Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №5

Розробка мобільних застосувань під Android

«ДОСЛІДЖЕННЯ РОБОТИ З ВБУДОВАНИМИ ДАТЧИКАМИ»

Варіант 6

Виконала:
студентка групи ІА-21
Антропова К.Д.

Київ 2025

Мета роботи: ознайомитись з можливостями вбудованих датчиків мобільних пристроїв та дослідити способи їх використання для збору та обробки даних.

Завдання на лабораторну роботу

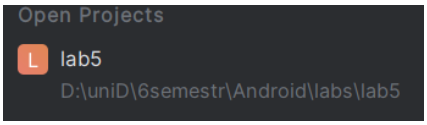
БАЗОВЕ (10/20 балів). Написати програму під платформу Андроїд, яка має інтерфейс для виведення даних з обраного вбудованого датчика (тип обирається самостійно, можна відслідковувати зміни значень і з декількох датчиків).

ПОВНЕ (20/20). Функціональність базового додатку додатково розширюється обробкою отриманих даних та виведенням їх у відповідній формі

Крокомір для тренувань

Хід роботи

1. Створюємо проект



2. В класі MainActivity пишемо такий код:

```
package com.example.lab5

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.example.lab5.R
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
import kotlin.math.abs

class MainActivity : AppCompatActivity(), SensorEventListener {

    private lateinit var sensorManager: SensorManager
    private var stepCounterSensor: Sensor? = null
    private var accelerometerSensor: Sensor? = null
    private lateinit var stepsTextView: TextView
    private lateinit var startButton: Button
    private lateinit var stopButton: Button
    private lateinit var resetButton: Button
    private lateinit var addStepButton: Button
    private lateinit var showStatsButton: Button
    private var initialSteps = 0
```

```
    private var currentSteps = 0
    private var isUsingAccelerometer = false
    private var lastYAcceleration = 0f
    private var accelerationThreshold = 0.1f
    private var lastUpdateTime = 0L
    private val stepDetectionInterval = 300L
    private var isTraining = false
    private var trainingStartTime: Long = 0L
    private lateinit var sharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        stepsTextView = findViewById(R.id.steps_text_view)
        startButton = findViewById(R.id.start_button)
        stopButton = findViewById(R.id.stop_button)
        resetButton = findViewById(R.id.reset_button)
        addStepButton = findViewById(R.id.add_step_button)
        showStatsButton = findViewById(R.id.show_stats_button)

        sharedPreferences = getSharedPreferences("StepCounterPrefs",
Context.MODE_PRIVATE)

        sensorManager = getSystemService(Context.SENSOR_SERVICE) as
SensorManager
        stepCounterSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER)
        accelerometerSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)

        if (stepCounterSensor != null) {
            stepsTextView.text = getString(R.string.steps_count, 0)
            Log.d("StepCounter", "Using STEP_COUNTER sensor")
        } else {
            if (accelerometerSensor != null) {
                isUsingAccelerometer = true
                stepsTextView.text = getString(R.string.steps_count, 0)
                Log.d("StepCounter", "Using ACCELEROMETER sensor")
            } else {
                stepsTextView.text = getString(R.string.sensor_not_found)
                resetButton.isEnabled = false
                startButton.isEnabled = false
                stopButton.isEnabled = false
                Log.d("StepCounter", "No sensors available")
            }
        }

        addStepButton.isEnabled = true

        startButton.setOnClickListener {
            if (!isTraining) {
                isTraining = true
                trainingStartTime = System.currentTimeMillis()
                initialSteps = currentSteps
                startButton.isEnabled = false
                stopButton.isEnabled = true
                resetButton.isEnabled = false
                Log.d("StepCounter", "Training started at
$trainingStartTime")
            }
        }

        stopButton.setOnClickListener {
```

```
            if (isTraining) {
                isTraining = false
                val trainingEndTime = System.currentTimeMillis()
                val duration = (trainingEndTime - trainingStartTime) / 1000
                val stepsDuringTraining = currentSteps - initialSteps
                saveTrainingData(stepsDuringTraining, trainingStartTime,
duration)
                startButton.isEnabled = true
                stopButton.isEnabled = false
                resetButton.isEnabled = true
                Log.d("StepCounter", "Training stopped. Steps:
$stepsDuringTraining, Duration: $duration seconds")
            }
        }

        resetButton.setOnClickListener {
            initialSteps = currentSteps
            updateStepsDisplay()
        }

        addStepButton.setOnClickListener {
            currentSteps++
            updateStepsDisplay()
            Log.d("StepCounter", "Manually added step. Total steps:
$currentSteps")
        }

        showStatsButton.setOnClickListener {
            val intent = Intent(this, StatsActivity::class.java)
            startActivity(intent)
        }

        stopButton.isEnabled = false
    }

    override fun onResume() {
        super.onResume()
        if (isUsingAccelerometer) {
            accelerometerSensor?.let {
                sensorManager.registerListener(this, it,
SensorManager.SENSOR_DELAY_NORMAL)
            }
        } else {
            stepCounterSensor?.let {
                sensorManager.registerListener(this, it,
SensorManager.SENSOR_DELAY_UI)
            }
        }
    }

    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }

    override fun onSensorChanged(event: SensorEvent?) {
        event?.let {
            when (it.sensor.type) {
                Sensor.TYPE_STEP_COUNTER -> {
                    currentSteps = it.values[0].toInt()
                    updateStepsDisplay()
                    Log.d("StepCounter", "Step counter updated:
$currentSteps")
                }
```

