Piotr KOPNIAK
INSTYTUT INFORMATYKI, WEII, POLITECHNIKA LUBELSKA,
ul. Nadbystrzycka 36B, 20-618 Lublin

Interfejsy programistyczne akcelerometrów dla urządzeń mobilnych typu Smartphone

Dr inż. Piotr KOPNIAK

Dr inż. Piotr Kopniak jest adiunktem w Zakładzie Ochrony Informacji Instytutu Informatyki na Wydziale Elektrotechniki i Informatyki Politechniki Lubelskiej. Od 2004 do 2007 roku zajmował się tematyką ochrony informacji, a w szczególności ukrywaniem informacji w obrazie cyfrowym. Obecnie w pracy naukowej zajmuje się badaniami związanymi z przetwarzaniem obrazu i przechwytywaniem ruchu (ang. Motion Capure). Prowadzi zajęcia m.in. z programowania w języku Java oraz programowania urządzeń mobilnych.

e-mail: copy@pluton.pol.lublin.pl

Streszczenie

Celem artykułu jest porównanie cech API dla najpopularniejszych platform programistyczno-systemowych zaawansowanych telefonów komórkowych, tzn. platformy Java Micro Edition, Android oraz nowego systemu Windows Phone 7. W części praktycznej przedstawiono parametry i wyniki pomiarów uzyskanych poprzez opisywane API akcelerometrów wbudowanych w telefony Nokia 5800 XpressMusic i LG GT540 Swift. Jak wykazano największe możliwości posiada obecnie platforma Java ME jednak szybko rozwijające się systemy Android i Windows Phone 7 mogą tą sytuację szybko zmienić ponieważ posiadają unikalne funkcje wykorzystujące kilka czujników jednocześnie.

Słowa kluczowe: telefon komórkowy, akcelerometr, programowanie.

Programming interfaces of accelerometers for Smartphone type mobile devices

Abstract

The aim of this paper is a comparison of API capabilities for the most popular system-programming platforms of cellular phones (Smartphones), i.e. Java Micro Edition platform, Android system and new Windows Phone 7 system. Section 2 contains an introduction to terms of an accelerometer and phone axis of acceleration measurement. Sections 3, 4 and 5 are descriptions of programming library members for sensor services of Java ME, Android and Windows Phone 7 platforms. Parameters of internal accelerometers for Nokia 5800 XpressMusic and LG GT540 Swift phones are given in Section 6. There are presented the results of measuring the Earth acceleration along all three phone axis. Information of accelerometers and acceleration values were read thanks to utilisation of API functions described before. They are Mobile Sensor API of Java ME for Nokia and Android system API in case of LG phone. As it is shown in Tab. 1 the measurement accuracy as well as the values of acceleration measured along particular axes are different for the tested devices. The maximum error reaches 7%. As it is shown all programming interfaces allow reading data asynchronously. Java ME platform has got the largest range of functions and supported kinds of sensors but fast developing Android and Windows Phone 7 systems may change this situation quickly. They have got some unique abilities, e.g. functions which use data from multiple sensors during attitude measurement.

Keywords: mobile phone, accelerometer, programming.

1. Wstęp

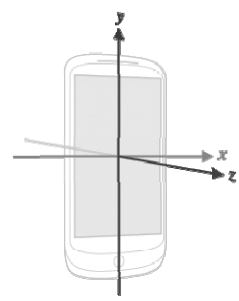
Ostatnia dekada przyniosła gwałtowny rozwój rynku elektronicznych urządzeń mobilnych. Telefon komórkowy przestał być jedynie urządzeniem do przesyłania drogą radiową cyfrowo rejestrowanego dźwięku, czy krótkich wiadomości tekstowych. Dzisiaj zaawansowany telefon jest miniaturowym, przenośnym komputerem osobistym o mocy obliczeniowej porównywalnej z komputerami biurkowymi sprzed dziesięciu lat. Dzięki wyposażaniu telefonów w czujniki przez producentów mamy możliwość wykorzystania ich jako przenośnych urządzeń pomiarowych. Zaawansowane telefony wyposażane są obecnie w różne rodzaje czujników, z których najczęściej występującym rodzajem są akcelerometry.

