Politechnika Wrocławska

Wykład 11

Zastosowanie programu MATLAB w zagadnieniach inżynierskich

Wojciech Myszka

Katedra Mechaniki, Inżynierii Materiałowej i Biomedycznej

14 grudnia 2024

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

1. Debugowanie programów
2. Akwizycja danych
3. Raspberry Pi
4. Matlab na telefonie
5. Przykłady
6. Obliczenia lokalne
7. Integracja z desktopem

Debugowanie programów

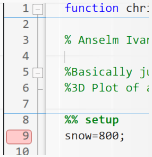
Politechnika Wrocławska

Debugowanie funkcji

1. To również właściwie wyższa szkoła jazdy, ale trzeba powiedzieć parę słów:
   - *bug* to po angielsku *robal*,
   - programiści nazywają tak błędy kodu,
   - zatem *de-bugowanie* (*debugging*) to usuwanie błędów z kodu.
2. *breakpoint* (*punkt przerwania*) to takie miejsce, w którym kod programu (skryptu) zatrzymuje swoje działanie pozwalając programiście na inspekcję zmiennych, wznowienie obliczeń lub tylko przejście do następnego polecenia.

Przykład I

1. Otwieramy funkcję w edytorze
2. Przechodzimy kursorem do linii, której chcemy ustawić punkt przerwania i naciskamy F12. Ewentualnie można jednokrotnie kliknąć na numer linii. Numer linii zaznaczony zostanie na czerwono. Wybrałem pierwszą **wykonywalną** instrukcję w funkcji, żeby wykonane programu zatrzymało się zaraz po wejściu do funkcji.

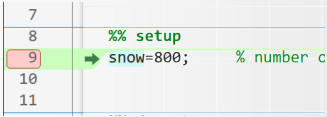


3. Uruchamiamy funkcję. W tym wypadku w głównym oknie MATLABa piszemy `christmas`

Przykład II

```
>> christmas
9    snow=800;      % number of snow flakes [0 .. 5000]
K>>
```

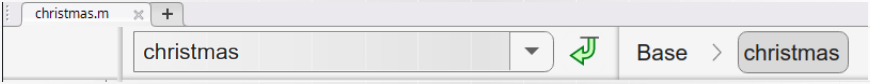
4. Widok okna edytora ciut zmienił się: zielona strzałka jest przy linii numer 9, najeżdżając kursorem na zmienną `varargin` w pierwszej linii możemy podejrzeć jej wartość.



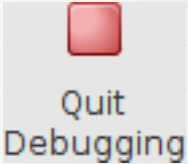
5. W głównym oknie MATLABa mamy dostęp do zmiennych funkcji.
6. W oknie edytora możemy korzystać z:

Przykład III

-  — klawisz Step — przejdź do następnego polecenia w kodzie;
-  — klawisz Step in — wejdź do wnętrza wywoływanej funkcji;
-  — klawisz Step out — wyjdź z funkcji;
-  — klawisz Run to cursor — wykonaj kolejne polecenia aż do miejsca, w którym znajduje się kursor;
- na górze edytora mamy historię wywołań funkcji *function call stack* ;



Przykład IV

- 

— pozwala zakończyć debugowanie (pracę krokową).

Debugowanie *Live script*

Początek jest ciut inny;

1. W miejscu, w którym chcemy ustawić pynkt przerwania kilkamy prawym plawiszem myszy na numerze linii i wybieramy z menu **Set Breakpoint**. Punkt przerwania zaznaczony będzie identycznie jak w edytorze „zwykłym”.
2. Alternatywnie można rozpocząć od naciśnięcia klawisza 

Akwizycja danych

MATLAB zbieranie danych

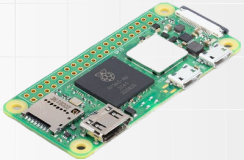
Jest kilka pakietów:

1. Data Acquisition Toolbox
2. Instrument Control Toolbox
3. Image Acquisition Toolbox™
4. Vehicle Network Toolbox (CAN bus interface devices)
5. Industrial Communication Toolbox

Raspberry Pi

Połączenie z Raspberry Pi

1. Raspberry Pi to komputer niezbyt wielkich rozmiarów.



Rysunek 1: Pi Zero

Rysunek 2: Raspberry Pi Pico

Rysunek 3: Raspberry Pi 4

Rysunek 4: Raspberry Pi 500

Interfejsy Raspberry Pi

1. USB
2. HDMI
3. Ethernet
4. Wi-Fi
5. Bluetooth
6. LED
7. Serial
8. GPIO
9. I2C
10. SPI
11. Camera

Politechnika Wrocławska

Instalacja oproramowania

1. Trzeba zainstalować pakiet oprogramowania (**matlab-rpi**) na Rasberry Pi i skonfigurować go

   Można też użyć prekonfigurowanej wersji systemu dostarczonej przez firmę Mathworks.