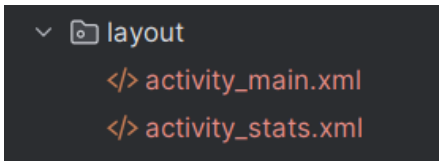
```
                Sensor.TYPE_ACCELEROMETER -> {
                    if (!isTraining) return
                    val currentTime = System.currentTimeMillis()
                    if (currentTime - lastUpdateTime < stepDetectionInterval)
{
                        return
                    }

                    val x = it.values[0]
                    val y = it.values[1]
                    val z = it.values[2]

                    val yAcceleration = y
                    val delta = yAcceleration - lastYAcceleration

                    Log.d("StepCounter", "Accelerometer: x=$x, y=$y, z=$z,
yAcceleration=$yAcceleration, delta=$delta")

                    if (abs(delta) > accelerationThreshold) {
                        currentSteps++
                        lastUpdateTime = currentTime
                        updateStepsDisplay()
                        Log.d("StepCounter", "Step detected via
accelerometer. Total steps: $currentSteps, delta: $delta")
                    }
                    lastYAcceleration = yAcceleration
                }
                else -> {
                    Log.d("StepCounter", "Unsupported sensor type:
${it.sensor.type}")
                }
            }
        }
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {

    }

    private fun updateStepsDisplay() {
        val stepsToShow = if (initialSteps == 0) {
            currentSteps
        } else {
            currentSteps - initialSteps
        }
        stepsTextView.text = getString(R.string.steps_count, stepsToShow)
    }

    private fun saveTrainingData(steps: Int, startTime: Long, duration: Long)
{
        val editor = sharedPreferences.edit()
        val trainingCount = sharedPreferences.getInt("training_count", 0)
        val newTrainingId = trainingCount + 1

        val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
        val startDate = dateFormat.format(Date(startTime))

        editor.putString("training_${newTrainingId}_date", startDate)
        editor.putInt("training_${newTrainingId}_steps", steps)
        editor.putLong("training_${newTrainingId}_duration", duration)
        editor.putInt("training_count", newTrainingId)
        editor.apply()
```

```
    }
}
```

