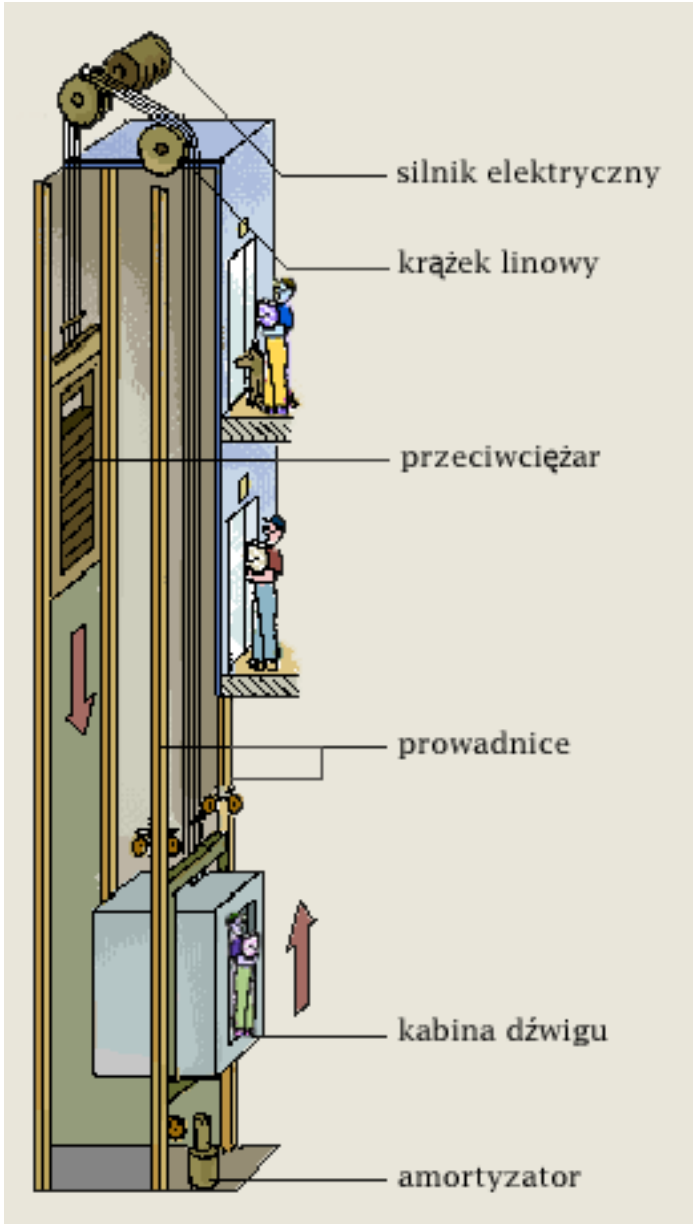
Analiza wpływu sterowania ruchem windy na jej dynamikę

Mgr inż. Krystian Polczyński
Dr inż. Adam Wijata

Łódź, 2022

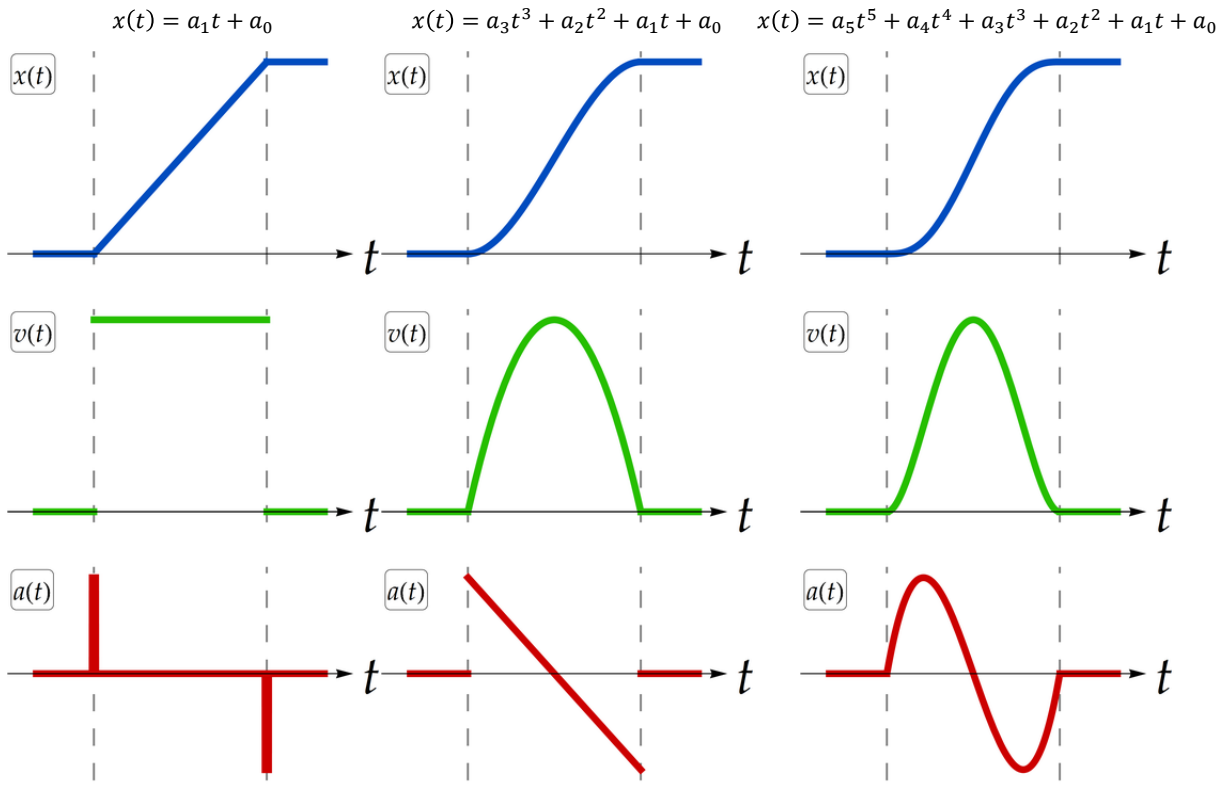
Winda – dźwig pionowy

Winda jest to maszyna, której zadaniem jest pionowe transportowanie ludzi bądź ładunków między piętrami (lub poziomami) w budynku, na statku czy innej konstrukcji. Najczęściej napędzane są one silnikami elektrycznymi, poprzez układy lin połączonych z wielokrążkami i systemem przeciwwagi. Innymi napędami stosowanymi w windach mogą być siłowniki hydrauliczne. Elementy składowe windy pokazano na rys. 1.



Rys. 1

Problematyka planowania trajektorii *„od punktu do punktu”* (ang. *point-to-point*)



Porównanie profili prędkości $v(t)$ i przyspieszenia $a(t)$ przejazdu od piętra do piętra dla trajektorii $x(t)$ danej: funkcją liniową, wielomianem trzeciego i piątego stopnia.

W większości rozwiązań przemieszczanie widny jest realizowane według wzorcowego profilu prędkości, jaki jest wymagany do ruchu windy od piętra do piętra.

Ze względu na fizyczne ograniczenia napędu oraz na komfort pasażerów trajektoria $x(t)$ przejazdu pomiędzy piętrami nie może być opisana funkcją liniową, ponieważ skutkuje to skokami prędkości $v(t)$ na początku i na końcu ruchu. Ciągłość prędkości można zapewnić opisując trajektorię wielomianem trzeciego stopnia, jednak w takim opisie występują skokowe zmiany wartości przyspieszenia $a(t)$, co wymagałoby skokowej zmiany wartości siły napędzającej. Aby uzyskać ciągły przebieg przyspieszania, trajektoria musi być opisana wielomianem co najmniej piątego stopnia.

Parametry trajektorii przejazdu windy a komfort pasażerów

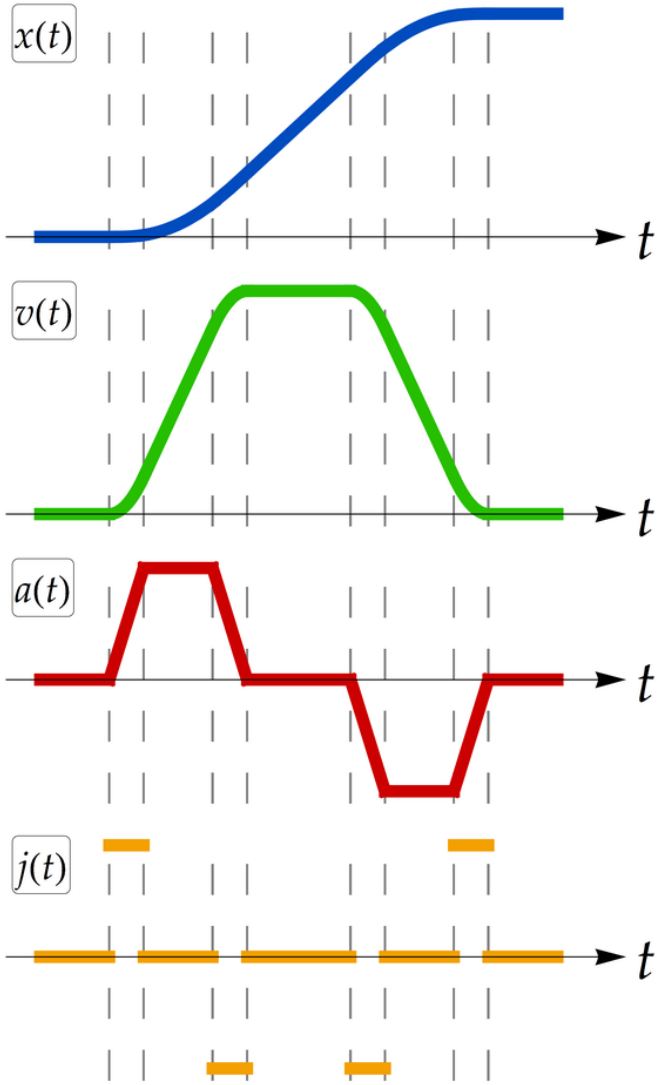
Na ocenę jakości jazdy windą składają się następujące parametry:

- czas przejazdu od zatrzymania do zatrzymania dla dystansu jednego piętra (ang. *floor-to-floor time),*
- błąd pozycjonowania przy zatrzymaniu,
- prędkość jazdy,
- przyspieszenie jazdy (maksymalna wartość nie powinna przekraczać $0{,}7\,\frac{m}{s^2}$),
- zryw* jazdy (maksymalna wartość nie powinna przekraczać $1{,}5\,\frac{m}{s^3}$),
- subiektywne odczucie płynność jazdy.

***Zryw** (lub szarpnięcie, ang. *jerk*) – jest miarą zmiany przyspieszenia w czasie. Obliczany jest jako pochodna przyspieszenia $j(t) = \frac{da}{dt} = \frac{d^3x}{dt^3}$. Zryw nie jest powszechnie używaną wielkością fizyczną, jednak znajduje zastosowanie w ocenie jakości jazdy windą, a sposób jego pomiaru oraz zalecane wartości opisuje norma ISO 8100-34:2021 *Lifts for the transport of persons and goods — Part 34: Measurement of lift ride quality.*

Na jakość jazdy wpływa nie tylko wartość zrywu, ale w ogóle jego występowanie. Profile prędkości dla wind opisuje się również przez procent czasu przejazdu w jakim zryw ma niezerową wartość. Zazwyczaj podczas przejazdu tylko jednego piętra od 50% do 80% czasu zryw ma niezerową wartość. Wysokie wartości zrywu są odczuwalne jako wstrząsy/uderzenia.

Trajektoria ruchu windy

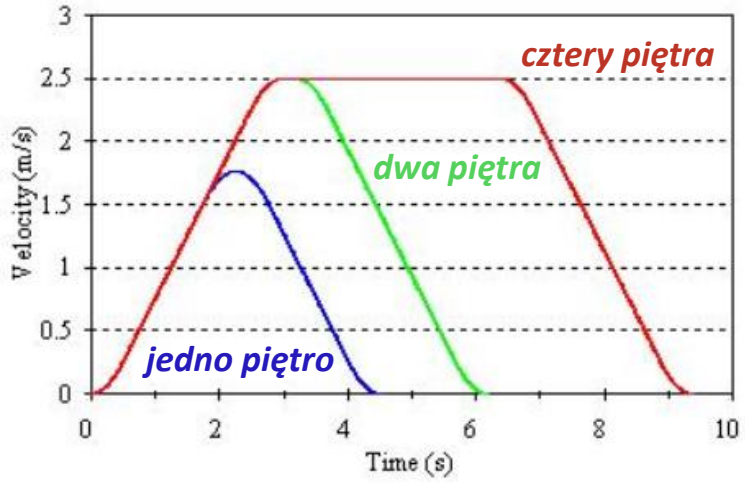


Aby łatwo kontrolować wartość zrywu, wzorcową trajektorię ruchu widny można opisać rozpoczynając od kawałkami ciągłej funkcji przyspieszenia, gdzie przyspieszenie liniowo rośnie/maleje lub ma wartość stałą.

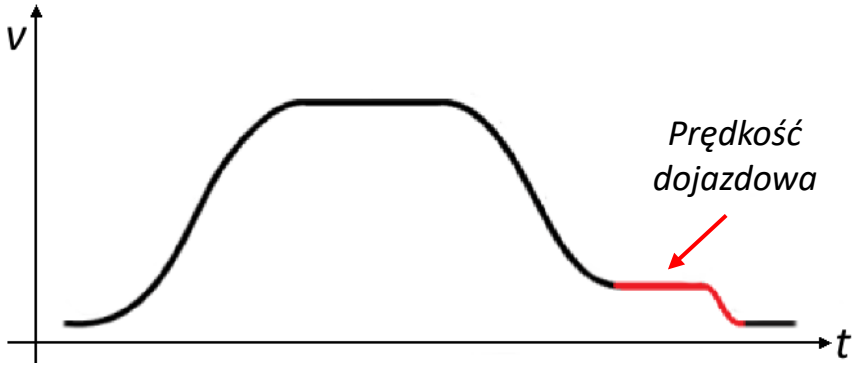
Możliwy wzorcowy profil trajektorii dla przejazdu od piętra do piętra przedstawiono na rysunku po lewej. Jak można zaobserwować, zryw $j(t)$ ma niezerową wartość w chwilach, w których przyspieszenie rośnie lub maleje.

Dobór parametrów trajektorii ruchu windy ogólnie nie jest zagadnieniem prostym. Przy jego projektowaniu i optymalizacji pod uwagę bierze się nie tylko komfort jazdy pasażerów, ale również czasy i ekonomię przejazdów, jak i możliwości zespołu napędowego – czasy osiągania maksymalnych prędkości/przyspieszeń. Szczegóły algorytmów planujących trajektorię są zazwyczaj objęte ochroną patentową i są tajemnicą handlową producentów sterowników.

Trajektoria ruchu windy



Dzięki współczesnym mikroprocesorowym sterownikom wzorcowy profil prędkości może się zmieniać w zależności od liczby pięter, które winda ma pokonać w jednym przejeździe. Dzięki czemu zapewniony jest szybki, komfortowy i efektywny transport.

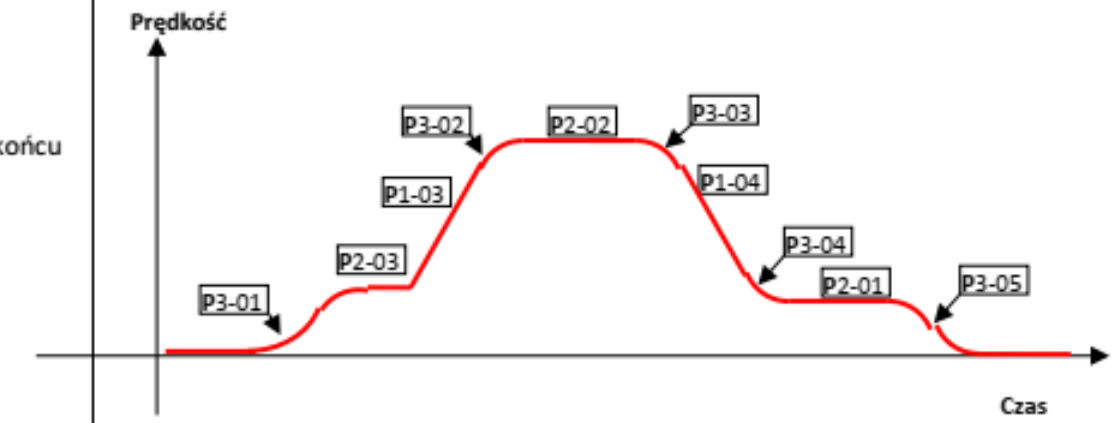


Sterowanie ruchem widny zazwyczaj odbywa się na podstawie pomiaru prędkości silnika zespołu napędowego. Informacja zwrotna o położeniu uzyskiwana jest natomiast z czujników krańcowych umieszczonych na wysokości poszczególnych pięter (oraz pomiędzy piętrami). Zazwyczaj to sygnał z czujnika krańcowego wymusza zaciśnięcie hamulca. Aby uniknąć dużego przeciążenia związanego z hamowaniem, powszechnie stosowaną techniką jest zmniejszenie prędkości widny w okolicy spodziewanego dojazdu do zadanego piętra.

