


1Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki al. prof. S. Kaliskiego 7, 85-796 Bydgoszcz				 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Przedmiot	Programowanie urządzeń mobilnych			Kierunek/Tryb	IS/ST
Nr laboratorium	9	Data wykonania	17.06	Grupa	1
Ocena		Data oddania	17.06	Imię Nazwisko	Cezary Tytko
Nazwa ćwiczenia	Obsługa sensorów, generowanie wykresów, tworzenie statystyk				

2. Cel ćwiczenia laboratoryjnego

Celem tego ćwiczenia jest stworzenie aplikacji do zbierania danych z wybranych sensorów urządzenia mobilnego. Aplikacja będzie rejestrowała dane z wybranych sensorów urządzenia i zgodnie z określonymi wymaganiami użytkownika tworzyła wykresy, zestawienia oraz reagowała na przekroczenie zadanych wartości. Uczestnik zajęć będzie miał okazję zaznajomić się z różnymi aspektami programowania aplikacji mobilnych.

Studenci zostaną poproszeni o utworzenie aplikacji, którą będzie umożliwiała zbieranie danych z wybranego sensora urządzenia. Aplikacja powinna posiadać ekran konfiguracyjny, gdzie użytkownik będzie miał możliwość wyboru sensora, wyboru interwału czasu pomiędzy kolejnymi odczytami wartości z sensora, wyboru czasu odświeżania wykresu i ustalenia wartości danych pomiarowych po przekroczeniu której będzie generowany alarm.

3. Opis projektu

Dostępne są sensory – acceleracja, reakcja na światło, których wyniki przedstawione na wykresach, z możliwością zapisania ich do pliku .csv na urządzeniu.

4. Implementacja

Kod został napisany w języku Kotlin w środowisku Android Studio.

MainActivity.kt:

```

1. package com.example.pumlab1
2.
3. import android.hardware.Sensor
4. import android.hardware.SensorEvent
5. import android.hardware.SensorEventListener
6. import android.hardware.SensorManager
7. import android.media.MediaPlayer
8. import android.os.Bundle
9. import android.os.Handler
10. import android.os.Looper
11. import android.view.ViewGroup
12. import android.widget.Button
13. import android.widget.LinearLayout
14. import android.widget.Toast
15. import androidx.activity.enableEdgeToEdge
16. import androidx.appcompat.app.AppCompatActivity
17. import androidx.core.view.ViewCompat
18. import androidx.core.view.WindowInsetsCompat
19. import java.io.File
20. import com.github.mikephil.charting.charts.LineChart
21. import com.github.mikephil.charting.data.Entry
22. import com.github.mikephil.charting.data.LineData
23. import com.github.mikephil.charting.data.LineDataSet
24. import kotlin.random.Random