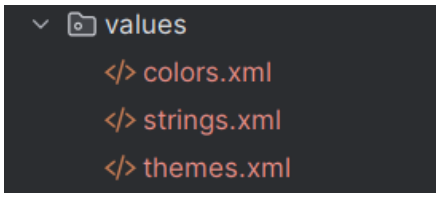
3. В класі StatsActivity пишемо такий код:

```
package com.example.lab5

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.example.lab5.R

class StatsActivity : AppCompatActivity() {

    private lateinit var statsTextView: TextView
    private lateinit var sharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_stats)

        statsTextView = findViewById(R.id.stats_text_view)
        sharedPreferences = getSharedPreferences("StepCounterPrefs",
Context.MODE_PRIVATE)

        displayStats()
    }

    private fun displayStats() {
        val trainingCount = sharedPreferences.getInt("training_count", 0)
        if (trainingCount == 0) {
            statsTextView.text = getString(R.string.no_trainings)
            return
        }

        val statsBuilder = StringBuilder()
        for (i in 1..trainingCount) {
            val date = sharedPreferences.getString("training_${i}_date",
"N/A") ?: "N/A"
            val steps = sharedPreferences.getInt("training_${i}_steps", 0)
            val duration =
sharedPreferences.getLong("training_${i}_duration", 0)
            val stepsPerMinute = if (duration > 0) (steps * 60 /
duration).toFloat() else 0f

            statsBuilder.append("Тренування $i\n")
            statsBuilder.append("Дата: $date\n")
            statsBuilder.append("Кількість кроків: $steps\n")
            statsBuilder.append("Тривалість: $duration секунд\n")
            statsBuilder.append("Швидкість: %.2f
кроків/хв\n\n".format(stepsPerMinute))
        }

        statsTextView.text = statsBuilder.toString()
    }
}
```

4. Додаємо layout файли з потрібним кодом:



5. Додаємо необхідний код в файлах в директорії values:



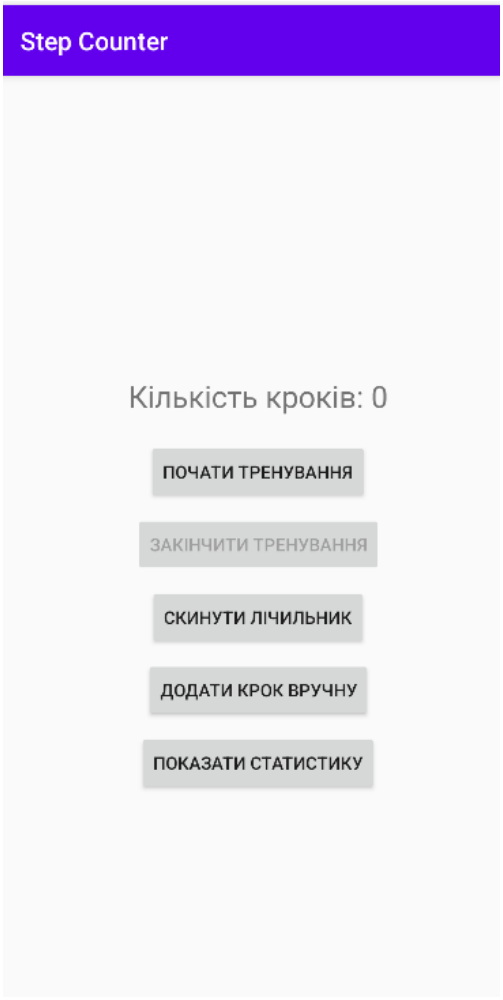
6. В AndroidManifest.xml додаємо:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.lab5">

    <uses-permission android:name="android.permission.ACTIVITY_RECOGNITION"
/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.StepCounter">
        <activity
            android:name=".StatsActivity"
            android:exported="false"/>
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