Trajektoria ruchu windy

10.8. Rampy i prędkości robocze

Powiązane parametry	Czynność
P1-03 (czas rampy przyspieszenia) P1-04 (czas rampy zwalniania) P2-01 (prędkość dojazdowa) P2-02 (wysoka prędkość) P2-03 (prędkość pośrednia) P2-04 (prędkość inspekcyjna) P3-01 (charakterystyka szarpnięcia na początku przyspieszania) P3-02 (charakterystyka szarpnięcia na końcu przyspieszania) P3-03 (charakterystyka szarpnięcia na początku zwalniania) P3-04 (charakterystyka szarpnięcia na końcu zwalniania) P3-05 (szarpnięcie przy zatrzymaniu)	Zaprogramować pożądany czas rampy zgodnie ze schematem profilu poniżej.



Przykładowy, możliwy do zaprogramowania wzorcowy profil prędkości ruchu widny dla falownika do zastosowań windowo-dźwigowych Optidrive P2 Elevator firmy Invertek Drives

Pomiar przyspieszeń – akcelerometr

Akcelerometr to czujnik wykrywający zmiany przyspieszenia ciała, do którego został przymocowany. Główna idea działania i budowy akcelerometru została przedstawiona na rys. 2. Zbudowany jest on z dwóch części – ruchomej i nieruchomej. Część nieruchoma (baza) mocowana jest do badanego obiektu, natomiast część ruchoma o znanej masie M zawieszona jest w niej przy pomocy elementów sprężystych o znanej sztywności k. Na skutek zmian przyspieszenia tego obiektu, część czynna akcelerometru zaczyna się poruszać na skutek pojawienia się siły bezwładności. W zależności od rodzaju akcelerometru przyspieszenie ciała jest mierzone na podstawie przemieszczenia części ruchomej, jakiego doznała ona podczas ruchu badanego obiektu albo na podstawie zmierzonej bezpośrednio siły powstałej w elemencie sprężystym.

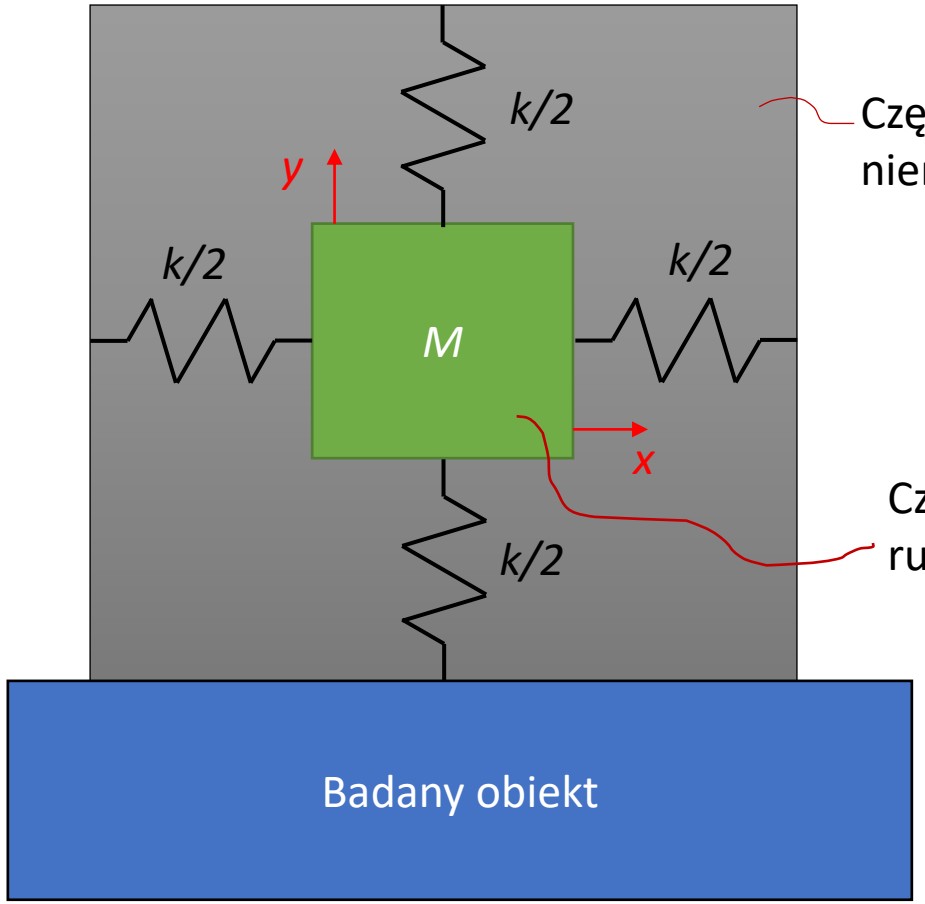
Przyspieszenia w kierunkach osi X, Y, Z

Przemieszczenie

Przyspieszenie

$a_x M = kx$ $\quad a_y M = ky$ $\quad a_z M = kz$

Siła w elemencie sprężystym

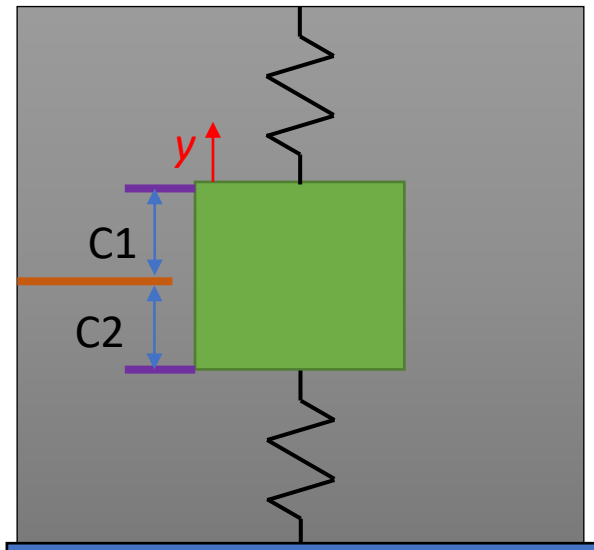


Rys. 2

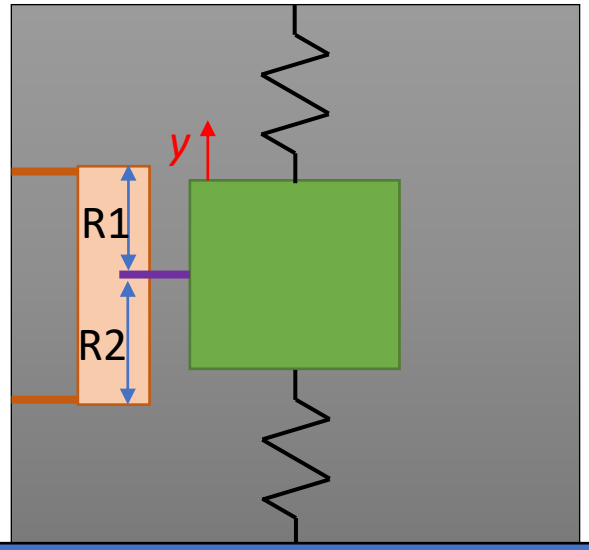
Pomiar przyspieszeń – akcelerometr

Pomiar przemieszczenie (x, y, z) drgającej masy może być realizowany poprzez pomiar zmiany pojemności *C1* i *C2* między okładzinami kondensatora (rys. 2a), bądź rezystancji *R1* i *R2* potencjometru (rys. 2b).
Pomiar siły bezwładności może odbywać się przy pomocy elementów piezoelektrycznych (rys. 2c) bądź tensometrycznych (rys. 2d), które pełnią rolę elementów sprężystych. Piezoelektryki poddane działaniu siły zewnętrznej generują na swojej powierzchniach napięcie, które jest zależne co do wartości od tej siły. Natomiast tensometry zmieniają rezystancję w zależności od siły, która na nie działa.

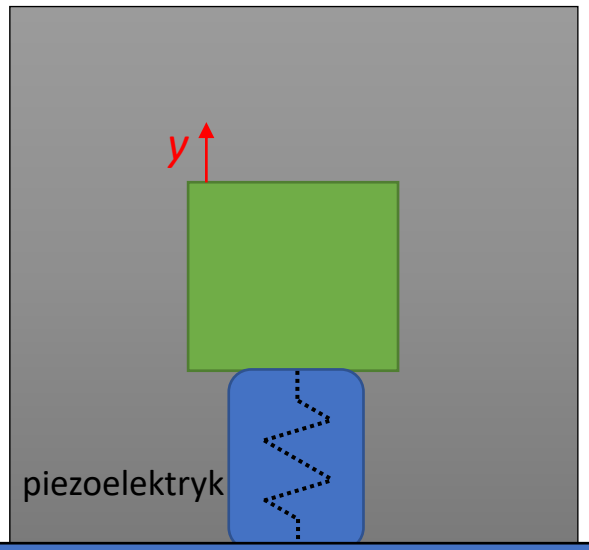
Rys. 3a



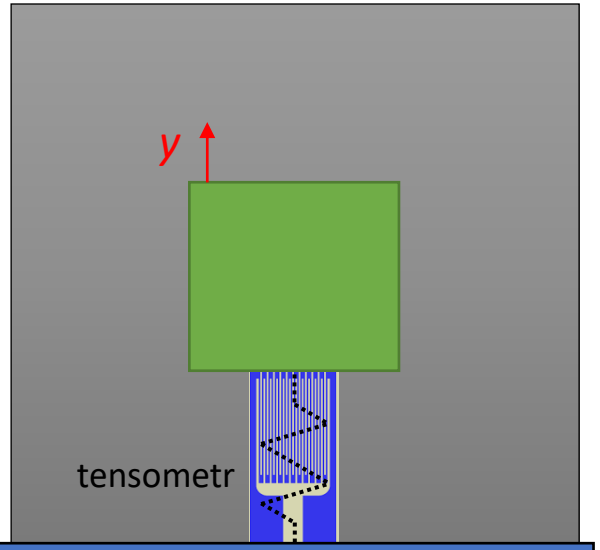
Rys. 3b

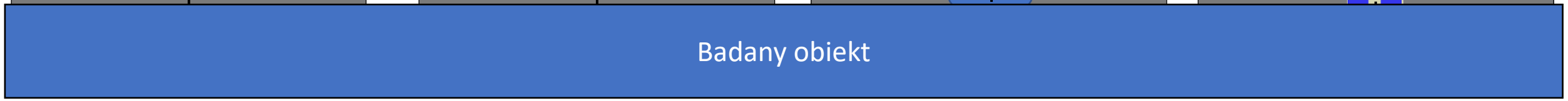


Rys. 3c



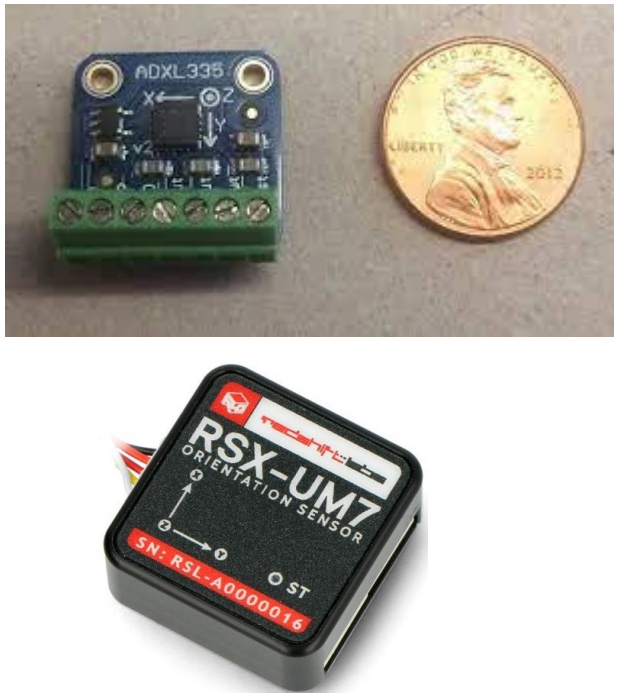
Rys. 3d





Pomiar przyspieszeń – akcelerometr

Przykładowe akcelerometry zostały pokazane na rys. 4. Akcelerometr wykonane w technologii MEMS rys.4a i akcelerometry w obudowach przemysłowych rys.4b.

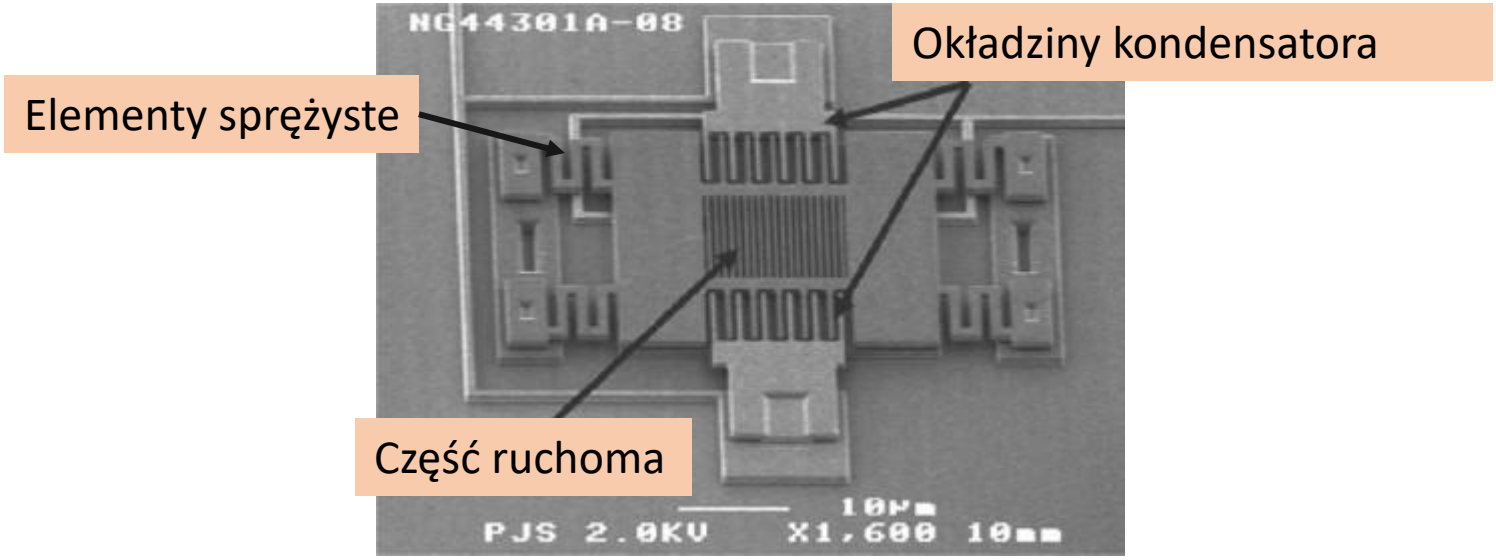


Rys. 4a

Rys. 4b

Pomiar przyspieszeń – akcelerometr

Akcelerometry MEMS charakteryzują się bardzo małymi wymiarami zewnętrznymi. Z tego też względu często wykorzystywane są w urządzeniach mobilnych takich jak np. telefony komórkowe. Wnętrze takiego akcelerometru pokazano na rys. 5.