```

```

25.
26. class MainActivity : AppCompatActivity(), SensorEventListener {
27.     private lateinit var sensorManager: SensorManager
28.     private val selectedSensors = mutableListOf<Sensor>()
29.     private val sensorData = mutableMapOf<Int, MutableList<Entry>>()
30.     private val charts = mutableMapOf<Int, LineChart>()
31.     private var threshold: Float = 0f
32.     private var interval: Long = 1000
33.     private val lastUpdateTimes = mutableMapOf<Int, Long>()
34.     private var startTime: Long = 0;
35.
36.
37.     private val simulatedSensorType = 9999 // sztuczny typ sensora dla symulacji
38.     private val simulatedEntries = mutableListOf<Entry>()
39.     private val random = Random(System.currentTimeMillis())
40.     private val handler = Handler(Looper.getMainLooper())
41.     private var simulatedXValue = 0f
42.
43.
44.     override fun onCreate(savedInstanceState: Bundle?) {
45.         super.onCreate(savedInstanceState)
46.         setContentView(R.layout.activity_main)
47.
48.         sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
49.
50.         // Przykład: wybór sensorów (zastąp rzeczywistym UI)
51.         val sensorList = listOfNotNull(
52.             sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
53.             sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
54.             sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)
55.         )
56.         selectedSensors.addAll(sensorList)
57.
58.         threshold = 10f // przykładowa wartość
59.         interval = 1000 // 1 sekunda
60.
61.         for (sensor in selectedSensors) {
62.             sensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL)
63.             sensorData[sensor.type] = mutableListOf()
64.             setupChart(sensor.type)
65.             lastUpdateTimes[sensor.type] = 0
66.         }
67.
68.
69.         // sensorData[simulatedSensorType] = simulatedEntries
70.         // setupChart(simulatedSensorType)
71.         //
72.         // startSimulatedSensorData()
73.
74.         // Zapis danych przy zamknięciu aplikacji (np. przycisk w UI)
75.         findViewById<Button>(R.id.saveButton)?.setOnClickListener {
76.             saveDataToFile()
77.         }
78.         startTime = System.currentTimeMillis()
79.     }
80.
81.     private fun setupChart(sensorType: Int) {
82.         val chart = LineChart(this)
83.         chart.layoutParams = LinearLayout.LayoutParams(
84.             ViewGroup.LayoutParams.MATCH_PARENT, 600
85.         )
86.         findViewById<LinearLayout>(R.id.chartContainer).addView(chart)
87.         charts[sensorType] = chart
88.     }
89.
90.     override fun onSensorChanged(event: SensorEvent?) {
91.         event ?: return
92.         val value = event.values[0]
93.         val sensorType = event.sensor.type
94.         val currentTime = System.currentTimeMillis()
95.
96.         val lastUpdateTime = lastUpdateTimes[sensorType] ?: 0
97.         if (currentTime - lastUpdateTime < interval) return
98.         lastUpdateTimes[sensorType] = currentTime
99.
100.         val timeSeconds = (currentTime - startTime) / 1000f
101.         val entries = sensorData[sensorType] ?: return
102.         entries.add(Entry(timeSeconds, value))

```

```

103.         updateChart(sensorType, entries)
104.
105. //         if (value > threshold) {
106. //             Toast.makeText(this, "Alarm: ${event.sensor.name} > $threshold", Toast.LENGTH_SHORT).show()
107. //             // Dźwiękowy alarm
108. //             MediaPlayer.create(this, R.raw.alarm)?.start()
109. //         }
110.     }
111.
112.     private fun updateChart(sensorType: Int, entries: List<Entry>) {
113.         val chart = charts[sensorType] ?: return
114.         val dataSet = LineDataSet(entries, "Sensor $sensorType")
115.         chart.data = LineData(dataSet)
116.         chart.invalidate()
117.     }
118.
119.     override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
120.
121.     override fun onDestroy() {
122.         super.onDestroy()
123.         sensorManager.unregisterListener(this)
124.     }
125.
126.     // Dodaj zapis do pliku
127.     private fun saveDataToFile() {
128.         val file = File(getExternalFilesDir(null), "sensor_data.csv")
129.         file.printWriter().use { out ->
130.             sensorData.forEach { (type, entries) ->
131.                 entries.forEach { entry ->
132.                     out.println("$type,${entry.x},${entry.y}")
133.                 }
134.             }
135.         }
136.         Toast.makeText(this, "Dane zapisane do pliku", Toast.LENGTH_SHORT).show()
137.     }
138.
139.     private fun startSimulatedSensorData() {
140.         handler.postDelayed(object : Runnable {
141.             override fun run() {
142.                 val value = random.nextInt(201) - 100 // losowe od -100 do 100
143.                 val timeSeconds = simulatedXValue
144.                 simulatedEntries.add(Entry(timeSeconds, value.toFloat()))
145.                 updateChart(simulatedSensorType, simulatedEntries)
146.                 simulatedXValue += 1f
147.                 handler.postDelayed(this, interval)
148.             }
149.         }, interval)
150.     }
151. }
152.
153.

