Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Розробка мобільних застосувань під Android**

**Лабораторна робота №5**

Виконав:

студент групи ІО-23

Корбут М. Я.

Залікова книжка №2313

Перевірив

Орленко С. П.

Київ - 2025

**Лабораторна робота №5**

**Опис програми**

Для цієї лабораторної я вирішив створити програму, що дозволить користувачу визначати кут підйому/спуску при встановленні пристрою в автомобілі. Програма реалізована на платформі Android. Вона використовує вбудований сенсор Rotation Vector для визначення кута нахилу пристрою (автомобіля). Отримані дані проходять згладжування та коригуються з урахуванням калібрування. Результат виводиться у вигляді числового значення та візуального індикатора — стрілки, що обертається. Колір тексту змінюється залежно від рівня нахилу для зручного візуального контролю.

**Основні компоненти додатка**

* Мова програмування: Java
* IDE: Android Studio
* Android SDK
* Компоненти UI: TextView, Button, ImageView
* Сенсори: Sensor.TYPE\_ROTATION\_VECTOR
* Збереження даних: SharedPreferences.

**Сенсори в Android**

Android-пристрої підтримують різноманітні сенсори для збору даних про положення, рух і навколишнє середовище. Для доступу до них використовується SensorManager, а для обробки даних — інтерфейс SensorEventListener.

У цій програмі використовується сенсор Sensor.TYPE\_ROTATION\_VECTOR. Це віртуальний сенсор, який поєднує дані з гіроскопа, акселерометра та магнітометра для визначення орієнтації пристрою в просторі. На відміну від сирих сенсорів, TYPE\_ROTATION\_VECTOR дає більш стабільні й фільтровані значення, що дозволяє точно визначати кут нахилу.

Дані з сенсора перетворюються в матрицю обертання за допомогою SensorManager.getRotationMatrixFromVector(...), а потім у кути орієнтації (азимут, нахил, крен) через SensorManager.getOrientation(...). У цьому проєкті використовується саме кут pitch (нахил уперед/назад), який потім обробляється і виводиться на екран.

**Ключові фрагменти коду**

*Ініціалізація сенсора та інтерфейсу*

rotationVectorSensor = sensorManager.getDefaultSensor(Sensor.TYPE\_ROTATION\_VECTOR);

rawTiltText = findViewById(R.id.rawTiltText);

arrowImage = findViewById(R.id.arrowImage);

Цей фрагмент відповідає за підключення до сенсора обертання та ініціалізацію елементів інтерфейсу.

*Обробка даних сенсора*

SensorManager.getRotationMatrixFromVector(rotationMatrix, event.values);

SensorManager.getOrientation(rotationMatrix, orientationAngles);

float rawTilt = (float) Math.toDegrees(orientationAngles[1]);

Обчислення кута нахилу у градусах на основі вектору обертання.

*Згладжування та калібрування значень*

smoothedTilt = alpha \* rawTilt + (1 - alpha) \* smoothedTilt;

float correctedTilt = -(smoothedTilt - calibrationOffset);

Застосовується експоненціальне згладжування та корекція з урахуванням калібрування.

*Збереження калібрування*

calibrationOffset = smoothedTilt;

SharedPreferences.Editor editor = prefs.edit();

editor.putFloat("calibrationOffset", calibrationOffset);

editor.apply();

Після натискання кнопки калібрування зберігається поточний кут як базовий.

**Фото додатку з реального пристрою**

Рис. 2. Збільшений кут нахилу

Рис. 1. Нейтральний кут

**Репозиторій**

Код було завантажено до репозиторію GitHub. Переглянути його можна за [посиланням](https://github.com/Misha1tigr/android-labs-korbut/tree/main/Lab5).

**Висновки**

Програма повністю виконує вимоги завдання: вона отримує дані від сенсора, обробляє їх (згладжує, коригує), надає зручний вивід даних у числовому та графічному вигляді. Наявність кнопки калібрування дозволяє налаштувати «нульовий» рівень, а кольорове підсвічування робить інтерфейс інтуїтивно зрозумілим для користувача. Програму було встановлено та протестовано на реальному пристрої.