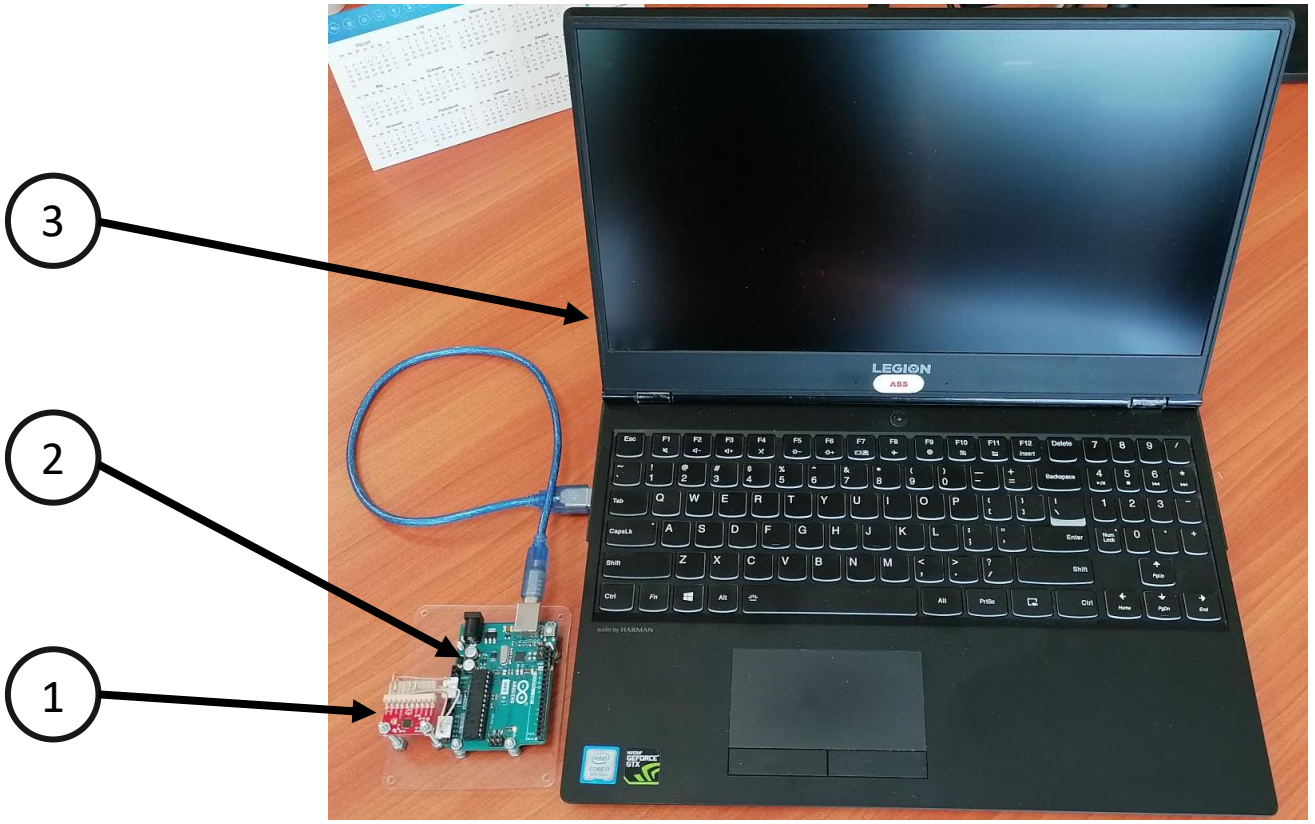


Rys. 5

Układ pomiarowy wykorzystywany w ćwiczeniu

Układ pomiarowy przyspieszeń wykorzystywany w ćwiczeniu składa się z 3 podstawowych elementów (rys. 6):

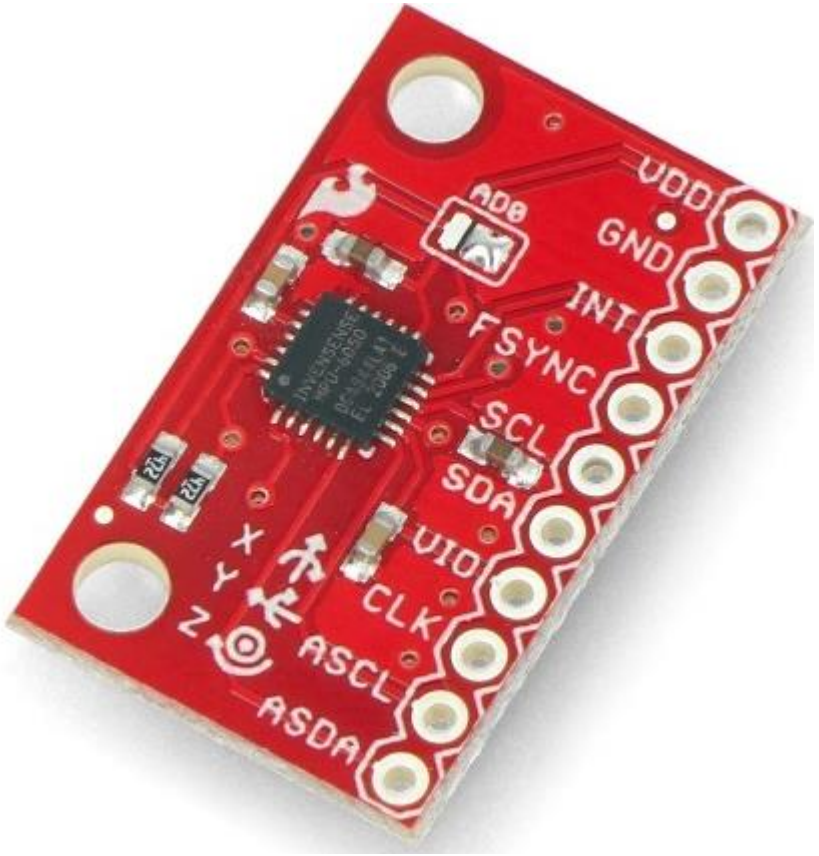
1. Układu elektronicznego *MPU6050*
2. Układu mikroprocesorowego – mikrokontroler *Arduino UNO*
3. Notebooka z programem do odczytu danych z Arduino.



Rys. 6

Układ MPU6050

Układ elektroniczny MPU6050 (rys. 7) firmy SparkFun zawiera w sobie 3-osiowy akcelerometr MEMS, a także 3-osiowy żyroskop MEMS (urządzenie mierzące prędkość kątową) i termometr cyfrowy. Jest on więc układem przeznaczonym przede wszystkim do wykonywania pomiarów ruchu. Dodatkowo posiada on wbudowaną jednostkę DMP (Digital Motion Processor), której zadaniem jest przetwarzanie zbieranych przez czujnik danych na konkretne położenie względem ziemi przy wykorzystaniu algorytmów MotionFusion. Zadaniem tej jednostki jest odciążenie mikrokontrolera Arduino poprzez przejęcie części obliczeń. Moduł zasilany jest napięciem 2.3-3.4 V, a jego pobór prądu podczas normalnej pracy wynosi ok. 5mA. Komunikacja pomiędzy układem a mikrokontrolerem odbywa się przez interfejs cyfrowy I2C (wyjścia SCL i SDA). Pomiary przyspieszeń w kierunku każdej osi przetwarzane są przez 16 bitowe przetwornik. Zarówno akcelerometr jak i żyroskop mogą pracować w różnych zakresach przyspieszeń i prędkości obrotu. Dla akcelerometru wynoszą one: ±2g, ±4g, ±8g, ±16g (1g to przyspieszenie równe wartości jednokrotności przyspieszenia ziemskiego 9.81 m/s^2), a żyroskopu: ±250, ±500, ±1000, ±2000 °/s. Wybór zakresu odbywa się na etapie programowania mikrokontrolera.



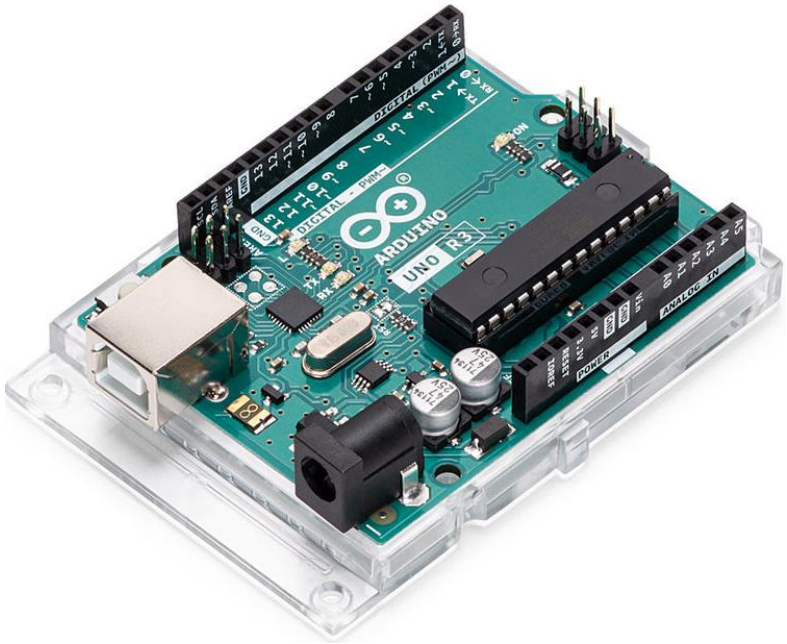
Rys. 7

Arduino UNO

Arduino Uno (rys. 8) to płytka oparta na mikrokontrolerze ATmega328P. Posiada 14 pinów cyfrowych wejść/wyjść (z czego 6 może być używanych jako wyjścia PWM), 6 wejść analogowych, rezonator ceramiczny 16 MHz (CSTCE16M0V53-R0), złącze USB, gniazdo zasilania, złącze ICSP i przycisk resetowania . Zawiera wszystko, co jest potrzebne do obsługi mikrokontrolera. Do zapoczątkowania pracy układu wystarczy po prostu podłączyć go do komputera za pomocą kabla USB, lub dostarczyć mu zasilanie z baterii bądź zasilacza AC/DC. Arduino Uno to moduł z mikrokontrolerem AVR ATmega328P – posiada 32 kB pamięci Flash, 2 kB pamięci operacyjnej RAM.

Arduino jest jednym z najpopularniejszych minikomputerów na świecie. Został on stworzony z myślą o osobach początkujących chcących rozwijać swoje zainteresowania w dziedzinach elektroniki i programowania.

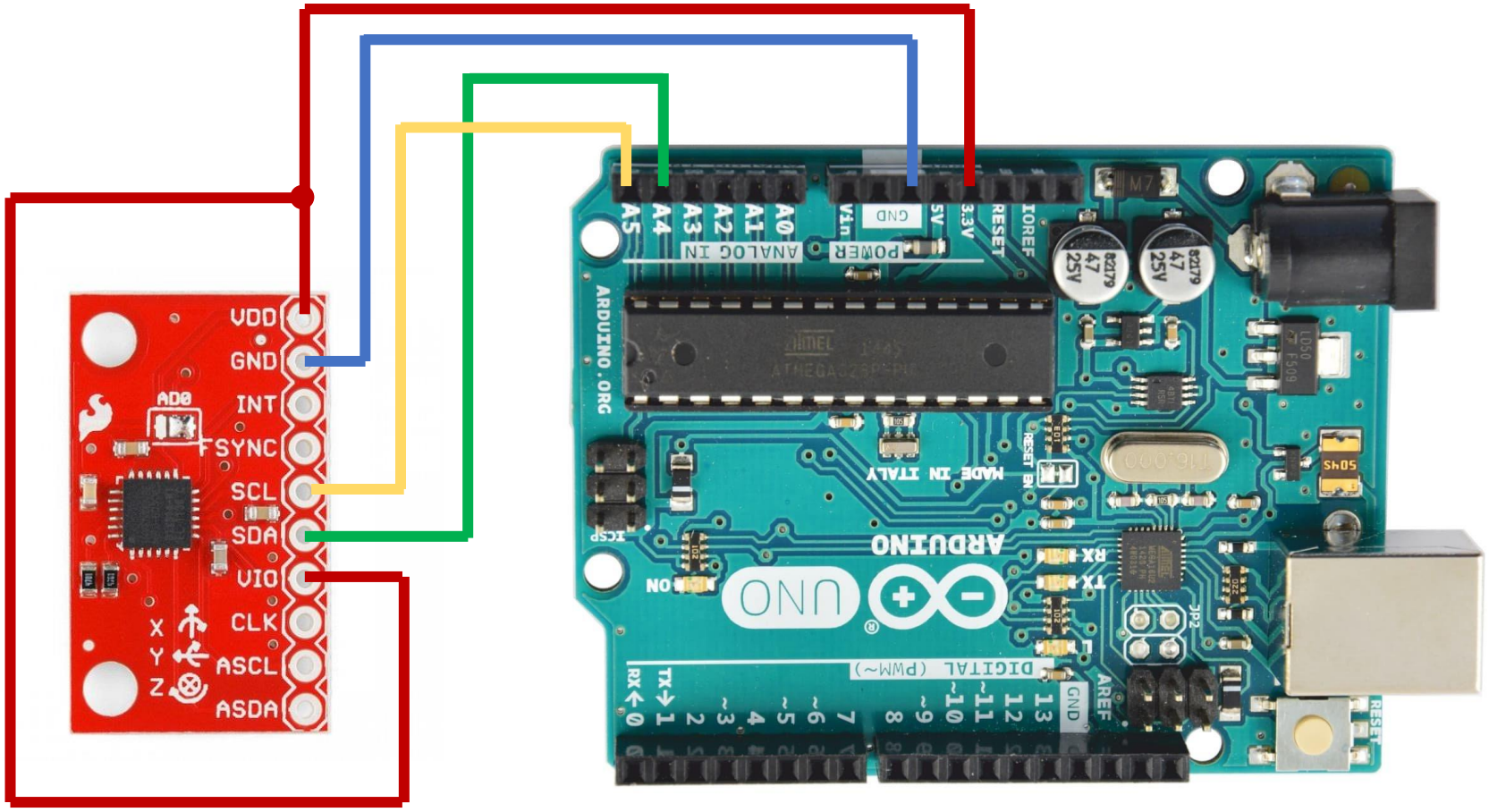
Dużym atutem płytek Arduino jest dedykowane do niego środowisko softwareowe wykorzystujące język Arduino. Język ten bazuje na składni języka C. Środowisko nazywane Arduino IDE posiada podstawowe narzędzia takie jak kompilator, konsolę błędów czy monitor portu szeregowego (tekstowy i graficzny). Ponadto do pamięci FLASH mikrokontrolera wgrano bootloader, tzn. program rozruchowy, dzięki któremu w celu rozpoczęcia komunikacji z komputerem wystarczy przewód USB, bez konieczności podłączania zewnętrznego programatora, tak jak ma to miejsce w przypadku programowania standardowych mikrokontrolerów AVR w językach C, BASCOM czy Assembler.



Rys. 8

Połączenie układ MPU6050 z Arduino UNO

Połączenie między MPU6050 a Arduino odbywa się przez port szeregowy I2C. Linia zegarowa SCL podpięta jest do pinu A5, a linia danych SDA do pinu A4. Ponadto MPU6050 podłączone jest do pinu zasilania 3.3 V dostępnego na płytce Arduino. Do poprawnego działania MPU6050 producenta SparkFun konieczne jest podciągnięcie zasilania 3.3V do pinu VIO opowiadającego za wartość napięcia dla wysokiego stanu logicznego w module. Opisane połączenia pokazano na rys. 9.