Producenci urządzeń mobilnych udostępniają interfejsy programistyczne (ang. Application Programming Interface - API) dla różnych języków programowania, np. Javy, C# czy C++, umożliwiające tworzenie oprogramowania współpracującego z wbudowanymi w urządzenia czujnikami. Są to biblioteki, które różnią się nie tylko przeznaczeniem dla konkretnego języka programowania, czy platformy systemowej, ale także posiadające różną funkcjonalność. Ze względu na istnienie kilku wiodących platform mobilnych [1] istotne okazało się zbadanie funkcjonalności i porównanie cech interfejsów programistycznych dla nich przygotowanych. Wyniki przeprowadzonych badań, zamieszczone w niniejszej pracy, mogą ułatwić wybór platformy dla konkretnego zastosowania pomiarowego oraz ułatwić przenoszenie oprogramowania korzystającego z czujników pomiędzy platformami. W pracy porównano API dla Javy Micro Edition, dynamicznie rozwijającego się systemu Android oraz nowego systemu Windows Phone 7.

2. Akcelerometry urządzeń mobilnych

Czujniki stosowane w telefonach komórkowych, można podzielić na dwie podstawowe grupy, tzn. czujniki rzeczywiste do pomiarów wielkości fizycznych, np.: magnetometry, termometry, barometry czy akcelerometry oraz czujniki wirtualne kombinujące i modyfikujące dane z innych czujników rzeczywistych. Przykładem czujników wirtualnych jest czujnik siły sygnału radiowego sieci komórkowej [2].

Czujnikiem, który został wybrany do badań jest akcelerometr. Zainteresowanie autora akcelerometrem wynika z kierunku pracy badawczej dotyczącej cyfrowej rejestracji ruchu ciała ludzkiego (ang. Motion Capture). Jedną z metod rejestracji takiego ruchu jest właśnie wykorzystanie czujników akcelerometrycznych umieszczanych na przegubach ciała ludzkiego [3].



Rys. 1. Model telefonu komórkowego z oznaczeniem osi ruchu [4]
Fig. 1. Model of a cell phone with movement axes [4]

3. Java Micro Edition

Pierwszym z omawianych interfejsów programistycznych jest biblioteka Mobile Sensor API (JSR 256) dla platformy Javy Micro Edition [2].

Platforma Java Micro Edition (JME) zawdzięcza swoją popularność modelowi środowiska uruchomieniowego języka Java. Dzięki zastosowaniu maszyny wirtualnej MIDlety działają niezależnie od rodzaju systemu operacyjnego urządzenia przenośnego. Platforma JME jest więc platformą najbardziej rozpowszechnioną na rynku telefonów komórkowych.

API czujników dostępne dla JME podzielono na dwa pakiety:
- `javax.microedition.sensor`
- `javax.microedition.sensor.control`

Pierwszy pakiet umożliwia odczyt informacji z czujników, drugi zawiera przykładowe kontrolki, za pomocą których można sterować czujnikami, tzn. uruchamiać i zatrzymywać pomiary, ustawiać dokładność, częstotliwość próbkowania lub przeprowadzać kalibrację. Biblioteka ta umożliwia nie tylko pobieranie danych, ale także monitorowanie czujników za pomocą odbiorców zdarzeń (typu `ConditionListener`) i reakcję jedynie na te wartości zmierzone, które spełniają zadane warunki, np. mieszczą się w pewnym przedziale.

Czujnik przed odczytem danych musi być odnaleziony lub jawnie wskazany (poprzez URL), żeby można było otworzyć do niego połączenie. Do zarządzania czujnikami służy klasa `Sensor Manager`. Metoda `findSensor()` tej klasy umożliwia odszukanie informacji o czujniku określonego typu (np. „`accelerometer`"). Zwracany w wyniku poszukiwań obiekt `SensorInfo` umożliwia odczyt adresu URL czujnika i otwarcie połączenia poprzez metodę statyczną `open()` klasy `Connector`. W wyniku jej wykonania zwracany jest obiekt `SensorConnection` umożliwiając odczyt danych z czujnika.

Dane z czujnika mogą być odczytywane synchronicznie poprzez wywołanie metody `SensorConnection.getData()` lub asynchronicznie poprzez rejestrację odbiorcy zdarzeń `DataListener`, który otrzymuje powiadomienia o zebranych danych poprzez automatyczne wywołania metody `dataRecived()`.

