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

Лабораторна робота №5
із дисципліни «Розробка мобільних застосувань під Android»
Тема: «Дослідження роботи з вбудованими датчиками

Виконав:
Студент групи ІА-24
Боднар А.Д.

Перевірив:
Орленко С. П.

Київ-2025

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

Хід роботи

Я обрав тему:
автоматичне регулювання яскравості та екрану в залежності від рівня освітлення, але ще б додати автозаглушення екрану при піднесенні до перешкоди (до вуха під час розмови або «в кишені»), щоб уникнути ненавмисних дотиків;

Код

```
private lateinit var sensorManager: SensorManager
private var lightSensor: Sensor? = null
private var proximitySensor: Sensor? = null

private lateinit var lightTextView: TextView

private var wakeLock: PowerManager.WakeLock? = null
private lateinit var powerManager: PowerManager
```

- Ініціалізація менеджера сенсорів і посилань на світловий і датчик наближення.
- lightTextView — показує освітленість.
- wakeLock і powerManager — керують вимкненням екрана при наближенні.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Дозвіл на зміну яскравості
    if (!Settings.System.canWrite(this)) {
        val intent =
Intent(Settings.ACTION_MANAGE_WRITE_SETTINGS)
        intent.data =
android.net.Uri.parse("package:$packageName")
        startActivity(intent)
    }

    setContentView(R.layout.activity_main)

    lightTextView = findViewById(R.id.light_level)

    // Ініціалізація сенсорів
    sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
    lightSensor =
```

```
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
    proximitySensor =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)

    powerManager = getSystemService(Context.POWER_SERVICE) as
PowerManager
}
```

- Перевірка і запит дозволу на зміну системної яскравості.
- Прив’язка елементів інтерфейсу та ініціалізація сенсорів і PowerManager для подальшої роботи з WakeLock.

```
override fun onResume() {
    super.onResume()
    lightSensor?.also {
        sensorManager.registerListener(this, it,
SensorManager.SENSOR_DELAY_NORMAL)
    }
    proximitySensor?.also {
        sensorManager.registerListener(this, it,
SensorManager.SENSOR_DELAY_NORMAL)
    }
}
```

- Реєстрація слухачів сенсорів, щоб почати отримувати значення при активному екрані.

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

- Відключаємо сенсори, щоби не витрачати ресурси.
- Знімаємо WakeLock (якщо був увімкнений), щоб не лишати екран вимкненим.

```
override fun onSensorChanged(event: SensorEvent?) {
    if (event == null) return

    when (event.sensor.type) {
        Sensor.TYPE_LIGHT -> {
            val lux = event.values[0]
            lightTextView.text = "Освітленість: $lux лк"

            // Автояскравість
```

```
            val brightness = (lux / 1000).coerceIn(0.1f, 1f)
            try {
                Settings.System.putInt(
                    contentResolver,
                    Settings.System.SCREEN_BRIGHTNESS_MODE,

Settings.System.SCREEN_BRIGHTNESS_MODE_MANUAL
                )
                Settings.System.putInt(
                    contentResolver,
                    Settings.System.SCREEN_BRIGHTNESS,
                    (brightness * 2000).toInt()
                )
            } catch (e: Exception) {
                Toast.makeText(this, "Немає дозволу на зміну
яскравості", Toast.LENGTH_SHORT).show()
            }
        }

        Sensor.TYPE_PROXIMITY -> {
            val distance = event.values[0]
            Log.d("PROXIMITY", "Відстань: $distance")

            if (distance < proximitySensor?.maximumRange ?: 0f)
{
                acquireProximityWakeLock()
            } else {
                releaseProximityWakeLock()
            }
        }
    }
}
```

- **Light Sensor**: Отримуємо освітленість і перетворюємо в яскравість екрана (від 0.1 до 1), оновлюємо системну яскравість.
- **Proximity Sensor**: Якщо перешкода близько — вмикаємо WakeLock, щоб вимкнути екран. Якщо перешкоди немає — вимикаємо WakeLock, екран вмикається.

```
<TextView
    android:id="@+id/light_level"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Освітленість: --- лк"
    android:textSize="24sp"
    android:textColor="#000000" />
```

Це єдиний елемент, що показує **живий рівень освітленості** в люксах, який постійно оновлюється сенсором.

```
private fun acquireProximityWakeLock() {
    if (wakeLock == null || wakeLock?.isHeld == false) {
        wakeLock = powerManager.newWakeLock(
            PowerManager.PROXIMITY_SCREEN_OFF_WAKE_LOCK,
            "AutoBrightness:ProximityLock"
        )
        wakeLock?.acquire()
        Log.d("WAKELOCK", "WakeLock активовано")
    }
}
```

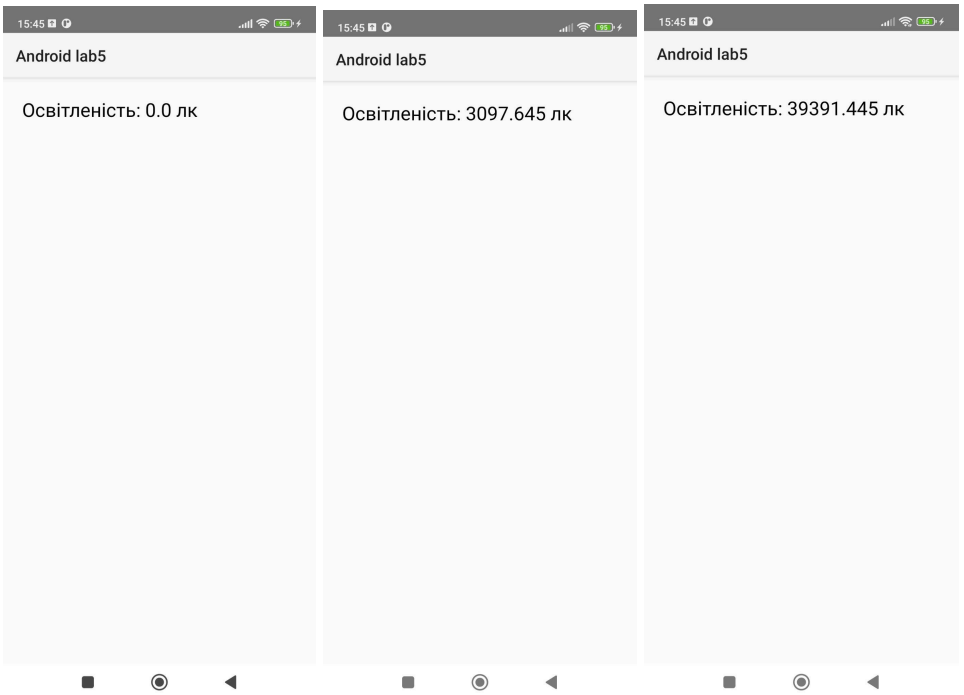
- Створює і активує WakeLock, щоб вимкнути екран, коли датчик виявляє перешкоду.

```
private fun releaseProximityWakeLock() {
    if (wakeLock?.isHeld == true) {
        wakeLock?.release()
        Log.d("WAKELOCK", "WakeLock відпущено")
    }
}
```

- Вимикає WakeLock, щойно перешкода зникає — **екран вмикається назад**.

Результати

При тестуванні використовувався власний телефон Xiaomi Redmi Note 11



Результати автояскравості та автозаглушення екрану є в відео котре йде разом з цим звітом; так як у такому звіті це показати дуже важко, якщо можливо