Rys. 9

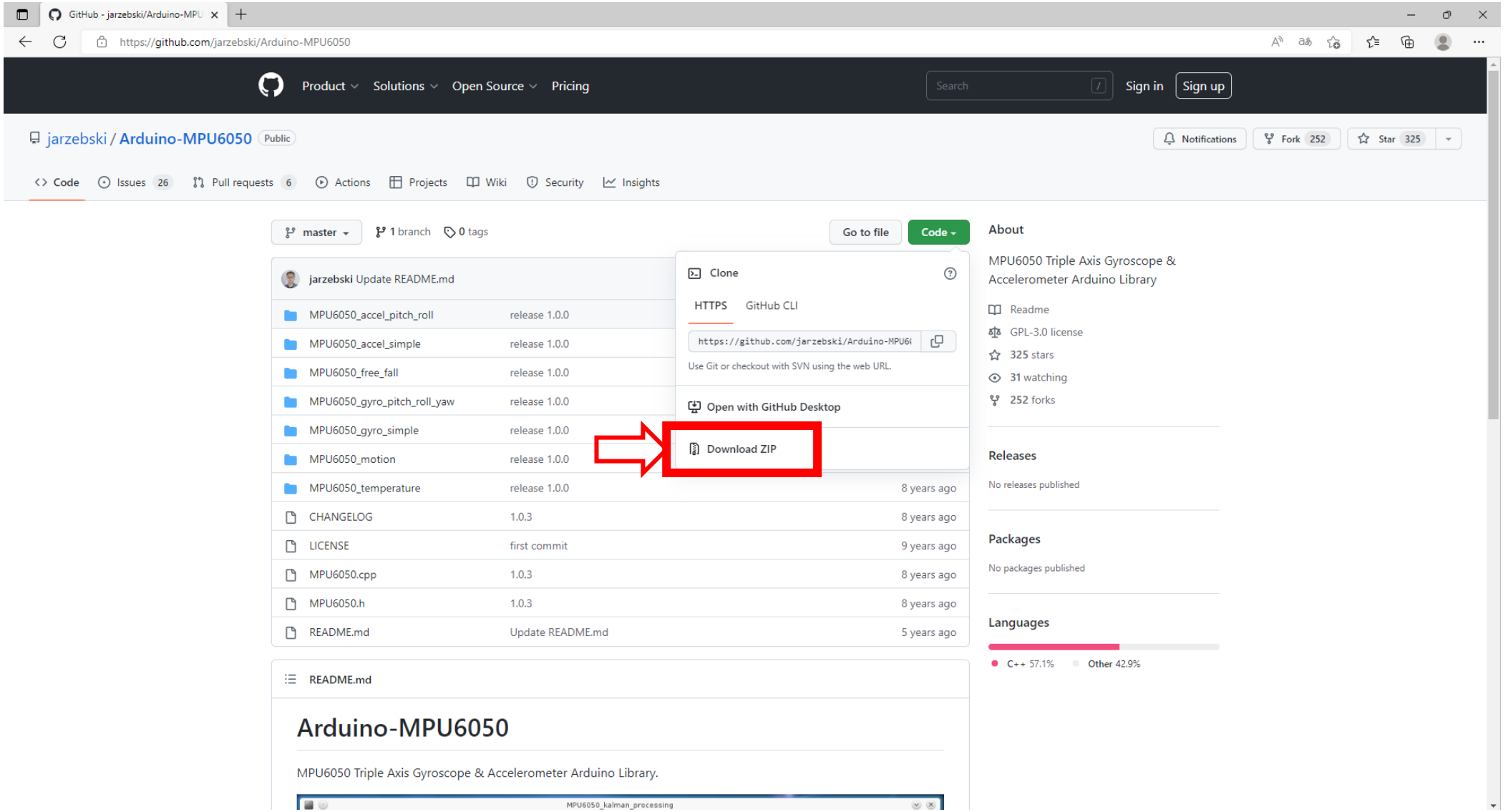
Program do pomiaru przyspieszeń

Zmontowany układ Arduino UNO i modułu MPU6050 należy podłączyć przy pomocy przewodu USB do laptopa z zainstalowanym środowiskiem Arduino IDE (można je pobrać ze strony https://www.arduino.cc/en/software).
W następnym kroku należy przygotować program, który będzie odczytywał dane pomiarowe z akcelerometru. Ponieważ moduł MPU6050 jest popularnym układem wykorzystywanym do pomiaru przyspieszeń (lub prędkości kątowych) istnieje do niego wiele gotowych programów wykorzystujących funkcje ułatwiające ich obsługę. Funkcje te nazywają się bibliotekami, które dołącza się do kodu programu (poprzez komendę #include). Jedna z takich bibliotek, która obsługuje moduł MPU6050 została przygotowana przez Korneliusza Jarzębskiego. Opracowany przez niego kod programu, odczytuje dane mierzone prze czujnik w czasie rzeczywistym.

Zanim przejdziemy do analizy tego programu musimy uruchomić program Arduino IDE i dokonać pewnych ustawień.

Pobieramy bibliotekę modułu MPU6050 w formacie .ZIP ze strony https://github.com/jarzebski/Arduino-MPU6050 (rys. 10) i wgrywamy ją w Arduino IDE poprzez zakładki: *Sketch->Include Library->Add .ZIP Library…->wybieramy plik biblioteki*.
Pliki wgranych bibliotek są dostępne w folderze domyślnym *C:\Users\Użytkownik\Dokumenty\Arduino\libraries*.

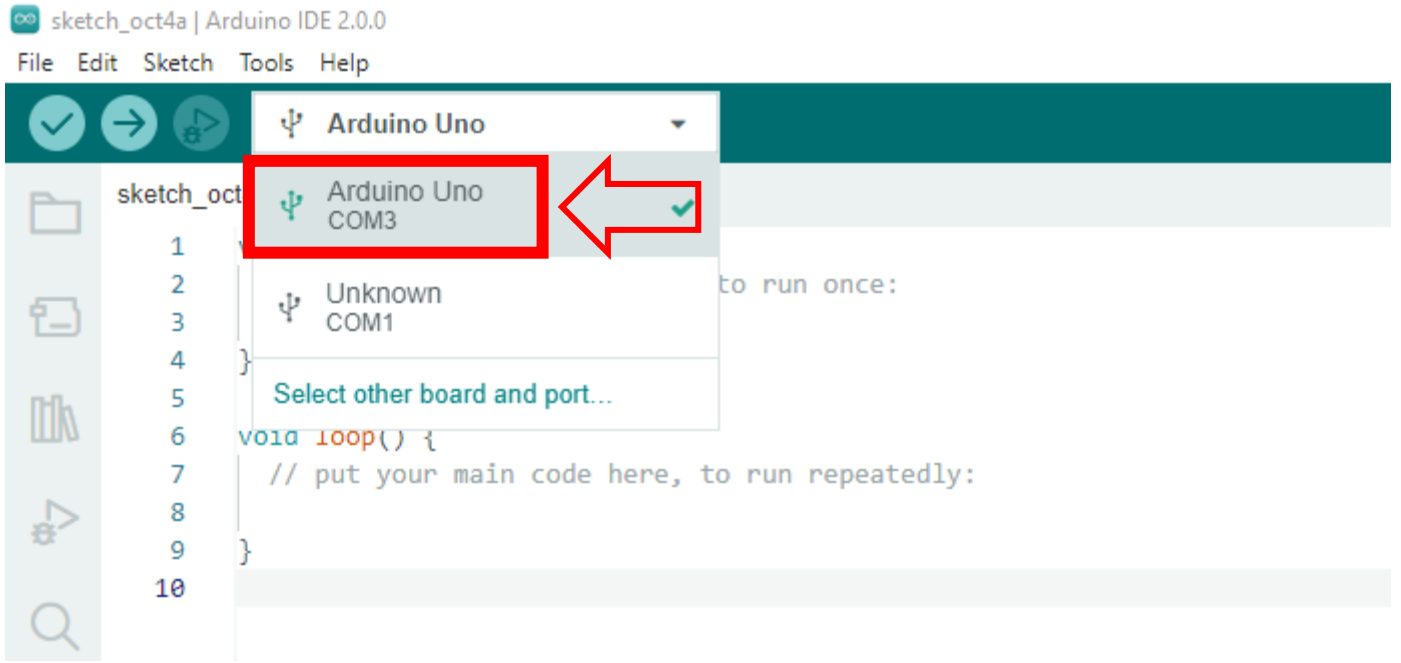
Program do pomiaru przyspieszeń – biblioteka MPU6050



Rys. 10

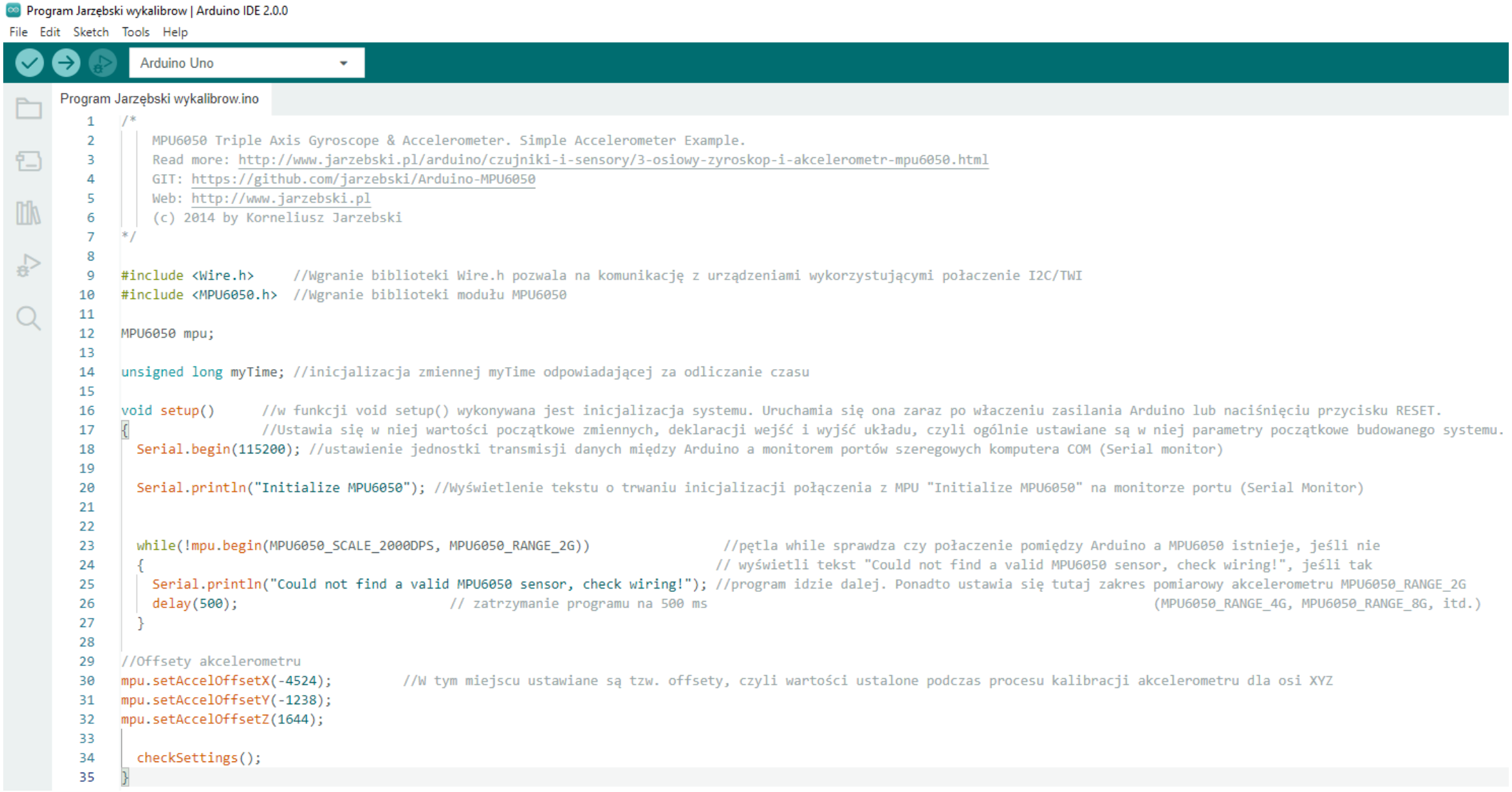
Program do pomiaru przyspieszeń

Następnie w programie Arduino IDE należy wybrać port do którego podłączona jest nasza płytka Arduino, patrz rys. 11. Przeanalizujmy więc ważniejsze części jego programu pokazanego na rys. 12a,b,c.



Rys. 11

Analiza ważniejszych linii kodu programu




```
/*
    MPU6050 Triple Axis Gyroscope & Accelerometer. Simple Accelerometer Example.
    Read more: http://www.jarzebski.pl/arduino/czujniki-i-sensory/3-osiowy-zyroskop-i-akcelerometr-mpu6050.html
    GIT: https://github.com/jarzebski/Arduino-MPU6050
    Web: http://www.jarzebski.pl
    (c) 2014 by Korneliusz Jarzebski
*/

#include <Wire.h>     //Wgranie biblioteki Wire.h pozwala na komunikację z urządzeniami wykorzystującymi połaczenie I2C/TWI
#include <MPU6050.h>  //Wgranie biblioteki modułu MPU6050

MPU6050 mpu;

unsigned long myTime; //inicjalizacja zmiennej myTime odpowiadającej za odliczanie czasu

void setup()      //w funkcji void setup() wykonywana jest inicjalizacja systemu. Uruchamia się ona zaraz po właczeniu zasilania Arduino lub naciśnięciu przycisku RESET.
{                 //Ustawia się w niej wartości początkowe zmiennych, deklaracji wejść i wyjść układu, czyli ogólnie ustawiane są w niej parametry początkowe budowanego systemu.
  Serial.begin(115200); //ustawienie jednostki transmisji danych między Arduino a monitorem portów szeregowych komputera COM (Serial monitor)

  Serial.println("Initialize MPU6050"); //Wyświetlenie tekstu o trwaniu inicjalizacji połączenia z MPU "Initialize MPU6050" na monitorze portu (Serial Monitor)


  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))                //pętla while sprawdza czy połaczenie pomiędzy Arduino a MPU6050 istnieje, jeśli nie
  {                                                                         // wyświetli tekst "Could not find a valid MPU6050 sensor, check wiring!", jeśli tak
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!"); //program idzie dalej. Ponadto ustawia się tutaj zakres pomiarowy akcelerometru MPU6050_RANGE_2G
    delay(500);                           // zatrzymanie programu na 500 ms                                                (MPU6050_RANGE_4G, MPU6050_RANGE_8G, itd.)
  }

//Offsety akcelerometru
mpu.setAccelOffsetX(-4524);         //W tym miejscu ustawiane są tzw. offsety, czyli wartości ustalone podczas procesu kalibracji akcelerometru dla osi XYZ
mpu.setAccelOffsetY(-1238);
mpu.setAccelOffsetZ(1644);

  checkSettings();
}
```

Rys. 12a

Analiza ważniejszych linii kodu programu

```
36
37  void checkSettings()        //funkcja void CheckSettings() ma za zadanie wyświetlic informacje o obecnych ustawieniach akcelerometru i wykonuje się tylko raz
38  {
39    Serial.println();                                                    // wyświetli pustą linijkę w Serial Monitor i przechozi do następnego wiersza kodu
40
41    Serial.print(" * Accelerometer:         ");                         // wyświetla w Serial Monitor wybrany zakres pomiarowy akcelerometru (2g, 4g, 8g, itd.):
42    switch(mpu.getRange())                                               //* Accelerometer:         +/- 2 g
43    {
44      case MPU6050_RANGE_16G:            Serial.println("+/- 16 g"); break;
45      case MPU6050_RANGE_8G:             Serial.println("+/- 8 g"); break;
46      case MPU6050_RANGE_4G:             Serial.println("+/- 4 g"); break;
47      case MPU6050_RANGE_2G:             Serial.println("+/- 2 g"); break;
48    }
49
50    Serial.print(" * Accelerometer offsets: ");                          // wyświetla ustawione dla akcelerometru offsety:
51    Serial.print(mpu.getAccelOffsetX());                                 //* Accelerometer offsets: -4524 / -1238 / 1644
52    Serial.print(" / ");
53    Serial.print(mpu.getAccelOffsetY());
54    Serial.print(" / ");
55    Serial.println(mpu.getAccelOffsetZ());
56
57    Serial.println();                                                    // wyświetli pustą linijkę w Serial Monitor
58
59    Serial.print("Time[ms]");                                          // wyświetli linijkę w Serial Monitor: Time[ms] aX[m/s^2] aY[m/s^2] aZ[m/s^2]
60    Serial.print(" ");
61    Serial.print("aX[m/s^2]");
62    Serial.print(" ");
63    Serial.print("aY[m/s^2]");
64    Serial.print(" ");
65    Serial.print("aZ[m/s^2]");
66    Serial.println();
67  }
```

Rys. 12b

Analiza ważniejszych linii kodu programu

```
69  void loop()       //void loop() jest funkcją nieskończonej pętli programu. Wykonywana jest aż do do momentu przerwania np. przyciskiem RESET, po czym zaczyna wykonywać się od nowa.
70  {
71    Vector normAccel = mpu.readNormalizeAccel(); //w tym miejscu tworzony jest wektor (tablica) składający się z przetworzonych wartości przyspieszeń pobranych z akcelerometru.
72
73
74    myTime = millis();    // przypisanie zmiennej myTime wartości czasu millis() w milisekundach, który liczony jest od momentu uruchomienia programu. Po zresetowaniu Arduino, liczenie zaczyna się od nowa
75    Serial.print(myTime); // wyświetlenie wartości czasu
76    Serial.print(" ");  // wyświetlenie  spacji jako przerwy
77
78
79    //accelerations print
80    Serial.print(float(normAccel.XAxis)); // wyświetlenie wartości przetworzonego przyspieszenia w kierunku osi X
81    Serial.print(" ");                    // wyświetlenie  spacji jako przerwy
82    Serial.print(float(normAccel.YAxis)); // wyświetlenie wartości przetworzonego przyspieszenia w kierunku osi Y
83    Serial.print(" ");                    // wyświetlenie  spacji jako przerwy
84    Serial.print(float(normAccel.ZAxis)); // wyświetlenie wartości przetworzonego przyspieszenia w kierunku osi Z
85    Serial.println();                     // wyświetlenie  spacji jako przerwy i przejście do następnego wiersza
86
87    delay(10);                            // zatrzymanie programu na 10 ms
88  }
89
90
91
```

Output

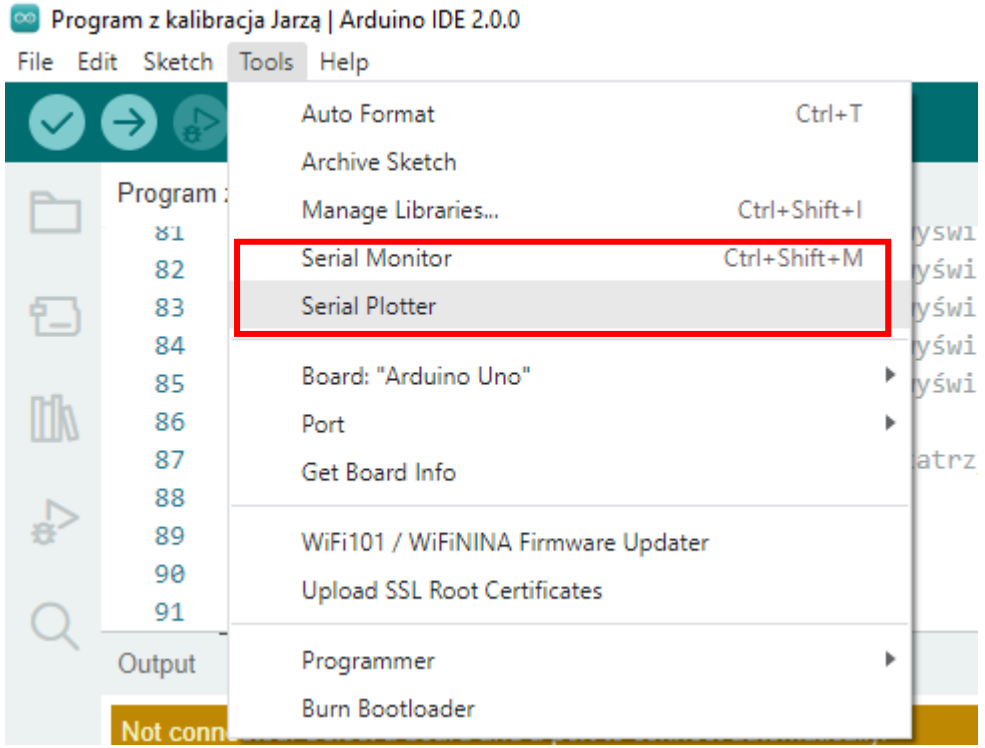
Rys. 12c

Serial Monitor i Serial Serial Plotter

W zakładce Tools programu Arduino IDE (rys. 13) można znaleźć dwie pozycje Serial Monitor i Serial Plotter.