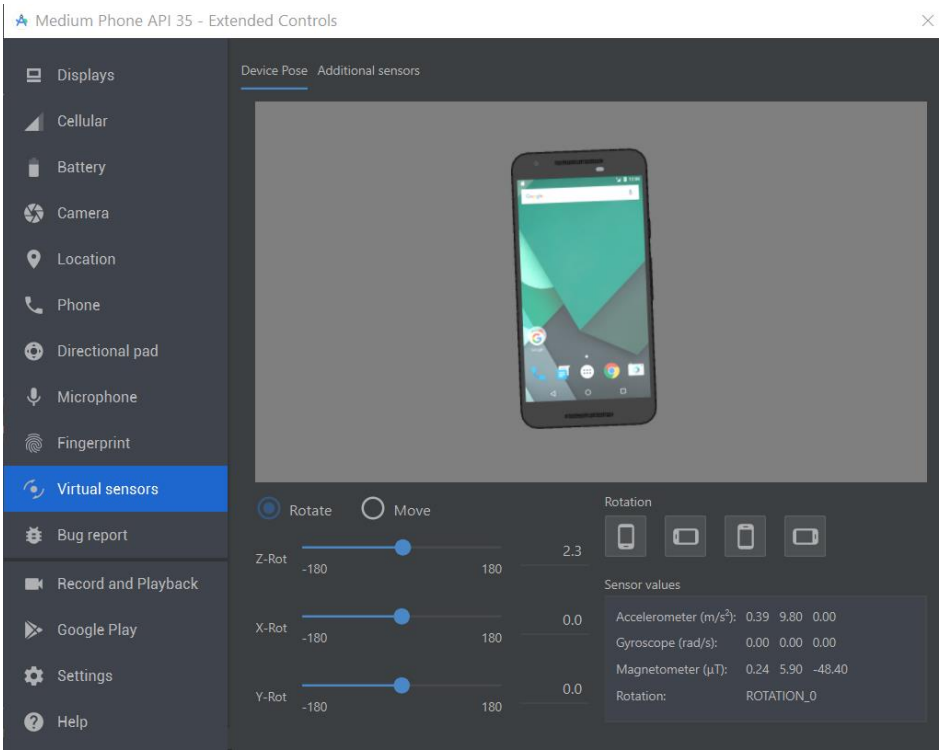
Також вносимо необхідні зміни в build.gradle.kts.

Результат:

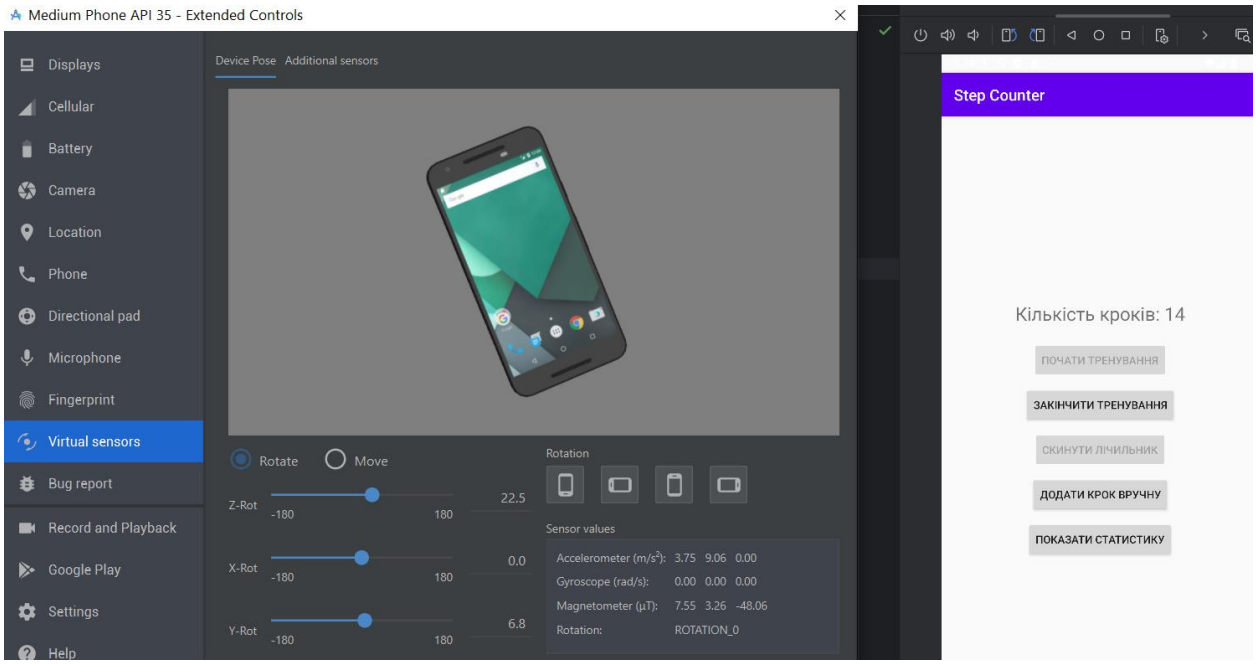
Заходимо в програму і бачимо компоненти програми, а саме відображення кількості кроків та усі потрібні кнопки (почати та закінчити тренування, скинути лічильник, додати крок вручну та перегляд статистики):