2. Trzaba zainstalować **MATLAB Support Package for Raspberry Pi Hardware**

3. Potrzebne będzie konto Mathworks

Możliwości

1. **Zdalna komunikacja:**

   Można łączyć się zdalnie, dokonywać pomiarów z sensorów, korzystać z kamery...

2. **Uruchamianie aplikacji:**

   Można tworzyć aplikacje które (po konwersji do C) będą uruchamiane na komputerze Raspberry

Politechnika Wrocławska

Wspierane urządzenia

Raspberry Pi Model	MATLAB Releases Supported	Supported in MATLAB Online?
Raspberry Pi 1 Model B (discontinued)	R2014a – Current	No
Raspberry Pi 1 Model B+	R2014b – Current	No
Raspberry Pi 2 Model B	R2014b – Current	Yes
Raspberry Pi 3 Model B	R2016a – Current	Yes
Raspberry Pi Zero W	R2018a – Current	No

Politechnika Wrocławska

Wspierane urządzenia

Raspberry Pi Model	MATLAB Releases Supported	Supported in MATLAB Online?
Raspberry Pi 3 Model B+	R2018b – Current	Yes
Raspberry Pi 4 Model B	R2020a – Current	Yes
Raspberry Pi Zero 2 W	R2023a – Current	Yes
Raspberry Pi Compute Module 4	R2023a – Current	Yes
Raspberry Pi 5	R2024b – Current	Yes

Note: Raspberry Pi 1 Model A, Raspberry Pi 1 Model A+, and Raspberry Pi

Matlab na telefonie

Politechnika Wrocławska

MATLAB Mobile

- dostępne dla Android
- dostępne dla IOS (Apple)
- po zalogowaniu się do konta Mathworks automatycznie synchronizuje pliki z Matlab Drive
- pozwala wykonywać obliczenia, prezentować wykresy
- nie umożliwia na manipulowanie wykresami 3D
- dosyć niewygodne (mały ekran, malutka klawiatura),...
- daje łatwy dostęp do czujników telefonu

Politechnika Wrocławska

Disclaimer

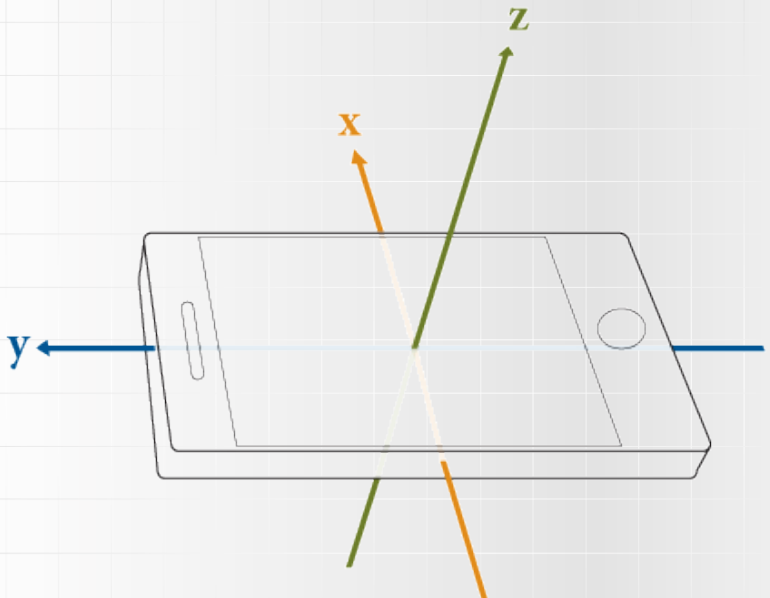
Nie żebym zachęcał do albo reklamował jakieś rozwiązania, ale każdy telefon komórkowy wyposażony jest w zestaw sensorów mogących dostarczać (mniej lub bardziej) **prawdziwe** dane pomiarowe. Matlab pozwala na zbieranie danych z czujników

- **Microphone** — Mikrofon
- **Acceleration** — Przyśpieszenie (trzy osie) m/s^2
- **Magnetic Field** — Trzy skłądowe pola magnetycznego (mikrotesle)
- **Orientation** — Odczyt pozycji we współrzędnych „X", „Y" i „Z", w stopniach, dla azymutu, nachylenia i obrotu
- **Angular Velocity** — szybkość kątowa w trzech osiach (rad/s)
- **Position** — współrzędne, wysokość kierunek ruchu, szybkość i dokładność pozioma

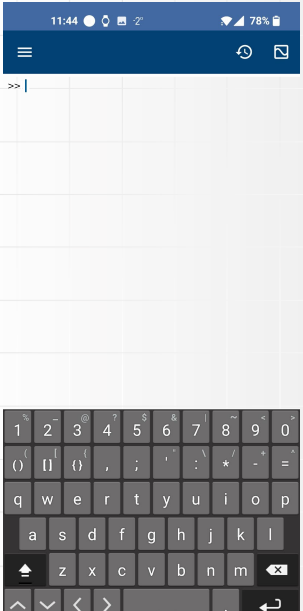
Oprócz tych czujników można również korzystać z kamery