Sam obiekt danych – obiekt typu `Data` – zawiera informacje o czasie rejestracji (Timestamp), błędzie pomiaru w postaci wartości odchylenia standardowego oraz umożliwia zwrócenie wartości pomiarów w postaci liczb całkowitych lub rzeczywistych. Dane rejestrowane są przez akcelerometr jednocześnie w trzech wymiarach dlatego zwracane są w postaci tablicy dwuwymiarowej.

Poszczególne kierunki pomiarów (x,y,z w przypadku akcelerometru) reprezentowane są przez kanały. Klasa `ChannelInfo` definiuje właściwości danych takie jak: typ, zakres, dokładność, jednostki i skala. Klasa `Channel` zarządza obiektami warunków `Condition` dołączonymi do kanałów czujnika. W momencie gdy wartości zmierzone spełniają zdefiniowane warunki wywoływana jest metoda `conditionMet()` odbiorcy zdarzeń `ConditionListener`. Dwie podstawowe klasy warunków to: `RangeCondition` – umożliwia definicję interesującego nas zakresu wartości oraz `LimitCondition` – umożliwia reakcję na osiągnięcie przez wartości mierzone założonego limitu.

Szkielet programu do odczytu danych z akcelerometru wygląda następująco:

```
public class HelloMIDlet extends MIDlet implements
                                                DataListener{
    ...
    SensorInfo[] infos = SensorManager.findSensors(
          "acceleration",SensorInfo.CONTEXT_TYPE_USER);
    String sensor_url = infos[0].getUrl();
    SensorConnection con=(SensorConnection)
                          Connector.open(sensor_url);
    con.setDataListener(this, BUFFER_SIZE);
    public void dataReceived(SensorConnection sensor,
                 Data[] data, boolean isDataLost) {
         double[] directions = getDirections(data);
         double x = directions[0];
         double y = directions[1];
         double z = directions[2];
    }
    ...
}
```

Dodatkowo czujniki mogą pracować w różnych kontekstach w zależności od tego, czy ich zdarzenia wywołuje użytkownik (tak jak w przypadku akcelerometru) czy system. Biblioteka umożliwia także określanie i weryfikację uprawnień umożliwiających korzystanie z czujników poprzez aplikacje.

Specyfikacja Mobile Sensor API (JSR 256) wyszczególnia aż 43 rodzaje czujników wstępnie przewidzianych do obsługi. Wśród nich znajdują się między innymi akcelerometr, amperomierz, żyroskop, kompas, odbiornik GPS, myszka, joystick, czytnik linii papilarnych oraz prędkościomierz [2].

4. Android

Rożne telefony z systemem Android wyposażone są w różnego rodzaju czujniki. Wśród nich znajdują się: czujniki temperatury, ciśnienia, światła, pola magnetycznego, odległości oraz żyroskopy. Jako czujnik traktowana jest także kamera telefonu.

Standardowe dwa czujniki w telefonach z Androidem to trzyosiowy akcelerometr mierzący przyspieszenie i pośrednio nachylenie urządzenia oraz trzyosiowy magnetometr umożliwiający wskazanie położenia geograficznego. Dwa wektory danych pochodzące z tych czujników umożliwiają prostą estymację rotacji urządzenia [5].

API związane z oprogramowaniem czujników zebrano w pakiecie: android.hardware. Klasą zarządzającą dostępem do czujników jest SensorManager. Klasa ta zawiera stałe, które reprezentują określone wartości niezmienne dla czujników, takie jak np. ciśnienie atmosferyczne na poziomie morza, minimalna i maksymalna wartość pola magnetycznego Ziemi, a także stałe określające dokładność i opóźnienie pomiaru. SensorManager posiada metody: zwracające referencje do czujników, macierze rotacji, orientacji, inklinacji, zwracające zmiany wychylenia, wysokość nad poziomem morza, a także metody do rejestracji i wyrejestrowywania odbiorców zdarzeń.

Kolejna klasą z pakietu, której instancje zwraca SensorManager, jest klasa Sensor. Określa ona rodzaje czujników poprzez definicję stałych reprezentujących: akcelerometr, czujnik grawitacji, żyroskop, czujnik światła, pola magnetycznego, orientacji, odległości, rotacji i temperatury. Metody klasy umożliwiają: określenie maksymalnego zakresu pomiarów, minimalnego opóźnienia, rozdzielczości, wersji, nazwy i producenta czujnika.

Dane z czujnika odczytywane są asynchronicznie w wyniku wywołania metody onSensorChanged() odbiorcy zdarzeń typu SensorEventListener. Przekazywany do metody obiekt zdarzenia SensorEvent zawiera informacje o chwili pomiaru, dokładności, wartościach zmierzonych i czujniku generującym zdarzenie.