```

Activity_main.xml:

```

1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     android:id="@+id/rootLayout"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical"
6.     android:padding="16dp">
7.
8.     <Button
9.         android:id="@+id/saveButton"
10.        android:layout_width="wrap_content"
11.        android:layout_height="wrap_content"
12.        android:text="Zapisz dane"/>
13.
14.     <ScrollView
15.         android:layout_width="match_parent"
16.         android:layout_height="0dp"
17.         android:layout_weight="1"
18.         android:fillViewport="true">
19.
20.         <LinearLayout
21.             android:id="@+id/chartContainer"
22.             android:layout_width="match_parent"

```

```

23.         android:layout_height="wrap_content"
24.         android:orientation="vertical" />
25.
26.     </ScrollView>
27.
28. </LinearLayout>
29.

```

5. Funkcje kluczowe

1. Odczyt i monitorowanie danych z czujników (akcelerometr, światło, zbliżeniowy) w czasie rzeczywistym.
2. Wyświetlanie danych sensora na wykresach liniowych aktualizowanych na bieżąco.
3. Zapis zebranych danych do pliku CSV po naciśnięciu przycisku.

6. Testowanie


Test działania sensorów:

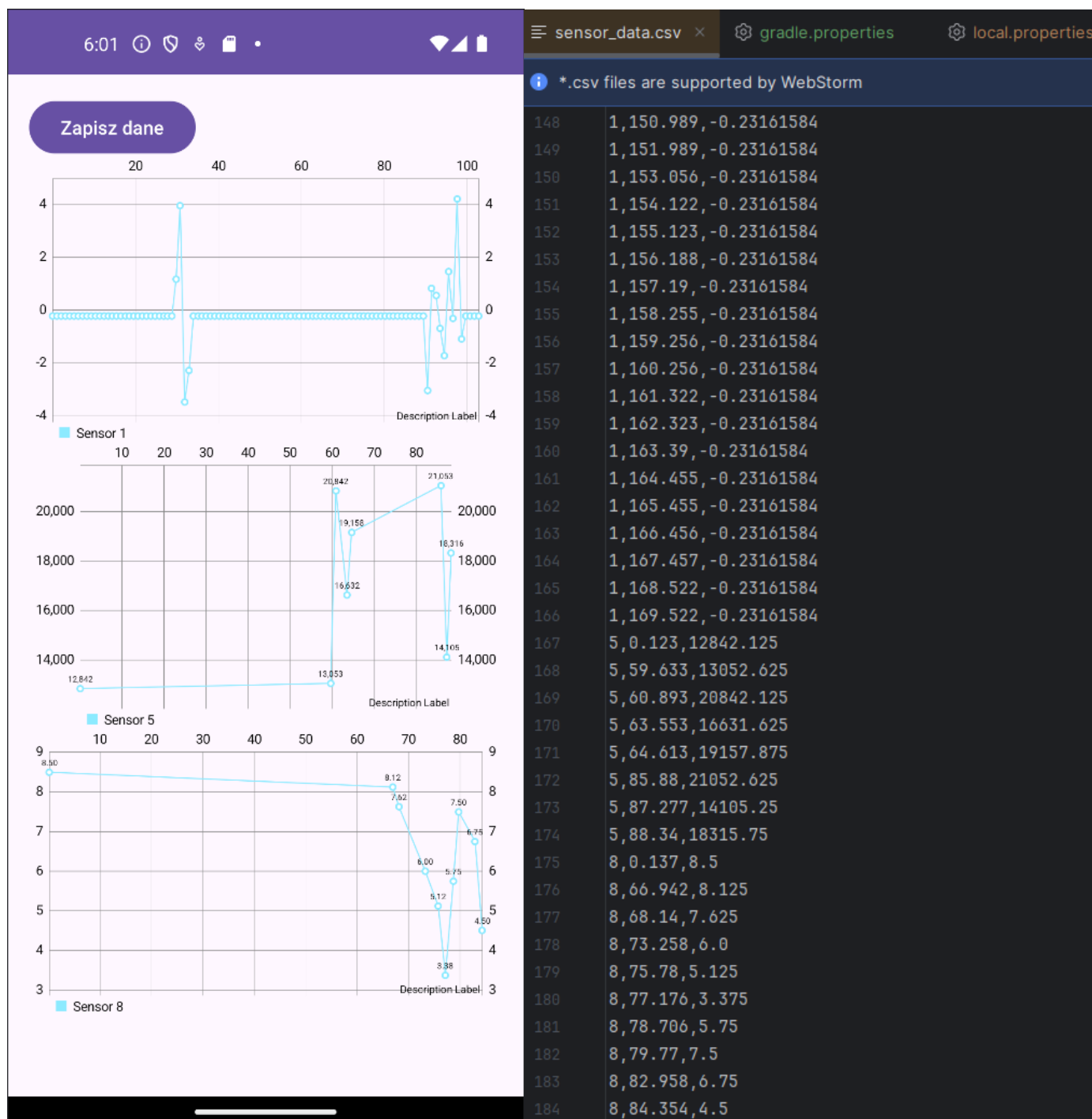
- Uruchomienie aplikacji na fizycznym urządzeniu z Androidem.
- Sprawdzenie, czy wykresy reagują na ruch (akcelerometr), zmiany światła oraz zbliżanie się do czujnika zbliżeniowego.
- Weryfikacja aktualizacji danych na wykresie zgodnie z ustalonym interwałem (np. co sekundę).