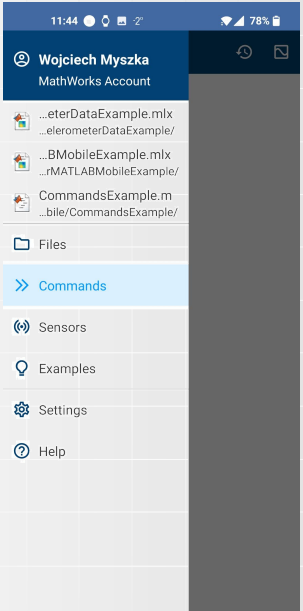
Politechnika Wrocławska

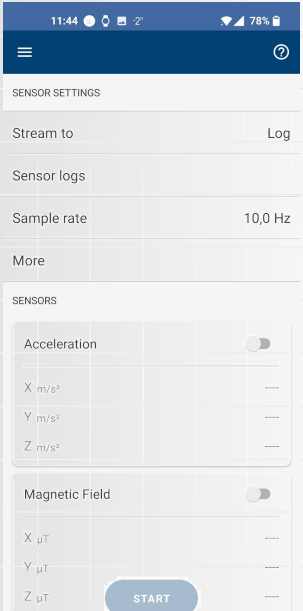
Orientacja osi



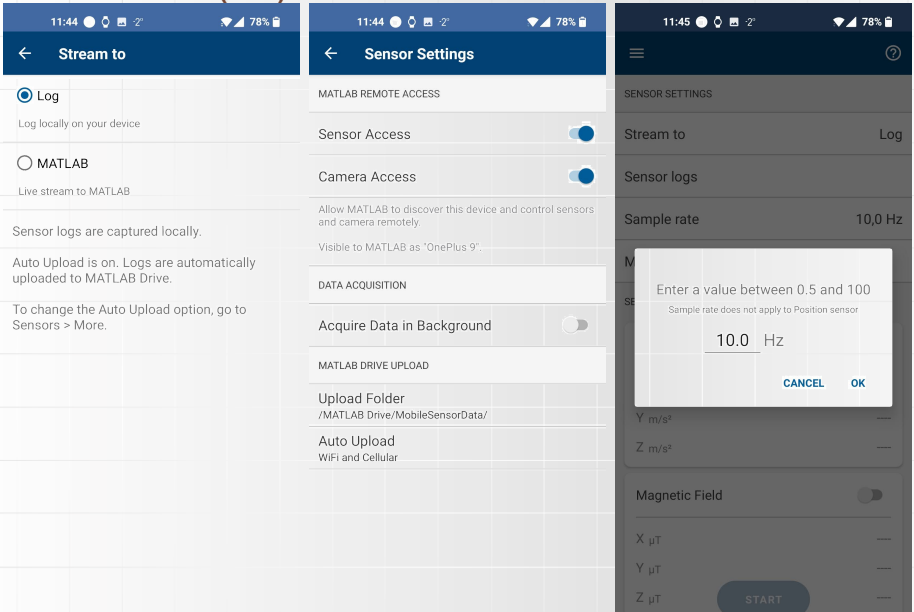
Widok ekranu





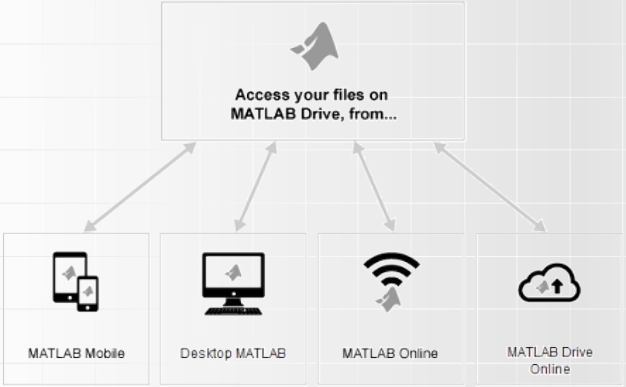


Widok ekranu (cd)



MATLAB Drive

- Dodatkowo MATLAB Mobile zapewnia synchronizację plików z MATLAB Drive



Akwizycja danych

1. Rozpoczęcie zbierania danych i zapis ich do lokalnego pliku jest bardzo proste
2. Po uruchomieniu aplikacji należy aktywować zbieranie danych i zaznaczyć, że mają być one gromadzone w plikach.
3. Do przetwarzania pliki można albo ściągnąć z MATLAB Drive, albo będą dostępne bezpośrednio w aplikacji (ikonka ).
4. Później wystarczy nacisnąć klawisz START.
5. Po zakończeniu zbierania danych — naciskamy STOP.