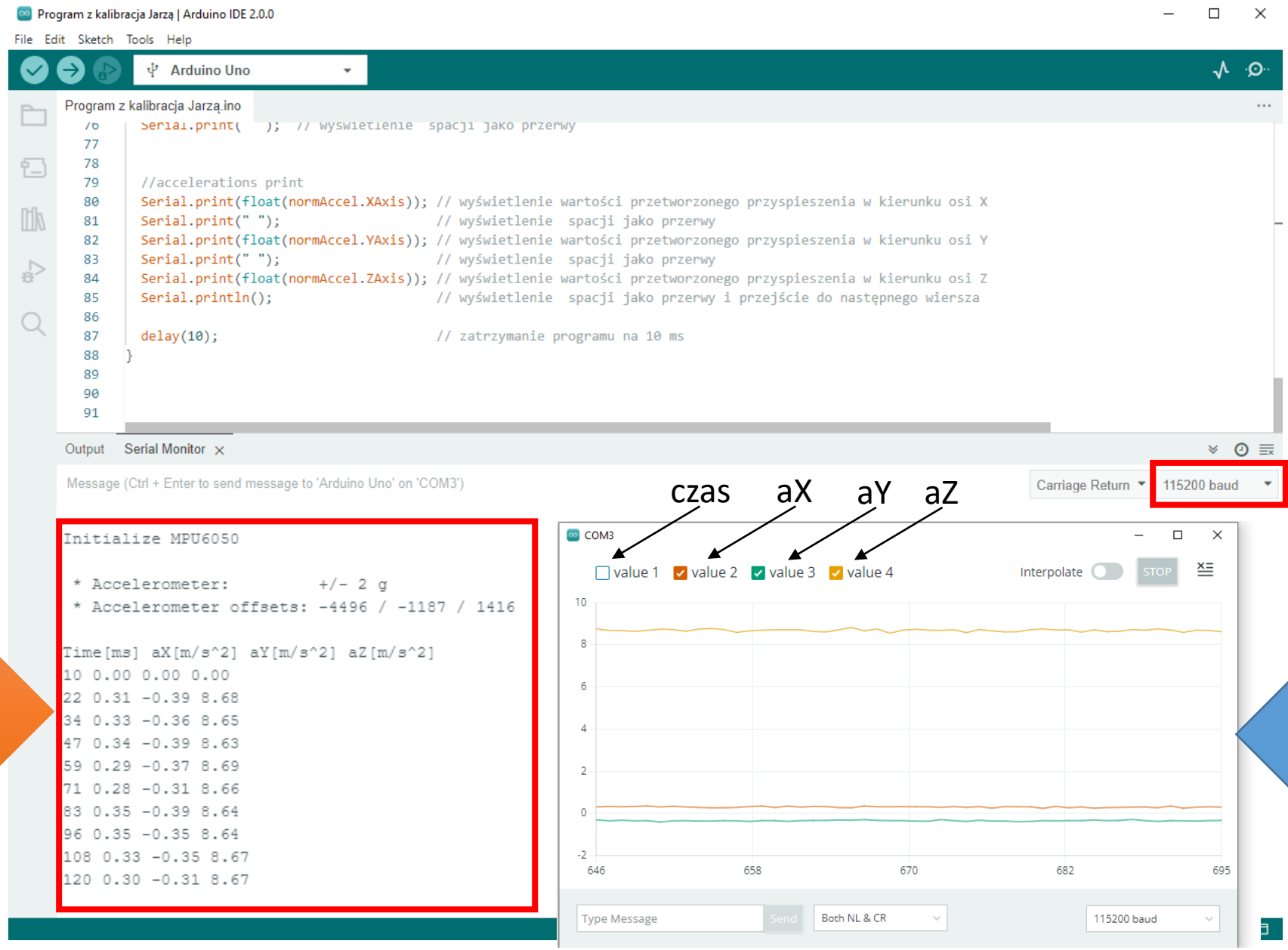
Serial Monitor (rys. 14) jest to narzędzie w postaci konsoli pokazującej informacje, wymieniane pomiędzy komputerem a Arduino. Jednym słowem to tu wyświetlają się wartości które w programie zostały ujęte w komendach Serial.print() i Serial.println().

Serial Plotter (rys. 14) jest natomiast narzędziem pozwalającym na śledzenie danych wysyłanych przez Arduino do komputera w postaci wykresu.



Rys. 13

Serial Monitor i Serial Serial Plotter



Tu należy ustawić tę samą wartość transmisji danych co w programie

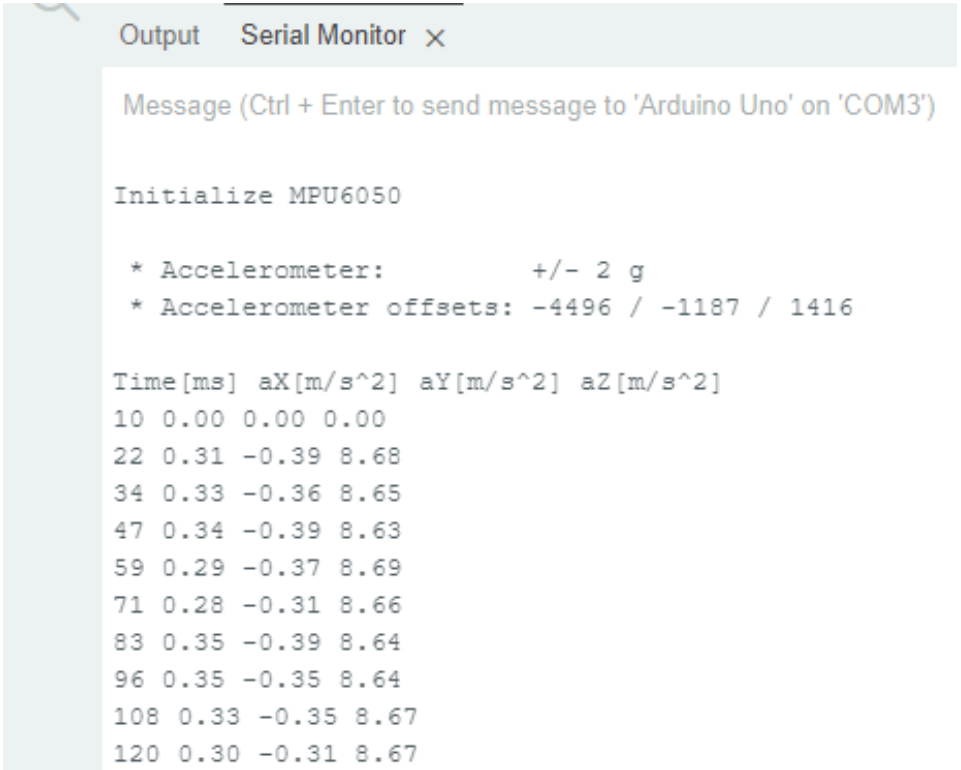
Informacje wyświetlane w serial monitor

Serial Plotter

Rys. 14

Kalibracja akcelerometru

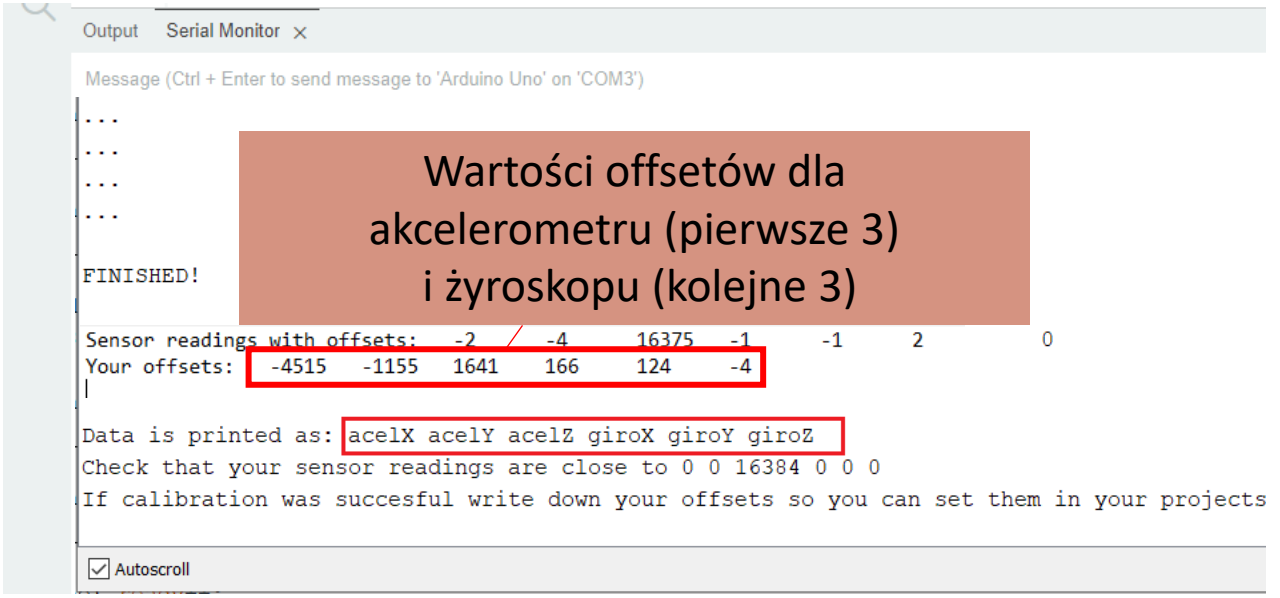
Ponieważ montując akcelerometr nie możemy mieć pewności, że jest on ustawiony idealnie w poziomie należy dokonać jego kalibracji. Bez kalibracji akcelerometru może okazać się, że będzie on wskazywał istnienie przyspieszeń w 3 osiach pomimo, że użytkownik zamontuje go w poziomie (przyjętym według swojego uznania). Przykład wskazań niewykalibrowanego akcelerometru leżącego na „wypoziomowanej” podłodze pokazano na rys. 15. Widocznym jest istnienie niezerowych wartości w osiach X i Y oraz wartości przyspieszenia w kierunku osi Z różnej od przyjętej 9.81 m/s².



Rys. 15

Kalibracja akcelerometru

Kalibracja polega na wyzerowaniu przyspieszeń w kierunkach osi X i Y oraz ustawieniu wartości przyspieszenia ziemskiego w kierunku osi Z. Ustawienie tych wartości odbywa się poprzez ustawienie odpowiednich offsetów w kodzie programu. Offsety są to wartości dodawane bądź odejmowane od wartości rejestrów wynikowych czujnika, w skrócie są to jakieś konkretne stałe wartości przyspieszeń, ale jeszcze nie przeliczone na jednostkę m/s^2. Do znalezienia wartości offsetów wykorzystuje się gotowe programy. Jeden z takich programów wraz z bibliotekami przygotował Jeff Rowberg. Po położeniu akcelerometru na płaskiej powierzchni przyjętej za poziomą uruchamia się taki program i odczekuje chwilę, aż kalibracja się skończy. Program na podstawie obliczonych średnich wartości z odczytów czujnika przetwarza je na odpowiednie offsety i podaje ich wartości w terminalu Serial Monitor (rys. 16a). W trakcie trwania kalibracji nie można poruszać akcelerometrem. Następnie offsety te wpisuje się do programu Korneliusza Jarzębskiego, rys. 16a.



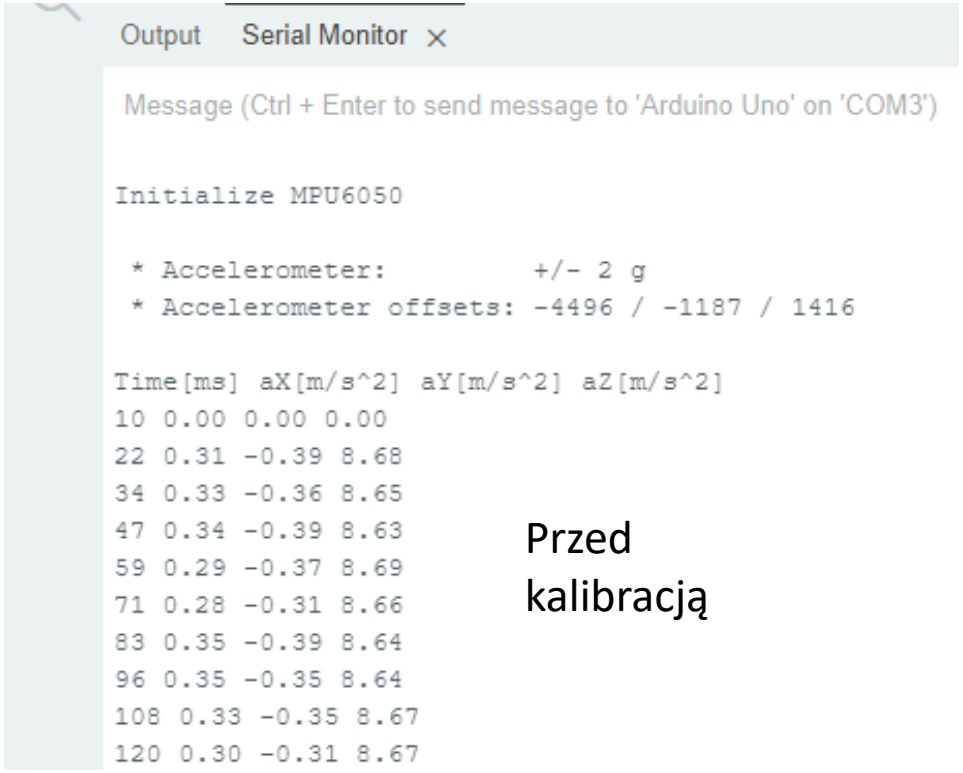
Rys. 16a

```
29  //Offsety akcelerometru
30  mpu.setAccelOffsetX(-4515);
31  mpu.setAccelOffsetY(-1155);
32  mpu.setAccelOffsetZ(1641);
```