Test zapisu danych do pliku:

- Zbieranie danych przez określony czas oraz wykonanie zapisu poprzez naciśnięcie przycisku.
- Potwierdzenie utworzenia pliku sensor_data.csv w katalogu aplikacji.
- Sprawdzenie poprawności zawartości pliku (typ sensora, czas, wartość).

7. Wyniki

Pixel 5 - API 34 Android 14.0 ("UpsideDownCake")			
Files Processes			
			
Name	Permissions	Date	Size
> proc	dr-xr-xr-x	2025-03-21 17:04	0 B
> product	drwxr-xr-x	2009-01-01 00:00	4 KB
> sdcard	lrw-r--r--	2009-01-01 00:00	21 B
> second_stage_resources	drwxr-xr-x	2009-01-01 00:00	4 KB
▼ storage	drwx--x---	2025-03-21 17:05	100 B
> 18F0-3C0E	drwxrwx---	1970-01-01 00:00	2 KB
▼ emulated	drwxrwx---	2025-03-21 17:05	4 KB
▼ 0	drwxrws---	2025-03-21 17:05	4 KB
> Alarms	drwxrws---	2025-03-21 17:05	4 KB
▼ Android	drwxrws--x	2025-03-21 17:05	4 KB
▼ data	drwxrws--x	2025-05-26 18:03	4 KB
> com.android.chrome	drwxrws---	2025-03-30 12:28	4 KB
> com.android.vending	drwxrws---	2025-03-30 12:35	4 KB
▼ com.example.pumlabs_2	drwxrws---	2025-05-26 17:40	4 KB
▼ files	drwxrws---	2025-05-26 17:40	4 KB
≡ sensor_data.csv	-rw-rw----	2025-05-26 18:02	3,7 KB
> com.google.android.apps.docs	drwxrws---	2025-05-26 18:03	4 KB
> com.google.android.apps.mess	drwxrws---	2025-03-30 12:29	4 KB
> com.google.android.apps.youtu	drwxrws---	2025-04-06 19:08	4 KB
> com.google.android.gms	drwxrws---	2025-03-21 17:07	4 KB
> com.google.android.googlequik	drwxrws---	2025-03-21 17:07	4 KB
> com.google.android.youtube	drwxrws---	2025-03-21 17:05	4 KB
> media	drwxrws--x	2025-03-21 17:05	4 KB
> obb	drwxrws--x	2025-03-21 17:06	4 KB
> Audinbooks	drwxrws---	2025-03-21 17:05	4 KB



8. Podsumowanie

Projekt realizuje przedstawienie na wykresów reakcji wbudowanych czujników urządzenia – acceleracja, reakcja na światło oraz wykres testowy i umożliwia zapisywanie wyników do pliku csv.

9. Trudności i błędy

- Nie wystąpiły żadne trudności ani błędy.

10. Źródła i odniesienia

- Nie korzystano ze źródeł i odniesień innych niż ta instrukcja.

11. Dodatkowe materiały

- Nie korzystano z dodatkowych materiałów.