Для перевірки того, чи дійсно наша програма рахує кроки, треба відкрити Extended Controls девайсу та регулювати там вісь Z:



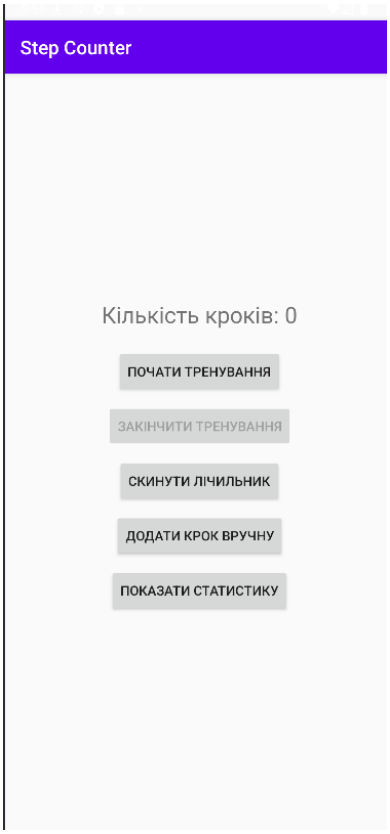
Ми натискаємо кнопку «Почати тренування» і регулюємо значення осі Z та бачимо як додаються кроки:



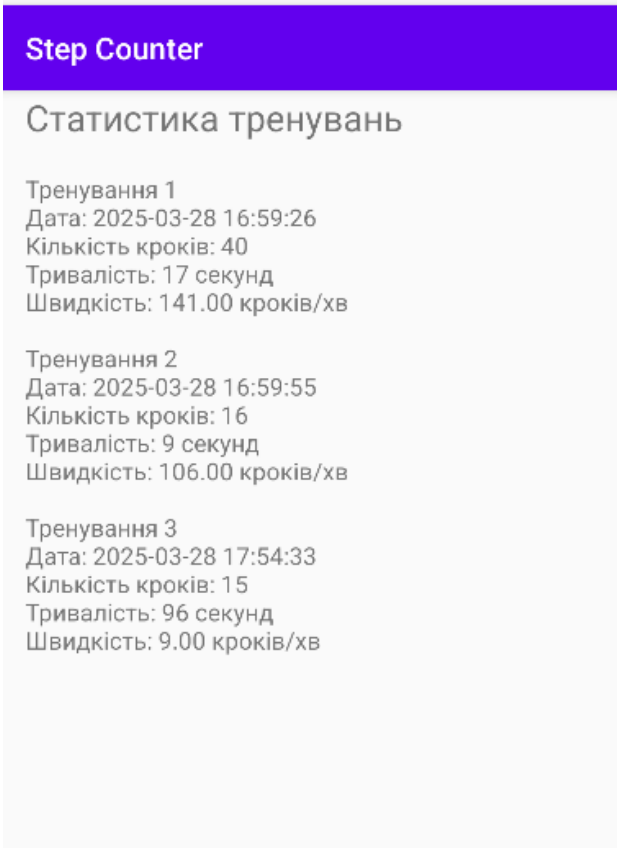
Також можна додати крок самостійно шляхом натискання клавіши «Додати крок вручну»:

(було 14, а стало 15)

При натисненні клавіши «Завершити тренування» знову стають активними кнопки початку тренування та скидання лічильника. При натисненні «Скинути лічильник» значення на лічильнику стає 0:



При натисненні кнопки «Показати статистику» відкривається статистика тренувань:



(Тренування 3 – тренування в звіті)

Отже, програма працює відповідно до вимог завдання.

Висновок : У ході виконання цієї лабораторної роботи я ознайомилась з можливостями вбудованих датчиків мобільних пристроїв та дослідила способи їх використання для збору та обробки даних, а також створила застосунок відповідно до вимог та отримала відповідні практичні навички.