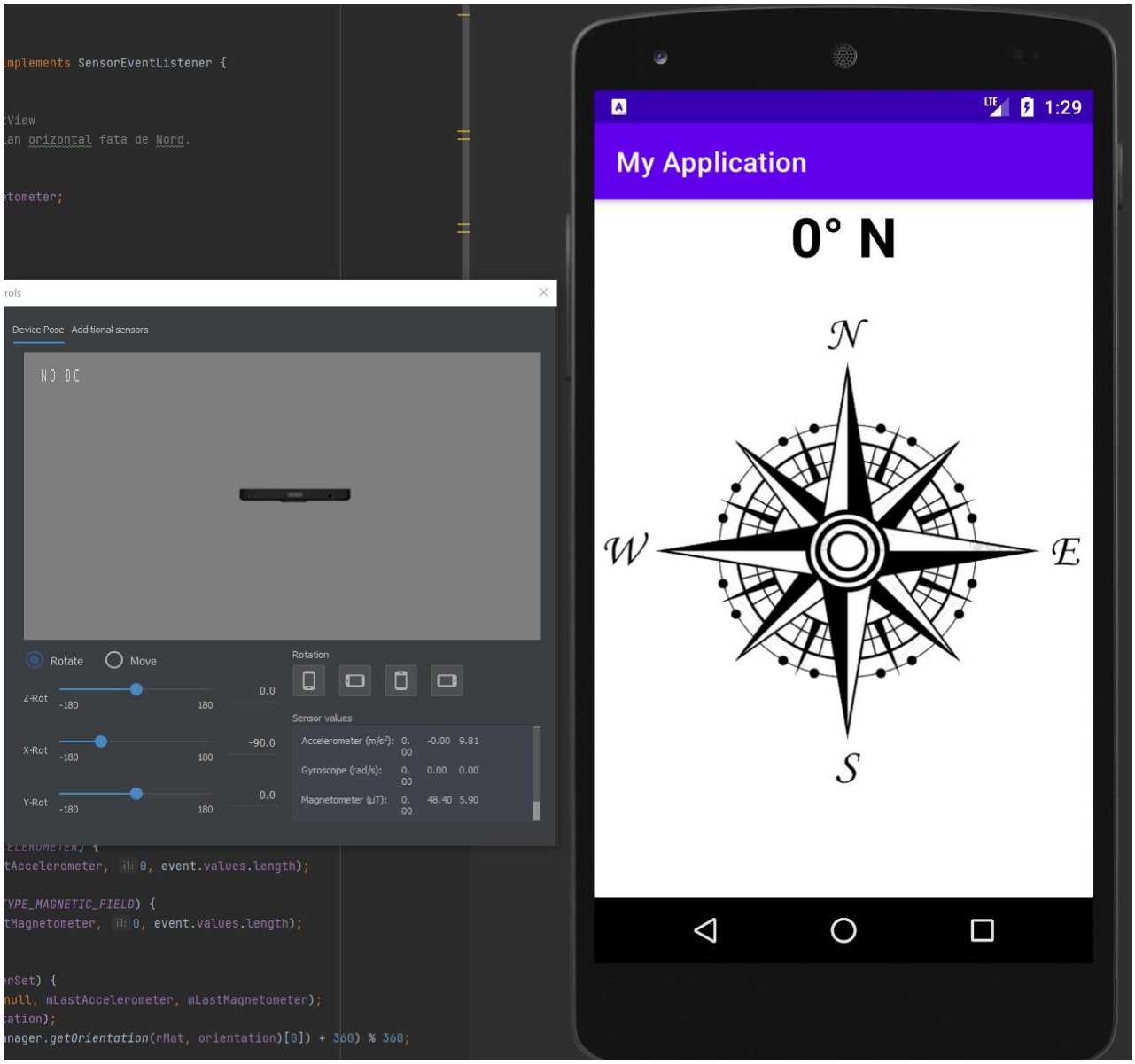
Proiect ADIV

Mirică Victor – 324CB

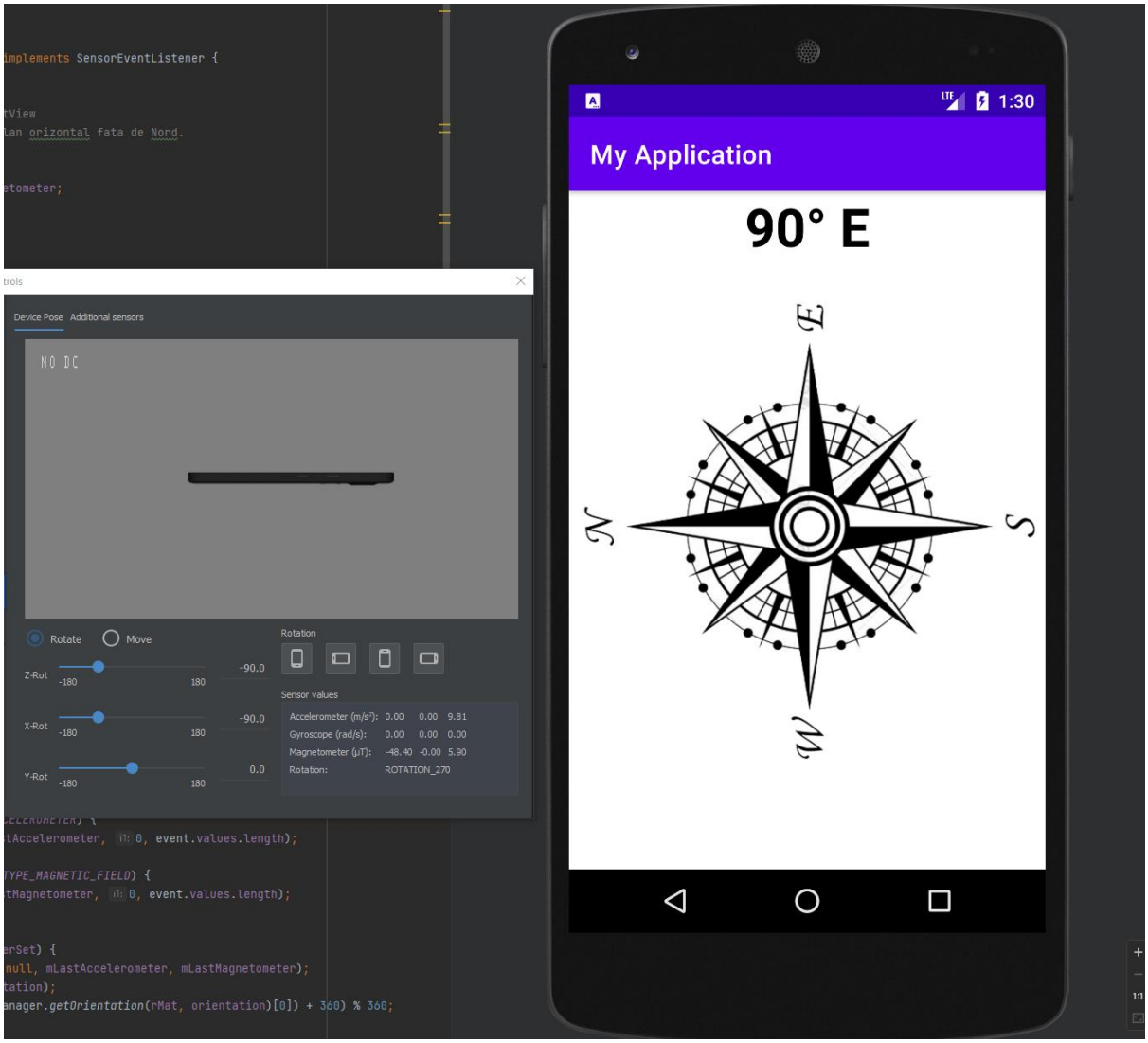
Aplicație Android – Busola

Aplicatia este una simpla, cu o interfata grafica minimalistica, ce foloseste senzorii telefonului pentru a indica directia si azimutul (unghiul in plan fata de nord) in care este orientat acesta.

Prin simularea aplicatiei in Android Studio, se poate observa ca aceasta este implementata corect:



Cand telefonul este orientat spre nord, directia este idicata corect (Se pot observa valorile magnetometrului)



Cand pozitia acestuia pe axa Z este schimbata cu -90, acesta va indica directia est, asa cum ne-am astepta.

Prezentarea Codului, scris in Android Studio.

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txt_azimuth"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="0°"
        android:textAlignment="center"
        android:textColor="#000"
        android:textSize="40sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/img_compass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/compass"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Pentru partea de layout, am creat un camp TextView, pentru directie si azimut, si un ImageView, pentru busola in sine.

```java
public class MainActivity extends AppCompatActivity implements SensorEventListener {