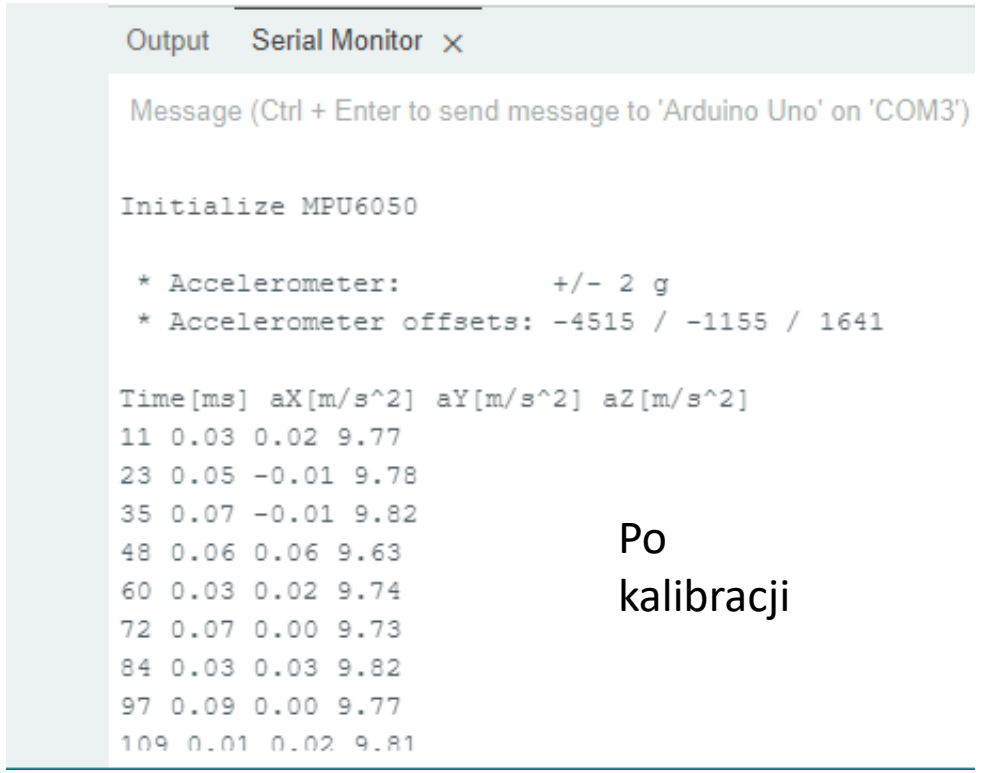
Rys. 16b

Kalibracja akcelerometru

Po programowej kalibracji akcelerometru wartości przyspieszeń wysyłane przez Arduino w kierunkach osi X i Y są w przybliżeniu równe 0, a w kierunku osi Z przyspieszeniu ziemskiemu 9.81 m/s² (rys. 17). Należy pamiętać, że wymagane offsety mogą się znacznie różnić w zależności od modułu MPU6050, dlatego dla każdego z nich należy wykonać osobną kalibrację. Jednakże dla danego modułu kalibrację wystarczy wykonać tylko raz. W przypadku tego ćwiczenia proces kalibracji został pominięty, a gotowe wartości offsetów wpisane do kodu programu.



Przed kalibracją

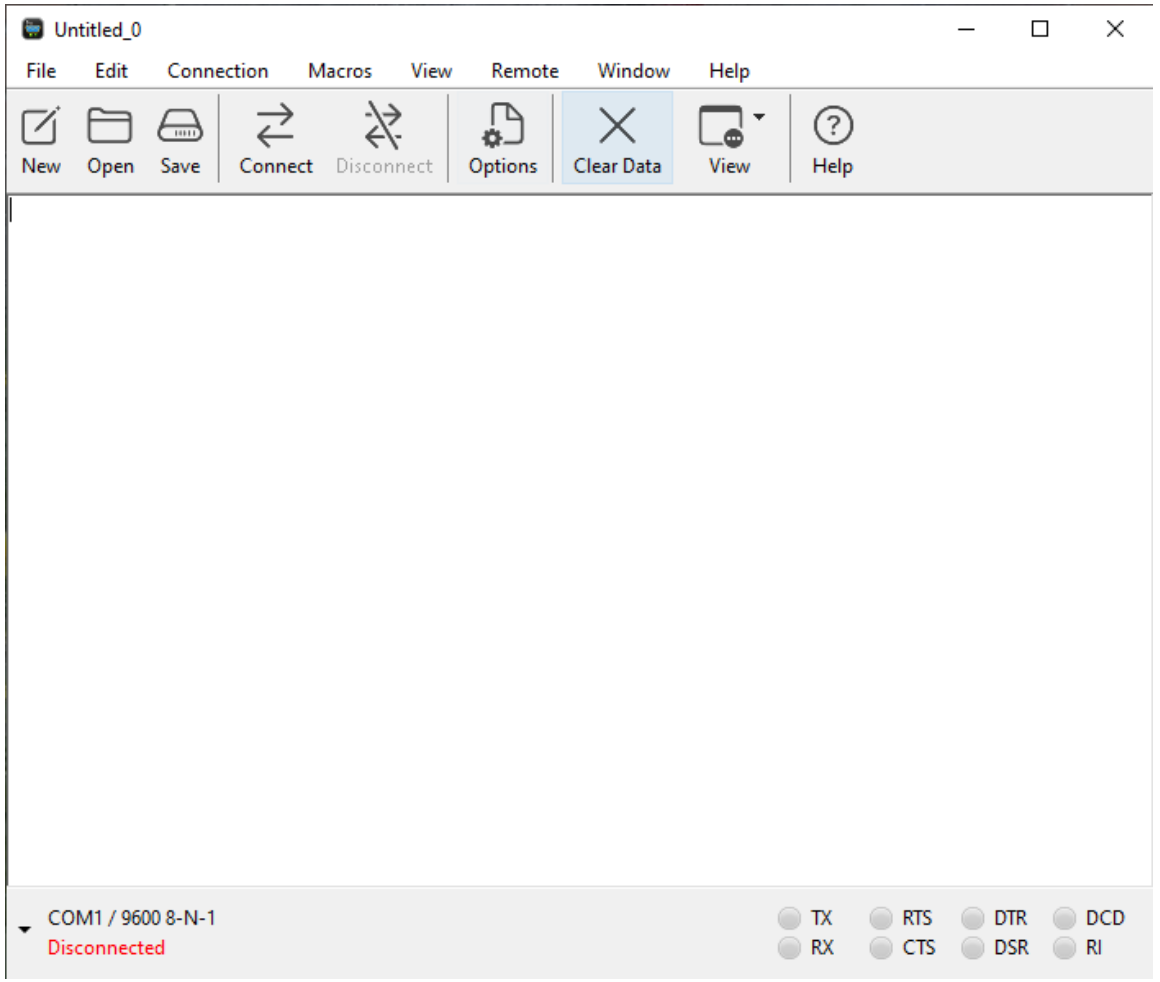


Po kalibracji

Rys. 17

Program CoolTerm

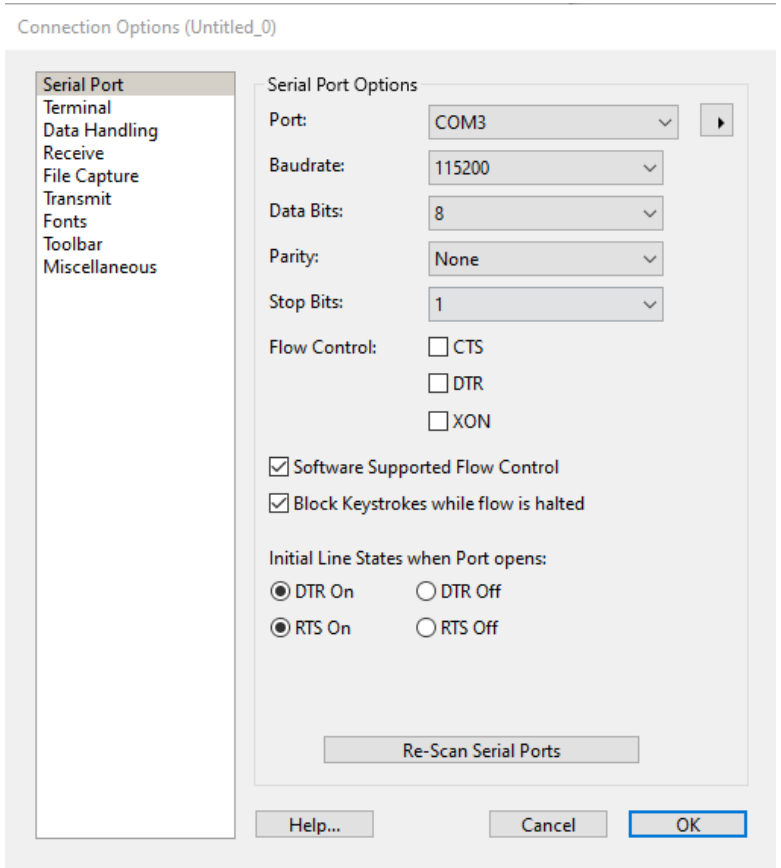
Jak już zostało pokazane w wcześniej najprostszym sposobem odczytu danych wysyłanych z Arduino UNO do komputera jest terminal Serial Monitor wbudowany w program Arduino IDE. Pewną wadą tego terminala jest to, że nie da się w sposób automatyczny zapisać wyświetlanych w nim danych w pliku. Jedynie co to możemy je zaznaczyć i skopiować do notatnika.
Większe możliwości odczytu i zapisu danych wysyłanych przez port szeregowy COM daje darmowy program CoolTerm (http://freeware.the-meiers.org/). Programu tego używa się zamiast arduinowskiego Serial Monitor, oba programy nie mogą pracować jednocześnie ze względu na konflikt interesów (i jeden i drugi chce odczytywać dane z tego samego portu COM). Po zainstalowaniu programu uruchamia nam się okienko widoczne na rys. 18.



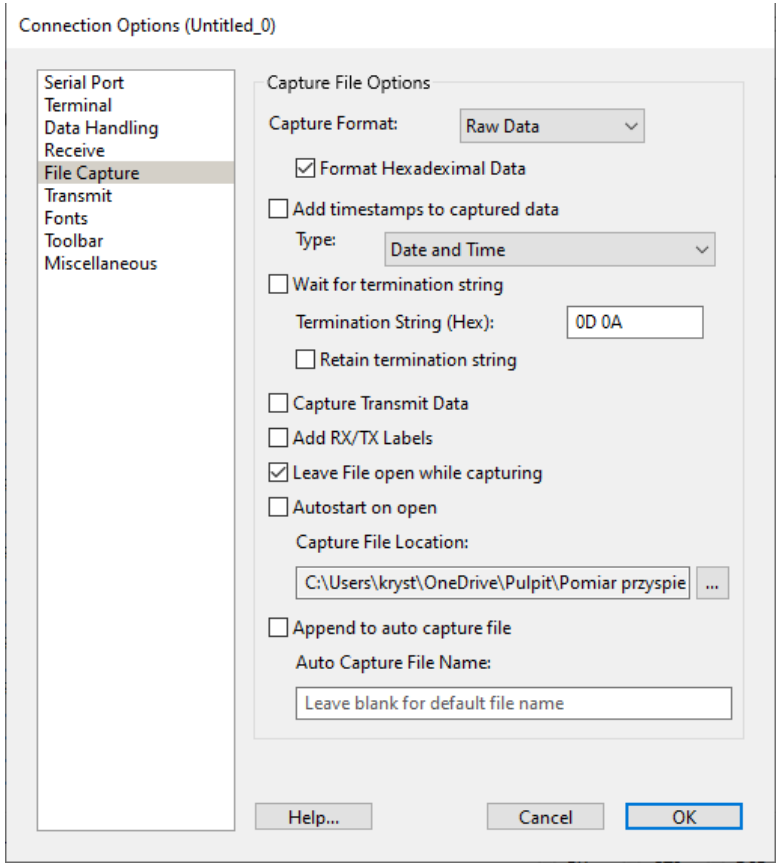
Rys. 18

Program CoolTerm

Teraz należy dokonać pewnych ustawień w programie dotyczących właściwości komunikacji pomiędzy Arduino a laptopem. W tym celu klikamy ikonkę *Options* i wyświetla nam się okno ustawień. W zakładce *Serial Port* należy wybrać numer portu (rys. 19a), do którego obecnie podłączone jest Arduino (prawdopodobnie będzie to ten sam port, który był ustawiony w programie Arduino IDE). W polu *Badurate* (rys. 19b) należy ustawić tę samą wartość transmisji danych jaka została wpisana w kodzie programu, czyli *115200*. W zakładce *File Capture* odpowiadającej za zapis danych do pliku, należy w naszym przypadku odznaczyć okienko *Wait for termination string.* Jeśli nie zostanie to odznaczone, to między kolejnymi odczytywanymi wierszami nie będzie „Entera”, a dane będą się zapisywać w „ciągu”. Akceptujemy zmiany przez kliknięcie OK.



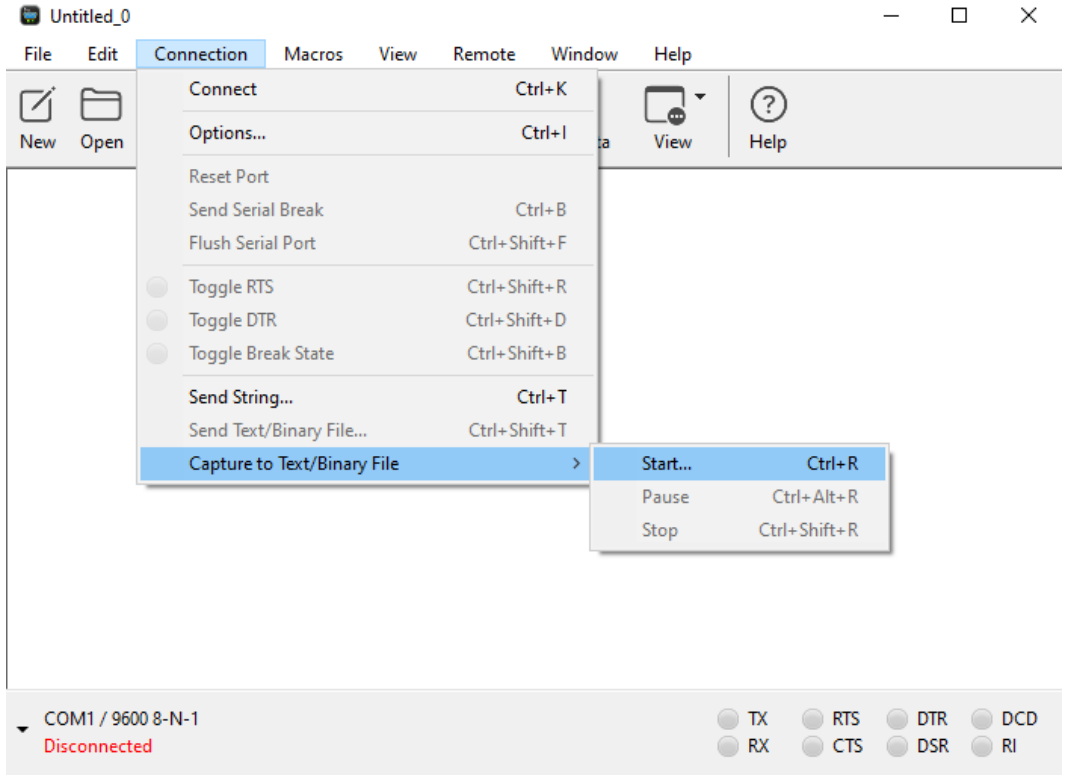
Rys. 19a



Rys. 19b

Program CoolTerm

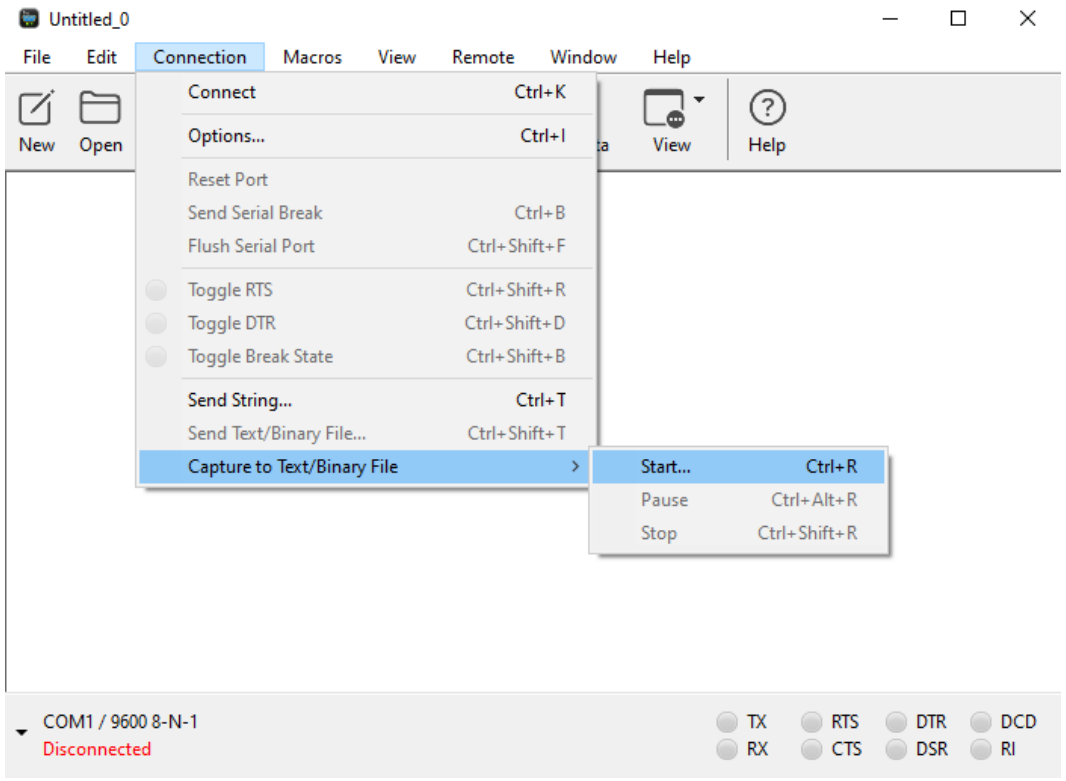
W kolejnym kroku, zanim zaczniemy odczytywać dane wysyłane do komputera z Arduino, włączymy opcje zapisywania ich w pliku tekstowym. Aby to zrobić należy wejść w zakładkę *Connection->Capture to Text/Binary File->Start…*(rys. 20), bądź użyć skrótu *Ctrl+R.* Po wybraniu folderu i nazwy dla pliku klikamy OK. Od tej pory wszystkie dane jaki zostaną wysłane przez Arduino do komputera i odczytane przez program CoolTerm zapiszą się do tego pliku.



