystemService(Context.CAMERA_SERVICE);
            String cameraId = null; // Usually front camera is at 0 position.
            if (manager != null) {
                cameraId = manager.getCameraIdList()[0];
                manager.setTorchMode(cameraId, true);
            }
        } catch (CameraAccessException e) {
            Log.e(TAG, e.toString());
        }
    } else {
        mCamera = Camera.open();
        parameters = mCamera.getParameters();
        parameters.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
        mCamera.setParameters(parameters);
        mCamera.startPreview();
    }
}
```