    ImageView compass_img; // Imaginea
    TextView txt_compass; // Valoarea variabilei TextView
    int mAzimuth; // Azimut, valoarea unghiului in plan orizontal fata de Nord.
    // Senzori
    private SensorManager mSensorManager;
    private Sensor mRotationV, mAccelerometer, mMagnetometer;
    boolean haveSensor = false, haveSensor2 = false;

    // Matricile de rotatie si de orientare
    float[] rMat = new float[9];
    float[] orientation = new float[3];
    private float[] mLastAccelerometer = new float[3];
    private float[] mLastMagnetometer = new float[3];
    private boolean mLastAccelerometerSet = false;
    private boolean mLastMagnetometerSet = false;

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

        // Se preia sensor_service, si id-ul pentru imagine si text
        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        compass_img = (ImageView) findViewById(R.id.img_compass);
        txt_compass = (TextView) findViewById(R.id.txt_azimuth);

        start();
    }
```

In clasa Main, am definit mai multe campuri: imageView, textView, valoarea azimut. De asemenea, am creat si campuri pentru diferitii senzori pe care pot sa ii folosesc si un sensor manager. In cazul in care telefonul are un rotation vector sensor il voi folosi pe acela, altfel, voi folosi alti doi senzori: accelerometru si magnetometru.