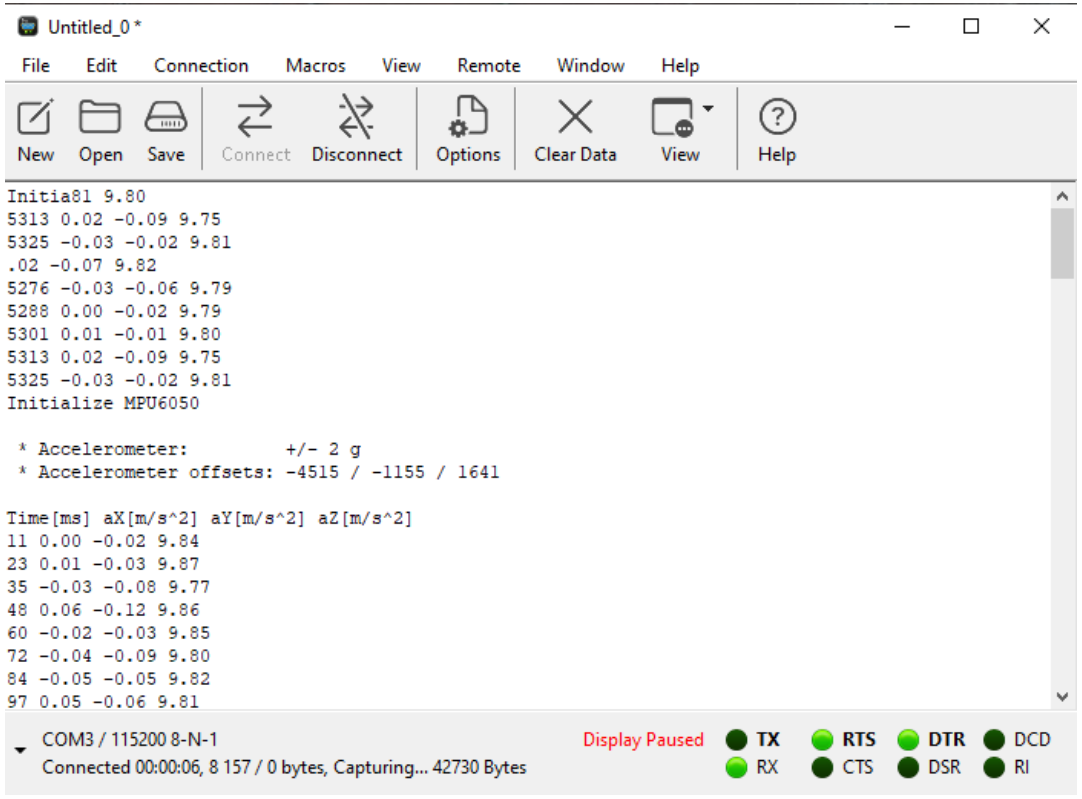
Rys. 20

Program CoolTerm

W kolejnym kroku, zanim zaczniemy odczytywać dane wysyłane do komputera z Arduino, włączymy opcje zapisywania ich w pliku tekstowym. Aby to zrobić należy wejść w zakładkę *Connection->Capture to Text/Binary File->Start…*(rys. 20a), bądź użyć skrótu *Ctrl+R.* Po wybraniu folderu i nazwy dla pliku klikamy OK. Od tej pory wszystkie dane jaki zostaną wysłane przez Arduino do komputera i odczytane przez program CoolTerm zapiszą się do tego pliku. Następnie do zainicjowania odczytu danych należy kliknąć przycisk *Connect*. Dane zaczynają się wyświetlać w terminalu (rys. 20b), natomiast do pliku zapiszą się dopiero po zatrzymaniu zapisu *Connection->Capture to Text/Binary File->Stop…*, lub skrót *Ctrl+Shift+R.*



Rys. 20a



Rys. 20b

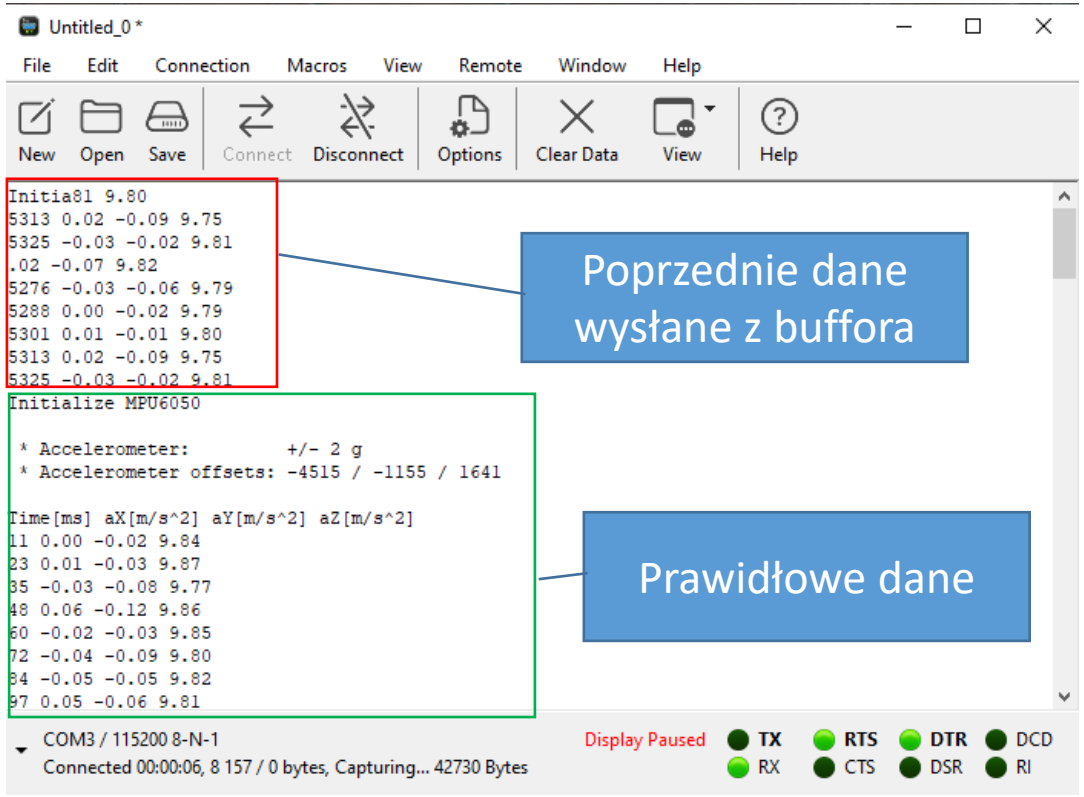
Program CoolTerm

Zapisane dane zgodnie z programem napisanym w Arduino IDE powinny zaczynać się od informacji:

Initialize MPU6050
* Accelerometer: +/- 2 g
* Accelerometer offsets: -4515 / -1155 / 1641 * Accelerometer: +/- 2 g
* Accelerometer offsets: -4515 / -1155 / 1641

Często zdarza się jednak, że podczas odczytu danych w terminalu i pliku tekstowym, gdzie zostały zapisane znajdziemy „pozostałości” po poprzednim odczycie (patrz rys. 20c). Wynika to z faktu, że połączenie USB jest kontrolowane przez chip z własnym buforem tzn. miejscem gdzie może być przechowywana pewna porcja danych.

Podsumowując, zapisane dane należy przetworzyć poprzez usunięcie wierszy zawierających niepotrzebne informacje pochodzące z poprzednich odczytów.



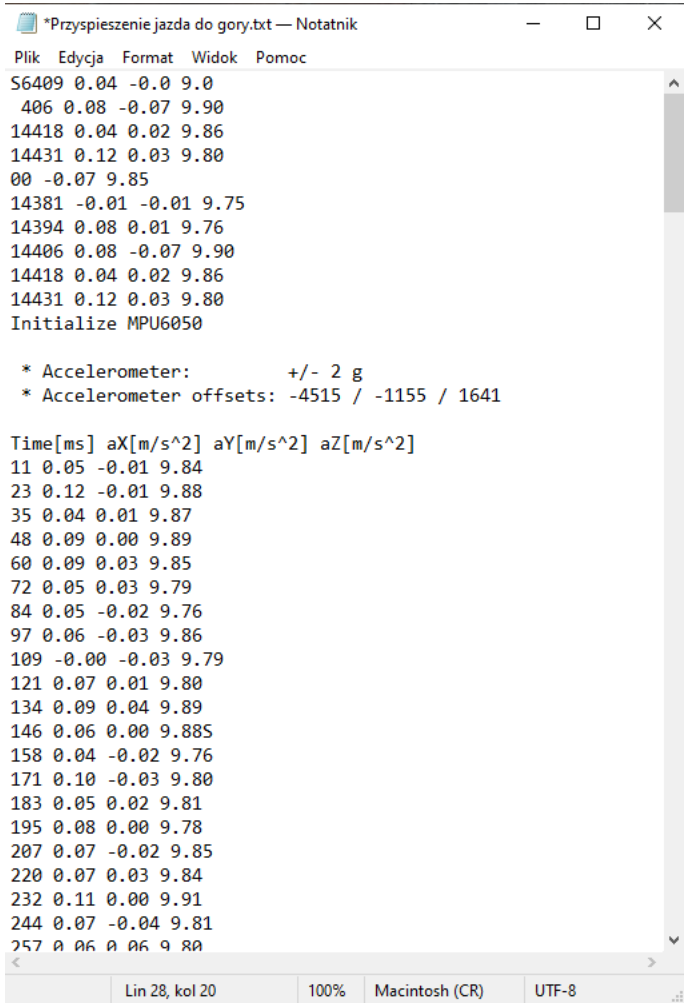
Rys. 20c

Matlab

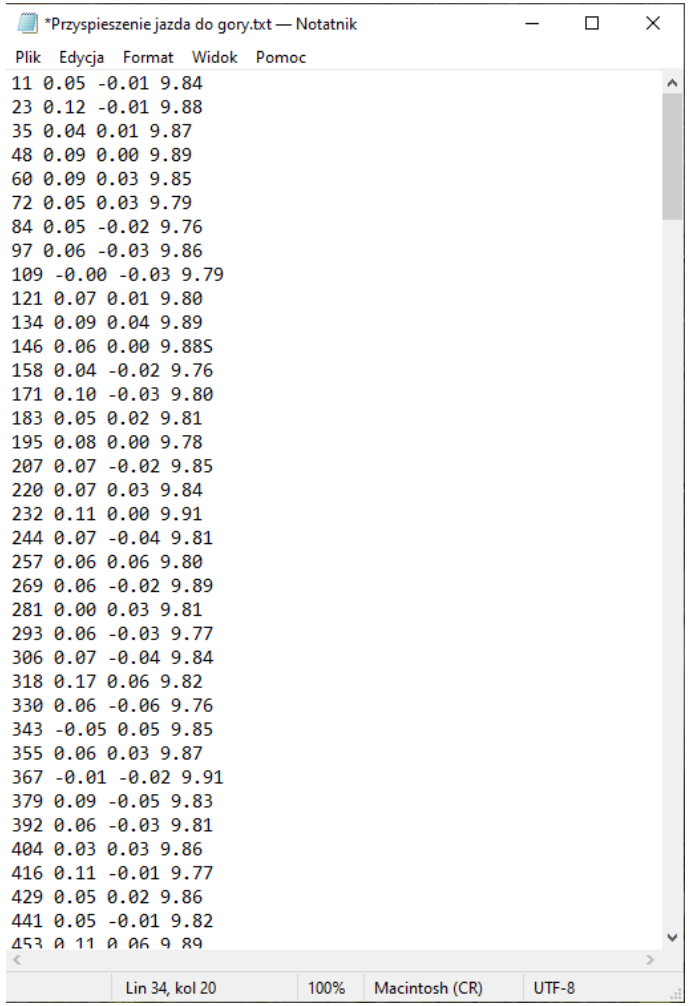
Przy pomocy środowiska obliczeniowego Matlab wykonamy analizę dynamiki windy na podstawie zmierzonych akcelerometrem przyspieszeń.

Analizę ta musimy rozpocząć od usunięcia niepotrzebnych wierszy danych w pliku tekstowym zawierającym pomiary eksperymentalne.

Rys. 21a pokazuje dane zapisane bezpośrednio z programu CoolTerm, a rys. 21b pokazuje jak powinien wyglądać plik tekstowy z danymi potrzebnymi do obliczeń w środowisku Matlab.