Przykłady

Wizualizacja danych GPS (zebranych przez telefon) I

Skrypt, omówię go dokładnie.

```
%% Mapa spaceru
% za https://www.mathworks.com/help/matlabmobile/ug/
% acquire-gps-data-and-plot-your-location-and-speed-on-a-map.
% html
% dane wejścowe: zapisany na telefonie log

lat=Position.latitude;
lon=Position.longitude;
t=Position.Timestamp;
spd=Position.speed;
%
% Skrypt w sprytny sposób zechce pokolorować trasę w zależności
```

Wizualizacja danych GPS (zebranych przez telefon) II

```
nBins = 2; % liczba kategorii
binSpacing = (max(spd) - min(spd))/nBins;
binRanges = min(spd):binSpacing:max(spd)-binSpacing;

% Add an inf to binRanges to enclose the values above
% the last bin.
binRanges(end+1) = inf;

% |histc| określa do której kategorii wpada każdy ponkt
[~, spdBins] = histc(spd, binRanges);
% transpozycja danych
lat = lat';
lon = lon';
spdBins = spdBins';
```

Wizualizacja danych GPS (zebranych przez telefon) III

```
% Create a geographical shape vector, which stores the line
% segments as features.
% trzeba dodać Map Toolbox
s = geoshape();

for k = 1:nBins

    % Keep only the lat/lon values which match the current bin.
    % Leave the rest as NaN, which are interpreted as breaks
    % in the line segments.
    latValid = nan(1, length(lat));
    latValid(spdBins==k) = lat(spdBins==k);
```

Wizualizacja danych GPS (zebranych przez telefon) IV

```
lonValid = nan(1, length(lon));
lonValid(spdBins==k) = lon(spdBins==k);

% To make the path continuous despite being segmented into
% different colors, the lat/lon values that occur after
% transitioning from the current speed bin to another speed
% bin will need to be kept.
transitions = [diff(spdBins) 0];
insertionInd = find(spdBins==k & transitions~=0) + 1;

% Preallocate space for and insert the extra lat/lon values.
latSeg = zeros(1, length(latValid) + length(insertionInd));
latSeg(insertionInd + (0:length(insertionInd)-1)) = ...
lat(insertionInd);
```

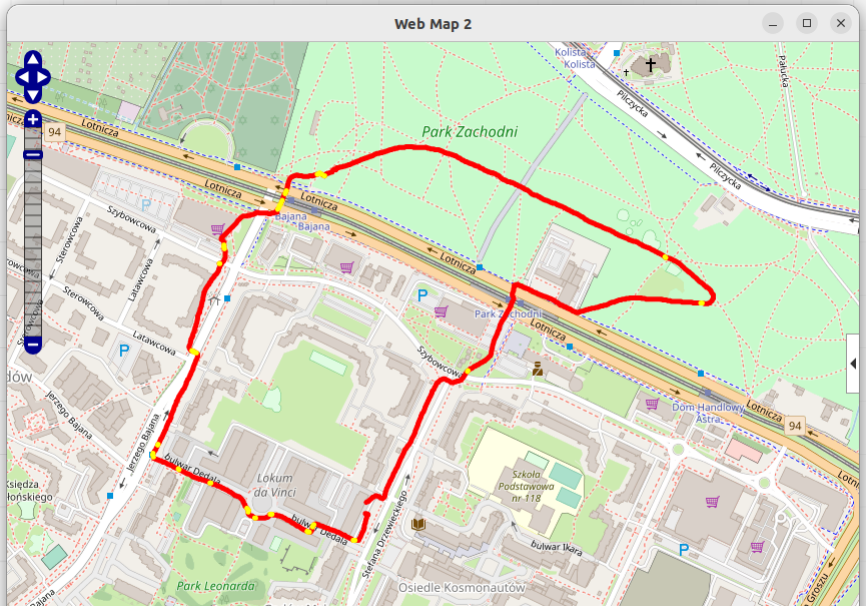
Wizualizacja danych GPS (zebranych przez telefon) V

```
        latSeg(~latSeg) = latValid;

        lonSeg = zeros(1, length(lonValid) + length(insertionInd));
        lonSeg(insertionInd + (0:length(insertionInd)-1)) = ...
        lon(insertionInd);
        lonSeg(~lonSeg) = lonValid;

        % Add the lat/lon segments to the geographic shape vector.
        s(k) = geoshape(latSeg, lonSeg);

    end
    wm = webmap('Open Street Map');
    colors = autumn(nBins);
```

Wizualizacja danych GPS (zebranych przez telefon) VI

```
wmline(s, 'Color', colors, 'Width', 5);
wmzoom(16);
```

Efekt


Web Map 2
Park Zachodni
Lotnicza
94
Bajana
Szybowcowa
Sterowcowa
Latawcowa
Jerzego Bajana
bulwar Dedala
Lokum da Vinci
Park Leonarda
Osiedle Kosmonautów
Szkoła Podstawowa nr 118
Dom Handlowy Astra
bulwar Ikara
Stefana Drzewińskiego
Pilicycka
Kolista
Pabiańska