Przykładowa klasa aktywności (podstawowej jednostki funkcjonalnej aplikacji systemu Android – odpowiednika okna w systemie komputera osobistego) wyświetlająca dane z akcelerometru wygląda następująco:

```
public class AccellerationActivity extends Activity{
    public void onCreate(Bundle savedInstanceState){
         SensorManager sensorManager = (SensorManager)
           getSystemService(Context.SENSOR_SERVICE);
         Sensor sensor = sensorManager.getSensorList(
                  Sensor.TYPE_ACCELEROMETER).get(0);
         sensorManager.registerListener(accListener,
                  sensor,SensorManager.SENSOR_DELAY_GAME);
    }
    SensorEventListener accListener =
                  new SensorEventListener() {
         public void onAccuracyChanged(Sensor sensor,
                                       int acc) {
         }
         public void onSensorChanged(SensorEvent event) {
                  float x = event.values[0];
                  float y = event.values[1];
                  float z = event.values[2];
                  refreshDisplay();
         }
    };
    private void refreshDisplay() {...}
}
```

5. Windows Phone 7

Windows Phone 7 (WP7) jest nowym systemem operacyjnym, który dopiero się rozwija. Dlatego obsługa czujników jest dużo mniej zaawansowana niż w przypadku konkurencji. Z poziomu API mamy do dyspozycji możliwości oprogramowania akcelero-

metru, kompasu i żyroskopu. Dodatkowo występuje tu ciekawe rozwiązanie, tzn. klasa `Motion`, która zbiera dane ze wszystkich trzech czujników i wykonuje niskopoziomowe obliczenia umożliwiając aplikacji prosty odczyt: położenia urządzenia w przestrzeni, przyspieszenia rotacji oraz liniowego względem ziemi oraz przyspieszenia w wyniku ruchu [6, 7].

W przypadku gdy potrzebujemy nieprzetworzonych danych z akcelerometru musimy wykorzystać klasę `Accelerometer`.

Stworzenie aplikacji obsługującej akcelerometr wymaga dodania dwóch kolekcji bibliotecznych (assembly):

- `Microsoft.Devices.Sensors` – API czujników
- `Microsoft.Xna.Framework` – zawiera `Vector3` dla danych z akcelerometru

W programie konieczne jest dodanie mechanizmu obsługi zdarzeń typu `CurrentValueChanged`. Odbiorca zdarzeń działa w wątku drugoplanowym i dlatego w celu odświeżenia ekranu wykorzystuje wątek obsługi interfejsu graficznego `Dispatcher`. Wykonuje wtedy metodę `UpdateUI`, do której przekazuje odczytane z akcelerometru dane. Dane z akcelerometru zwracane są w postaci struktury `AccelerometerReading`, która zawiera informacje o przyspieszeniu i chwili czasowej w której dokonany został pomiar.

Niestety API Windows Phone 7 nie zawiera na razie żadnych metod weryfikacji specyfikacji wbudowanych czujników. W związku z tym nie uzyskamy programowo informacji o ich modelach i dokładności.

6. Badania

Możliwości omawianych interfejsów programistycznych zweryfikowano w praktyce na rzeczywistych urządzeniach. Do badań wykorzystano dwa telefony: Nokię 5800 XpressMusic z systemem operacyjnym Symbian i maszyną wirtualną Javy ME oraz LG GT540 Swift z zainstalowanym systemem Android.

Tab. 1. Specyfikacje akcelerometrów w telefonach Nokia 5800 i LG GT540
Tab. 1. Specifications of accelerometers built in Nokia 5800 and LG GT540

Nokia 5800 XpressMusic		**LG GT540 Swift**	
Parametr	**Wartość**	**Parametr**	**Wartość**
Model	Nokia	Nazwa	BMA 150
Zakres pomiarowy [m/s^2]	<-19,62; 19,62>	Zakres pomiarowy [m/s^2]	<-19,61; 19,61>
Jednostka	m/s^2	Wersja	1
Rozdzielczość [m/s^2]	0,15	Rozdzielczość [m/s^2]	1,0
Dokładność	0,1	Maksymalny zakres	1,0
Typ danych	double	Zasilanie [mA]	20
Średnie wartości przyspieszenia ziemskiego wzdłuż osi x, y i z [m/s^2]	x = 9,1 y = 9,3 z = 9,5	Średnie wartości przyspieszenia ziemskiego wzdłuż osi x, y i z [m/s^2]	x = 10,5 y = 9,3 z = 10,4

Tabela (tab. 1) zawiera specyfikacje zainstalowanych akcelerometrów w badanych urządzeniach oraz wartości przyspieszeń oddziałujących na urządzenia znajdujące się w spoczynku i ustawianych tak by przyspieszenie ziemskie oddziaływało wzdłuż osi x, y i z. Dane te uzyskano poprzez wywołania dostępnych funkcji API.