Urmatoarele campuri sunt folosite pentru matricea de orientare, folosita la calculul azimutului.

```
public void start() {
    // Porneste aplicatia
    // Verificam daca dispozitivul are Rotation vector sensor
    if (mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR) == null) {
        // In cazul in care acesta nu are, vom folosi senzorul geomagnetic si accelerometrul
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        mMagnetometer = mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
        haveSensor = mSensorManager.registerListener( listener: this, mAccelerometer, SensorManager.SENSOR_DELAY_UI);
        haveSensor2 = mSensorManager.registerListener( listener: this, mMagnetometer, SensorManager.SENSOR_DELAY_UI);
    }
    else{
        mRotationV = mSensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
        haveSensor = mSensorManager.registerListener( listener: this, mRotationV, SensorManager.SENSOR_DELAY_UI);
    }
}
```

La inceputul aplicatiei verific daca dispozitivul are rotation vector sensor, pe care il voi folosi. In caz contrar, ma voi folosi de accelerometru si magnetometru.

```
public void stop() {
    // Opreste aplicatia
    if (haveSensor) {
        mSensorManager.unregisterListener( listener: this, mRotationV);
    }
    else {
        mSensorManager.unregisterListener( listener: this, mAccelerometer);
        mSensorManager.unregisterListener( listener: this, mMagnetometer);
    }
}
```

La oprirea aplicatiei, elimin senzorii carora le-am dat register la start.

```java
@Override
public void onSensorChanged(SensorEvent event) {
    // Cand un senzorul simte o schimbare
    if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
        // Preia matricea de rotatie si calculeaza valoarea azimut
        SensorManager.getRotationMatrixFromVector(rMat, event.values);
        mAzimuth = (int) (Math.toDegrees(SensorManager.getOrientation(rMat, orientation)[0]) + 360) % 360;
    }

    // Creeaza matricea de rotatie, pentru a calcula valoarea amzimut.
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        System.arraycopy(event.values, 0, mLastAccelerometer, 0, event.values.length);
        mLastAccelerometerSet = true;
    } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
        System.arraycopy(event.values, 0, mLastMagnetometer, 0, event.values.length);
        mLastMagnetometerSet = true;
    }
    if (mLastAccelerometerSet && mLastMagnetometerSet) {
        SensorManager.getRotationMatrix(rMat, null, mLastAccelerometer, mLastMagnetometer);
        SensorManager.getOrientation(rMat, orientation);
        mAzimuth = (int) (Math.toDegrees(SensorManager.getOrientation(rMat, orientation)[0]) + 360) % 360;
    }

    // Roteste imaginea busolei la "azimuth" grade
    mAzimuth = Math.round(mAzimuth);
    compass_img.setRotation(-mAzimuth);


    // Verificam cum vom seta stringul
    // Pentru directie
    String where = "NW";

    if (mAzimuth >= 350 || mAzimuth <= 10)
        where = "N";
    if (mAzimuth < 350 && mAzimuth > 280)
        where = "NW";
    if (mAzimuth <= 280 && mAzimuth > 260)
        where = "W";
    if (mAzimuth <= 260 && mAzimuth > 190)
        where = "SW";
    if (mAzimuth <= 190 && mAzimuth > 170)
        where = "S";
    if (mAzimuth <= 170 && mAzimuth > 100)
        where = "SE";
    if (mAzimuth <= 100 && mAzimuth > 80)
        where = "E";
    if (mAzimuth <= 80 && mAzimuth > 10)
        where = "NE";

    // Setam stringul
    txt_compass.setText(mAzimuth + "° " + where);
}
```

De fiecare data cand senzorul simte o schimbare, trebuie sa preiau matricea de rotatie si sa calculez valoarea azimut. In cazul in care am rotation vector sensor, matricea se poate prelua direct.

In cazul in care nu folosesc acest senzor, matricea va trebui sa fie preluata in urma valorilor accelerometrului si a magnetometrului.

Dupa ce aceasta este preluata, azimutul se va calcula la fel.

Tot ce mai avem de facut este da updatam interfaca grafica. Pentru busola, vom roti imaginea la valoarea azimutului. Pentru textview, vom afisa azimutul in sine si de asemenea, bazat pe valoare acestuia, si un indicator dintre Nord, Sud, Est, Vest si valorile intermediare.