Zliczanie kroków I

- Użyjemy akcelerometrów
- Ponieważ dane zbierane są w trzech składowych — trzeba wyliczyć moduły tych wektorów
- Akcelerometr mierzy również przyśpieszenie ziemskie — stąd spora stała (≈ 10); odejmujemy ją
- Pomiary zaburzone są — stąd „eliminujemy” wszystkie piki mniejsze niż odchylenie standardowe
- Poniżej kod programu, który krok, po kroku objaśnię

```
%% kroki.m
% Liczy kroki na podstawie pomiarów z akcelerometru
% na podstawie: https://www.mathworks.com/help/
% matlabmobile_android/ug/counting-steps-by-capturing-
% acceleration-data.html
```

Politechnika Wrocławska

Zliczanie kroków II

```
figure("Name", "trzy składwe")
plot(Acceleration.Timestamp, Acceleration.X, ...
Acceleration.Timestamp, Acceleration.Y, ...
Acceleration.Timestamp, Acceleration.Z);
legend('X', 'Y', 'Z');
xlabel('Czas (s)');
ylabel('Przyśpieszenie (m/s^2)');
saveas(gcf,"w01.png");
x = Acceleration.X;
y = Acceleration.Y;
z = Acceleration.Z;
t = Acceleration.Timestamp;
mag = sqrt(sum(x.^2 + y.^2 + z.^2, 2));
```

Zliczanie kroków III

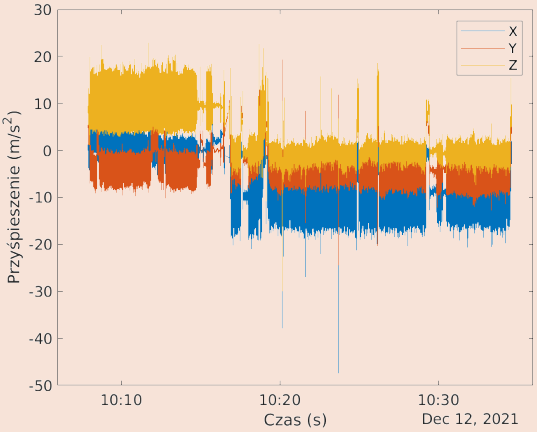
```
disp("średnia" + string(mean(mag)));

magNoG = mag - mean(mag);
figure("Name", "kroki");
plot(t,magNoG);
xlabel('Time (s)');
ylabel('Acceleration (m/s^2)');

minPeakHeight = std(magNoG);

[pks,locs] = findpeaks(magNoG,'MINPEAKHEIGHT',minPeakHeight);

numSteps = numel(pks)
```

Zliczanie kroków IV

```
hold on;
plot(t(locs), pks, 'r', 'Marker', 'v', 'LineStyle', 'none');
title('Counting Steps');
xlabel('Time (s)');
ylabel('Acceleration Magnitude, No Gravity (m/s^2)');
hold off;
saveas(gcf,"w02.png");
```