Tabela przedstawia różne zestawy parametry dla obu urządzeń. Wynika to z tego, że API dla Javy ME oraz systemu Android różnią się od siebie i umożliwiają odczyt wartości innych parametrów akcelerometru.

Zakresy pomiarowe obu urządzeń są bardzo zbliżone jednak dokładność akcelerometru w Nokii jest prawie siedmiokrotnie większa niż zainstalowanego w telefonie LG, ale Nokia jest produktem z wyższej półki cenowej.

Nie przetestowano API akcelerometru dla systemu Windows Phone 7 ze względu na to, że na telefonie z tym systemem operacyjnym nie jest możliwa instalacja własnych aplikacji, a jedynie aplikacji pochodzących ze sklepu internetowego Windows Phone Marketplace.

7. Wnioski

Najbardziej zaawansowany interfejs programistyczny z badanych oferuje platforma JME. Implementacja referencyjna umożliwia odczyt parametrów akcelerometru określających zakres pomiarowy, dokładność i rozdzielczość. Dane mogą być pobierane w sposób asynchroniczny – poprzez obsługę zdarzenia i synchroniczny – na żądanie aplikacji.

API o nieco mniejszej funkcjonalności stworzono dla systemu Android. Interfejs ten wspiera kilka różnych rodzajów czujników. W celu ustalania położenia telefonu API czujników może wykorzystać nie tylko dane z akcelerometru jak to miało miejsce w przypadku JME, ale także magnetometru. API umożliwia odczyt danych w sposób asynchroniczny.

Ostatnia z badanych platform - System Windows Phone 7 posiada najbardziej ubogie API do współpracy z czujnikami. Wprawdzie można ustawić okres próbkowania (czego nie było w JME i Androidzie) jednak nie ma możliwości określenia podstawowych parametrów, takich jak dokładności pomiaru. Nowością jest tu także możliwość wyliczenia ruchu urządzenia na podstawie zbiorczych danych z aż trzech czujników, tj. akcelerometru, magnetometru i żyroskopu.

Najlepszym wyborem na dzień dzisiejszy jest API JME, ale platformy Android i WP7 szybko się rozwijają, stosowane są na bardzo wydajnych telefonach i wkrótce mogą zaoferować dużo lepsze rozwiązania. Oferują także możliwość obliczania położenia na podstawie danych z kilku czujników, co ma bardzo duże znaczenie ze względu na duży błąd pomiaru stosowanych obecnie akcelerometrów – sięgający 7 % - co łatwo zauważyć na podstawie wyników z tabeli 1.

8. Literatura

[1] Pettey C., Stevens H.: Gartner Says 428 Million Mobile Communication Devices Sold Worldwide in First Quarter 2011, a 19 Percent Increase Year-on-Year, Press Releases, Egham, May 19, 2011.

[2] JSR 256: Mobile Sensor API, http://download.oracle.com/otndocs/jcp/mobile_sensor-1.2-mrel2-oth-JSpec/, 2011.

[3] Slyper R, Hodgins J. K.: Action Capture with Accelerometers, Eurographics/ Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, Dublin, 2008.

[4] Osie telefonu – obraz, http://developer.android.com/images/axis_device.png, 2011.

[5] Steele J.: The Android Developer's Cookbook. Building Applications with the Android SDK, Addison Wesley, 2011.

[6] Miles R.: Windows Phone Programming in C#, Windows Phone Marketplace, Program Ideas, November, 2010.

[7] MSDN, Microsoft.Devices.Sensors Namespace, http://msdn.microsoft.com/en-us/library/microsoft.devices.sensors%28v=VS.92%29.aspx, May 19, 2011.

otrzymano / received: 08.09.2011
przyjęto do druku / accepted: 02.11.2011 *artykuł recenzowany*