Zliczanie kroków V

Zmiana orientacji telefonu

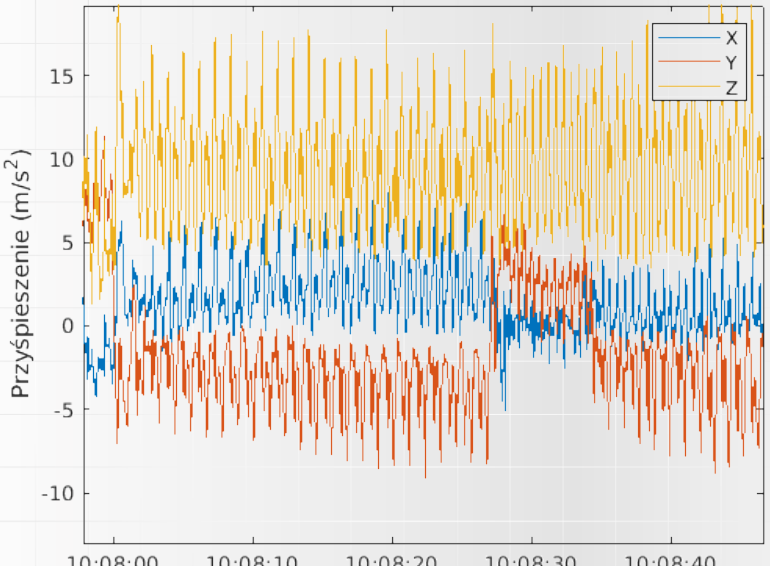


Politechnika Wrocławska

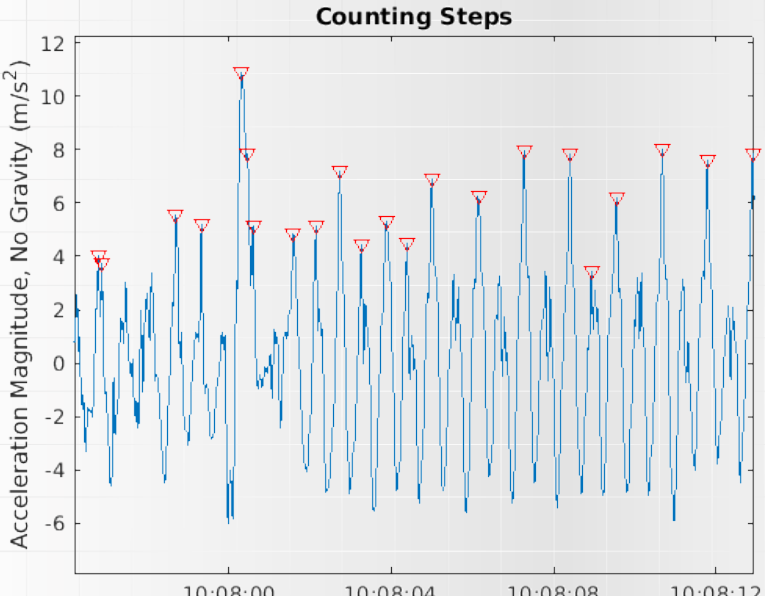
Podsumowanie

1. Kroki dodatkowo policzyłem za pomocą zegarka — 2633 kroków
2. Aplikacji w telefonie — 2671 kroków
3. Wartość z obliczeń — 2908 kroków
4. ...

Akcelerometry



Kroki


Counting Steps
Acceleration Magnitude, No Gravity (m/s^2)
12
10
8
6
4
2
0
-2
-4
-6
10:08:00
10:08:04
10:08:08
10:08:12

Obliczenia lokalne

Politechnika Wrocławska

Przetwarzanie lokalne

1. Przełączamy („Stream to MATLAB”)
2. włączam jedynie zbieranie danych z akcelerometru.
3. W oknie poleceń piszę

```
m = mobiledev;
m.Logging = 1;
```

4. Teraz rzucam telefonem i zatrzymuję zbieranie danych

```
m.Logging = 0;
```

5. Do zebranych danych dobieramy się korzystając z funkcji `accellog()`

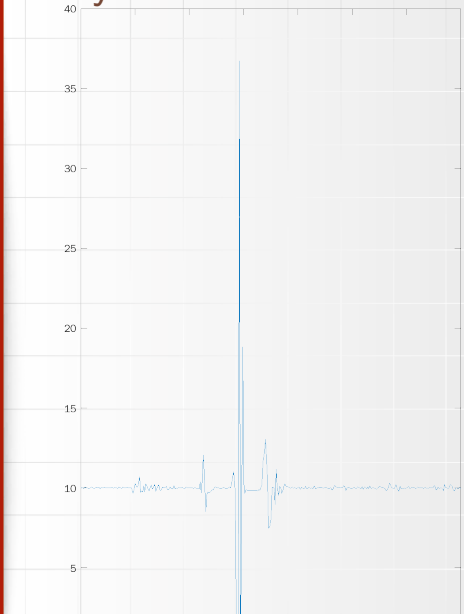
```
[a,t] = accellog(m)
```

(czas będzie czasem względnym)

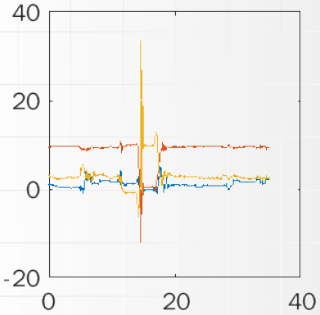
6. Można dane przetwarzać, albo rysować wykresy

Politechnika Wrocławska

Otrzymane obrazki



Politechnika Wrocławska



Inne funkcje I

Zamiast włączać funkcje logowania z menu, można użyć poleceń:

1. `m.AccelerationSensorEnabled = 1`
2. `m.AngularVelocitySensorEnabled = 1`
3. `m.MagneticSensorEnabled = 1`
4. `m.OrientationSensorEnabled = 1`
5. `m.PositionSensorEnabled = 1`

Wstawienie zera spowoduje wyłączenie sensora

Zbieranie danych

1. `accellog(m)` — log akceleratorów
2. `angvellog(m)` — log prędkości kątowych
3. `magfieldlog(m)` — log magnetometrów
4. `orientlog(m)` — log danych o orientacji

Politechnika Wrocławska

Inne funkcje II

5. `poslog(m)` — log danych GPS: długość, szerokość geograficzna, prędkość, kurs, wysokość (npm), poziomą dokładność danych GPS
6. `discardlogs(m)` — kasuje zebrane danych
7. `clear m` — kończy sesję „współpracy z sensorami”

Kamera I

1. Po wydaniu polecenia

   ```
   m = mobiledev
   ```

   dostajemy informację, która wygląda jakoś tak:

   ```
   m =
   mobiledev with properties:
                       Connected: 1
                 AvailableCameras: {'back' 'front'}
                          Logging: 0
                 InitialTimestamp: ''

      AccelerationSensorEnabled: 0
   AngularVelocitySensorEnabled: 0
   ```

Kamera II

```
    MagneticSensorEnabled: 0
OrientationSensorEnabled: 0
    PositionSensorEnabled: 0
```

2. Jest tam kamera.
3. Wybieramy przednią lub tylną

   ```
   cam = camera(m,'back') % lub `front`
   ```

   Zwrotna informacja wygląda jakoś tak:

Kamera III

```
cam =
  Camera with properties:
                       Name: 'back'
    AvailableResolutions: {'640x480'  '1280x720'}
                 Resolution: '1280x720'
                      Flash: 'on'
                  Autofocus: 'off'
```

Możemy modyfikować właściwości rozdzielczość, lampa błyskowa czy ustawianie ostrości

4. Zdjęcie wykonujemy poleceniem

```
img = snapshot(cam,'immediate'); % lub `manual`
```

wyświetlamy poleceniem

Kamera IV

```
image(img);
```

img jest obrazem typu RGB

można go zapisać poleceniem

```
imwrite(img,"a.jpg")
```

Integracja z desktopem

Android Sensor Support from MATLAB

1. Po zainstalowaniu dodatkowego pakietu na desktopie dostajemy nowe, ciekawe możliwości.

   Trzeba poszukać *Android Sensor Support from MATLAB*

2. W szczególności mamy nieomal identyczny dostęp do czujników jak na telefonie
3. Analogiczny pakiet jest dostępny dla systemu IOS.
4. Trzeba być zalogowanym do swojego konta Mathworks i na telefonie i na komputerze

Przykład I

1. Uruchamiamy aplikację na telefonie (i sprawdzamy czy jest połączona z kontem)
2. Dodatkowo trzeba zadbać, żeby za szybko nie przechodziła w stan uśpienia
3. na desktopie wydajemy polecenie

```
m = mobiledev
 m =
mobiledev with properties:
                    Device: 'OnePlus 12'
                 Connected: 1
                   Logging: 1
           AvailableCameras: {'back' 'back-1' 'back-2' 'front'}
```

Przykład II

```
          AvailableMicrophones: {'CPH2581-back' 'CPH2581-bottom'}
            SelectedMicrophone: 'CPH2581-back'
              InitialTimestamp: '14-Dec-2024 12:11:15.852'
    AccelerationSensorEnabled: 1    (131 Logged values)
    AngularVelocitySensorEnabled: 1    (16 Logged values)
        MagneticSensorEnabled: 1    (27 Logged values)
     OrientationSensorEnabled: 1    (21 Logged values)
        PositionSensorEnabled: 1
            MicrophoneEnabled: 0
    Current Sensor Values:
                  Acceleration: [0.1490 0.4708 9.8638] (m/s^2)
               AngularVelocity: [0.0000 -0.0004 0.0000] (rad/s)
                 MagneticField: [14.7000 13.6500 -19.6875] (microt
                   Orientation: [-47.4635 -2.7898 -0.8418] (degree
```

Przykład III

```
Position Data:
        Latitude: [0x1 double] (Waiting for data)
       Longitude: [0x1 double] (Waiting for data)
           Speed: [0x1 double] (Waiting for data)
          Course: [0x1 double] (Waiting for data)
        Altitude: [0x1 double] (Waiting for data)
HorizontalAccuracy: [0x1 double] (Waiting for data)
Show all properties
```

Czyli mamy dostęp do logów i do kamery...

4. Zbierzmy trochę danych

Przykład IV

```
m.Logging=1
m =
mobiledev with properties:
                        Device: 'OnePlus 9'
                     Connected: 1
               AvailableCameras: {'back' 'front'}
                       Logging: 1
                  InitialTimestamp: ''

    AccelerationSensorEnabled: 1
AngularVelocitySensorEnabled: 0
       MagneticSensorEnabled: 0
   OrientationSensorEnabled: 0
      PositionSensorEnabled: 0
```

Przykład V

```
Current Sensor Values:
                Acceleration: [0x3 double] (Waiting for data)
```

5. I rzucam telefonem (na kanapę)

```
m.Logging=0
>> m.Logging=0
m =
mobiledev with properties:
                       Device: 'OnePlus 9'
                    Connected: 1
              AvailableCameras: {'back' 'front'}
                      Logging: 0
              InitialTimestamp: '13-Dec-2021 15:26:15.490'
```

Przykład VI

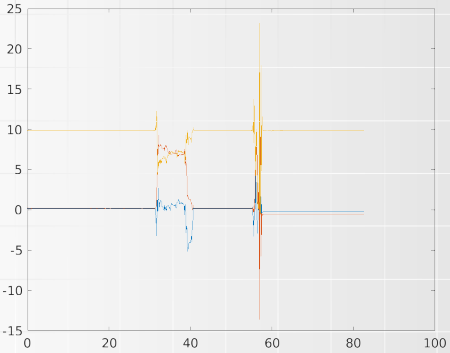
```
    AccelerationSensorEnabled: 1    (825 Logged values)
AngularVelocitySensorEnabled: 0
       MagneticSensorEnabled: 0
    OrientationSensorEnabled: 0
       PositionSensorEnabled: 0
```

Jak widać zebrał trochę danych.

6. Odzyskajmy dane z akcelerometrów

```
[a,t]=accellog(m);
plot(t,a);
```

Przykład VII



7. Te pierwsze „zaburzenia” to ruch telefonem, gdy sprawdzałem czy po wydaniu polecenia rozpoczęło się zbieranie danych.

Teraz kamera I

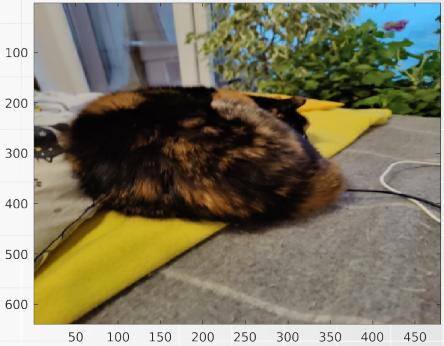
1. Teraz może jeszcze kamera.

```
cam = camera(m,'back')
cam =
  Camera with properties:
                      Name: 'back'
    AvailableResolutions: {'640x480'  '1280x720'}
              Resolution: '640x480'
                   Flash: 'off'
                Autofocus: 'on'
```

2. I fotka

```
img = snapshot(cam,'manual');
image(img)
```

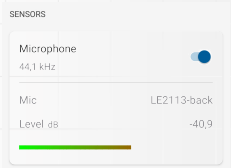
Teraz kamera II



3. Czyli działa.

Teraz mikrofon I

1. Udostępniamy na urządzeniu mobilnym mikrofon



2. Wykonujemy polecenie

```
>> m=mobiledev;
```

3. Naciskamy `Start` na telefonie (dźwięk jest nagrywany); możemy sprawdzić ile już nagrane

Teraz mikrofon II

```
m =
mobiledev with properties:
                        Device: 'OnePlus 9'
                     Connected: 1
                       Logging: 1
                AvailableCameras: {'back' 'front'}
           AvailableMicrophones: {'LE2113-back' 'LE2113-bottom'}
             SelectedMicrophone: 'LE2113-back'

                InitialTimestamp: ''

    AccelerationSensorEnabled: 0
AngularVelocitySensorEnabled: 0
       MagneticSensorEnabled: 0
```

Teraz mikrofon III

```
OrientationSensorEnabled: 0
   PositionSensorEnabled: 0
       MicrophoneEnabled: 1    (386880 Logged samples)
```

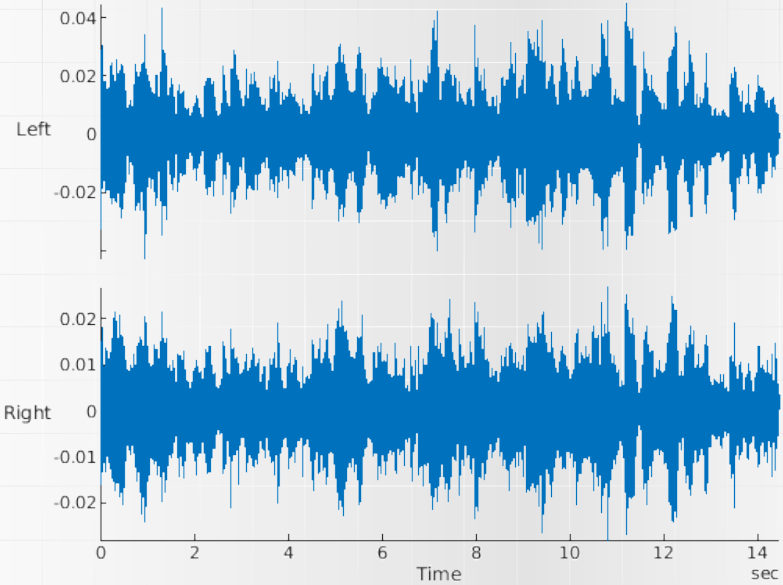
4. Naciskamy Stop i odczytujemy zebrane dane używając funkcji `readAudio`

   ```
   audTT = readAudio(m,OutputFormat="timetable");
   ```

5. Zebrane dane można obejrzeć:

   ```
   stackedplot(audTT)
   ```

Teraz mikrofon IV



Politechnika Wrocławska