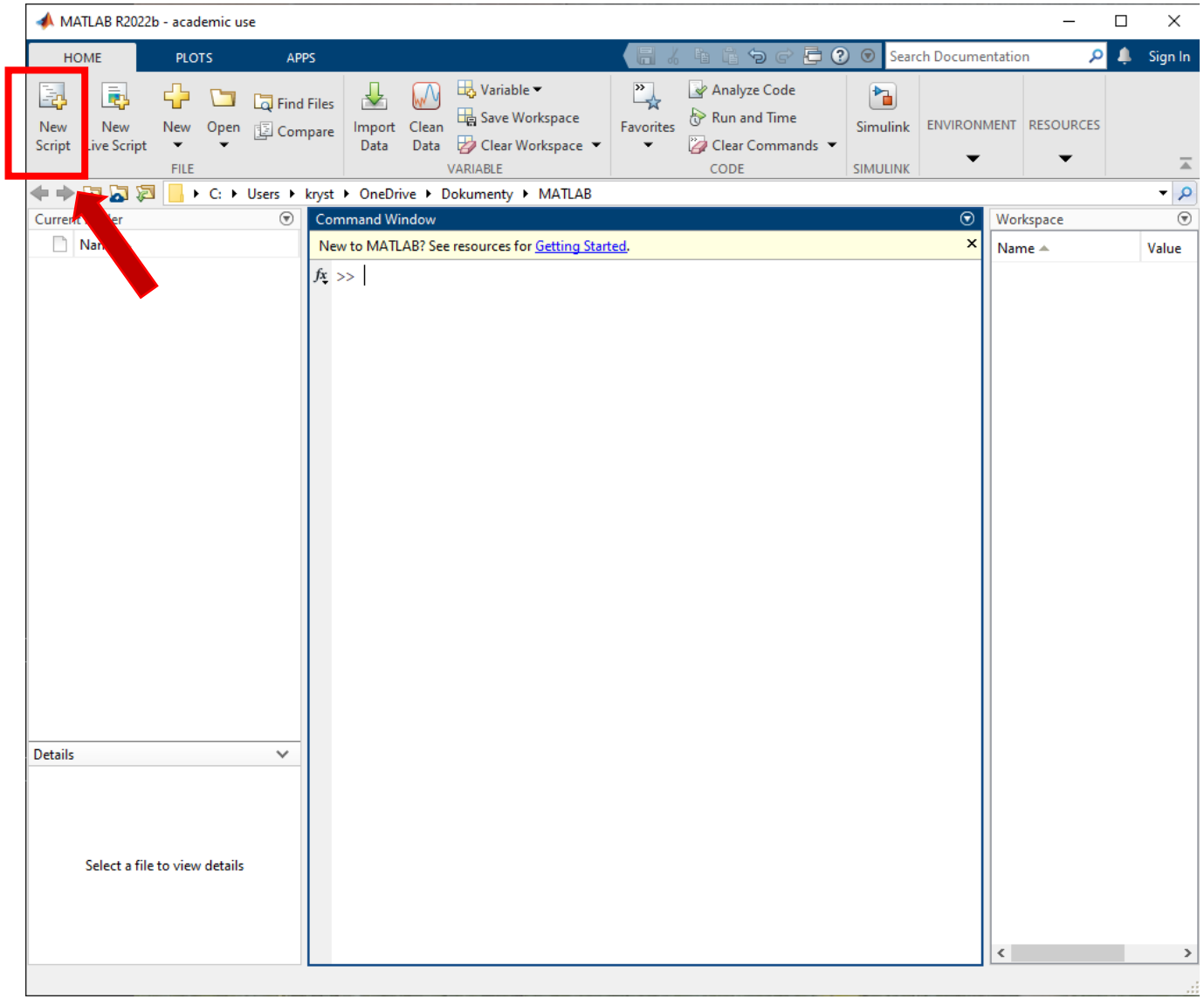
Rys. 21a



Rys. 21b

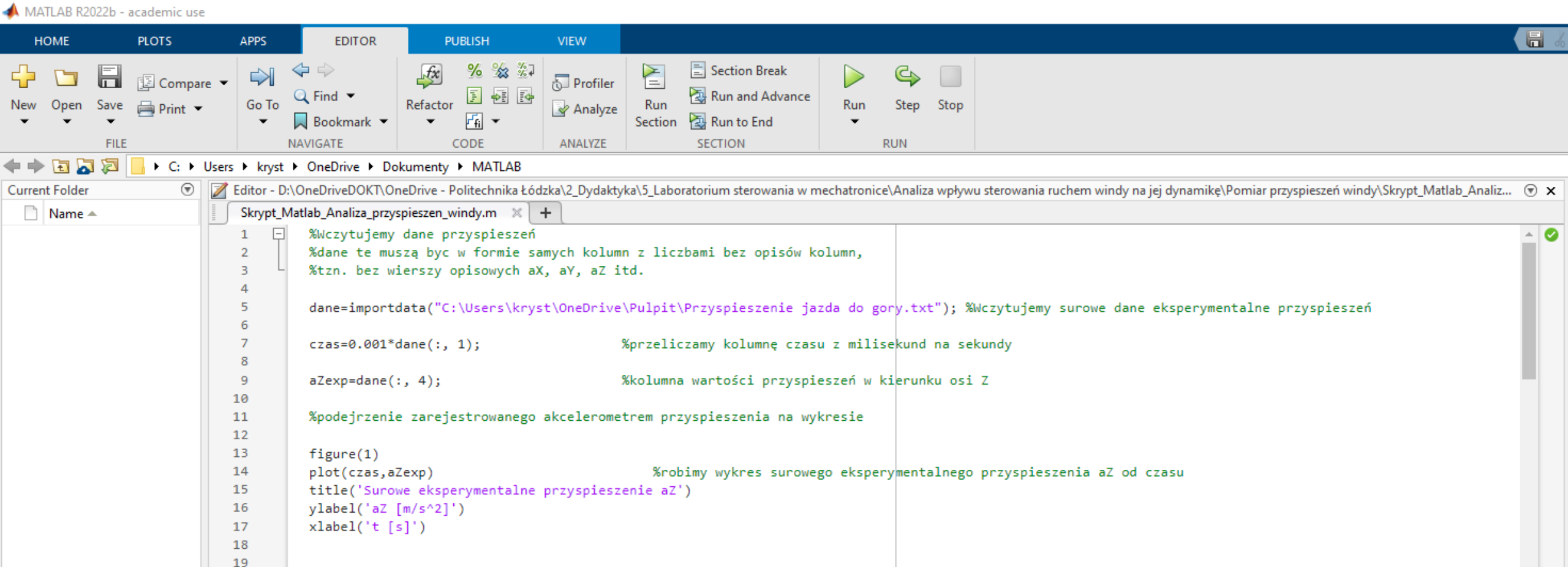
Matlab

Teraz przechodzimy do napisania skryptu, który po wykonaniu w Matlabie dokona obliczeń prędkości i położenia windy oraz zrywu. Klikając w lewym górnym rogu *New Script* otworzy nam się cos na rodzaj notatnika (rys. 22), w którym napiszemy program do analizy dynamicznej. Warto sobie ten plik zapisać.



Rys. 22

Matlab



Po napisaniu powyższego programu i kliknięciu przycisku *Run* powinniśmy otrzymać wykres surowych danych zarejestrowanych akcelerometrem tylko w osi Z, patrz rys. 23a.

Matlab

MATLAB R2022b - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

New Open Save Compare Print — FILE
Go To Find Bookmark — NAVIGATE
Refactor — CODE
Profiler Analyze — ANALYZE
Run Section Section Break Run and Advance Run to End — SECTION
Run Step Stop — RUN

C: ▸ Users ▸ kryst ▸ OneDrive ▸ Dokumenty ▸ MATLAB

Current Folder

Name
fig1.png
figure2.png

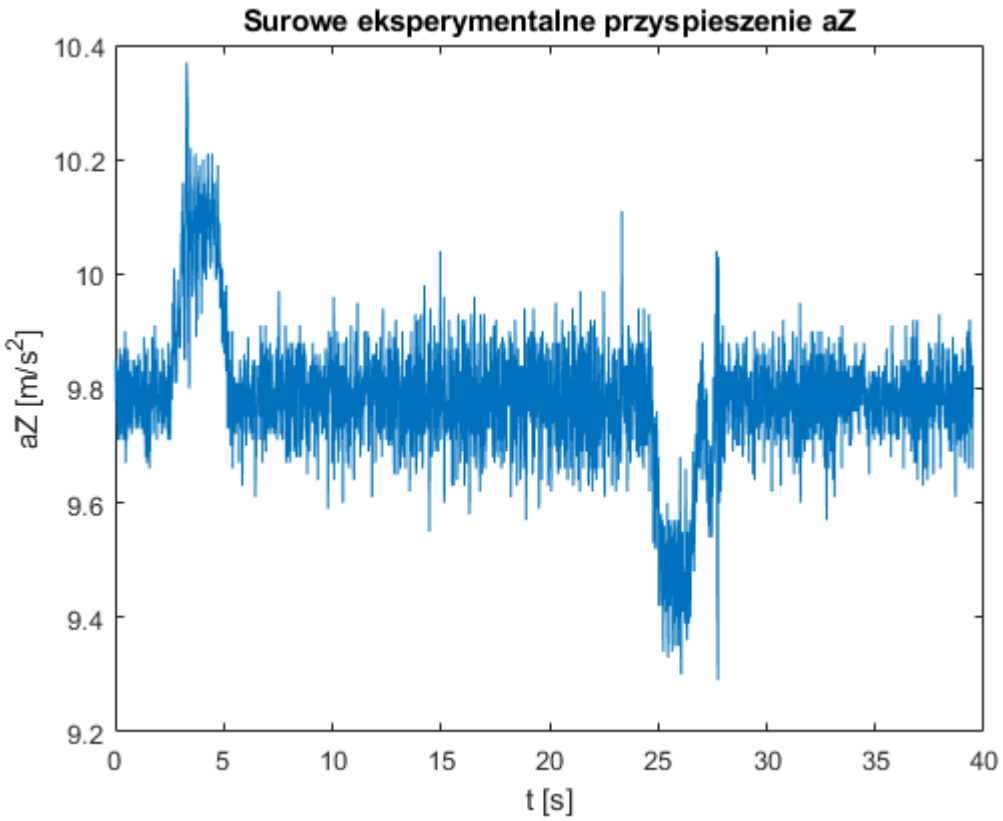
Editor - D:\OneDriveDOKT\OneDrive - Politechnika Łódzka\2_Dydaktyka\5_Laboratorium sterowania w mechatronice\Analiza wpływu sterowania ruchem windy na jej dynamikę\Pomiar przyspieszeń windy\Skrypt_Matlab_Analiz...

Skrypt_Matlab_Analiza_przyspieszen_windy.m

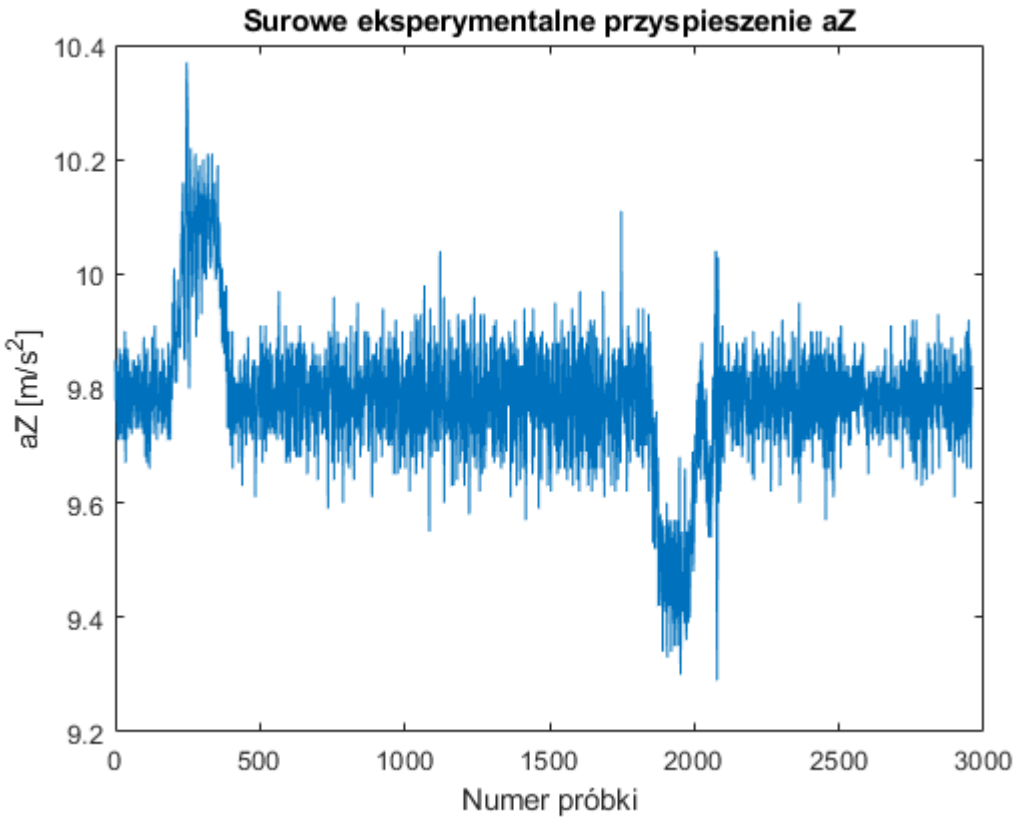
```
7   czas=0.001*dane(:, 1);                  % przeliczamy kolumnę czasu z milisekund na sekundy
8
9   aZexp=dane(:, 4);                       % kolumna wartości przyspieszeń w kierunku osi Z
10
11  %podejrzenie zarejestrowanego akcelerometrem przyspieszenia na wykresie
12
13  figure(1)
14  plot(czas,aZexp)                            %robimy wykres surowego eksperymentalnego przyspieszenia aZ od czasu
15  title('Surowe eksperymentalne przyspieszenie aZ')
16  ylabel('aZ [m/s^2]')
17  xlabel('t [s]')
18
19
20  % Jak widzimy wykres ten jest przesunięty w osi rzędnych o stałą wartość przyspieszenia ziemskiego g równą około 9.81 m/s^2.
21  % Dlatego w celu znalezienia wartości przyspieszeń generowanych tylko przez poruszającą się windę należy pomniejszyć zarejestrowane przyspieszenia
22  % o wartość przyspieszenia ziemnskiego g. Żeby to zrobić należy oszacować na podstawie zarejestrowanych danych ile wynosi wartość przyspieszenia g.
23  % Oszacowanie to wykonamy licząć średnią arytmetyczną wartości przyspieszenia na przedziale czasowym, gdy winda nie miała innego
24  % przyspieszenia niż ziemskie, tj. np. na przedziale czasowym 10-20 sekund. Jednakże w Matlabie łątwiej nam będzie operować nie na przedziale czasowym, a
25  % a przedziale wyznaczonym przez numery zarejestrowanych próbek, pomocny w tym będzie poniższy wykres.
26
27  figure(2)
28  plot(aZexp)                             %robimy wykres surowego przyspieszenia eksperymentalnego aZ od numeru próbki
29  title('Surowe eksperymentalne przyspieszenie aZ')
30  ylabel('aZ [m/s^2]')
31  xlabel('Numer próbki')
32
```

Po napisaniu powyższego programu i kliknięciu przycisku *Run* powinniśmy otrzymać wykres surowych danych aZ w dziedzinie numeru próbki, patrz rys. 23b.

Matlab

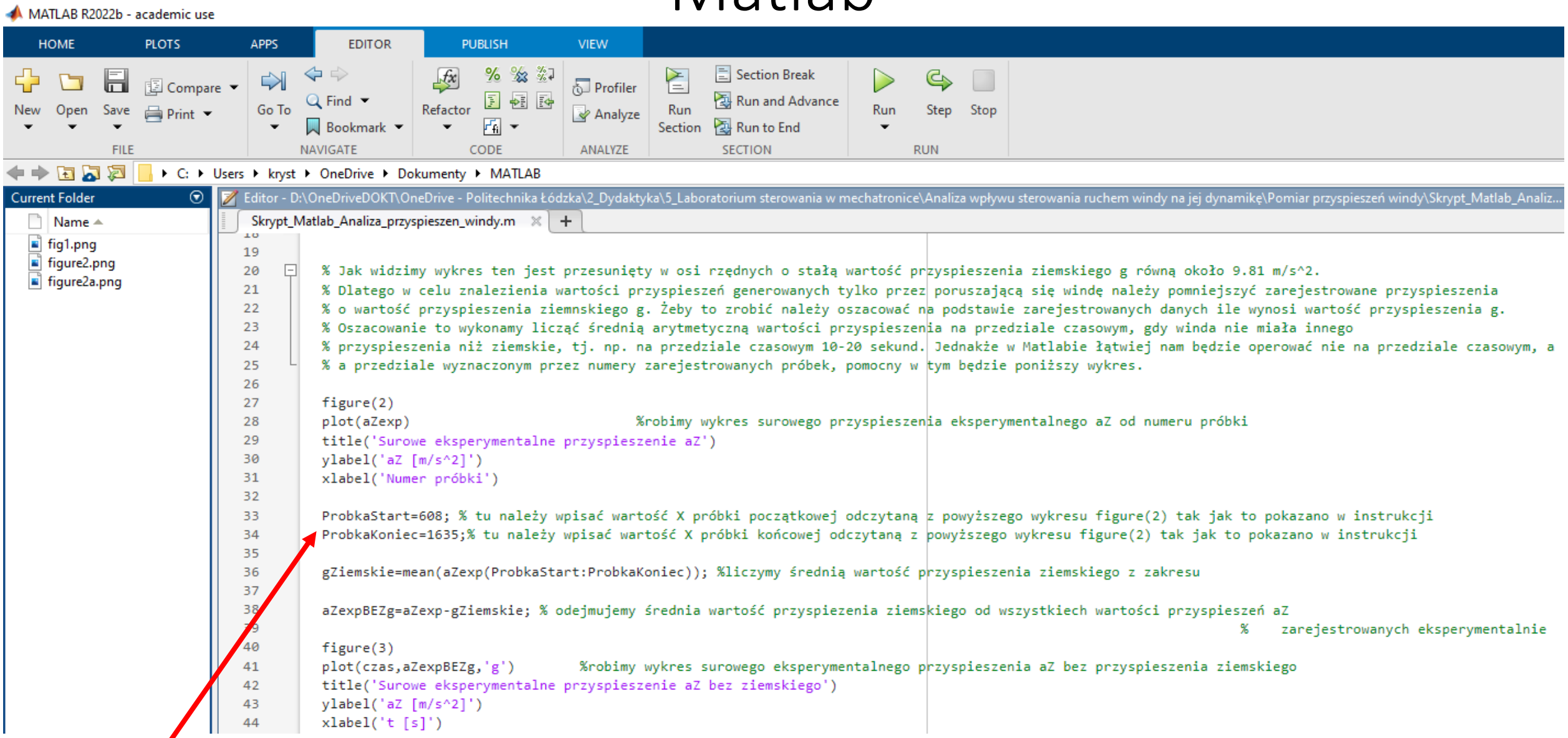


Rys. 23a



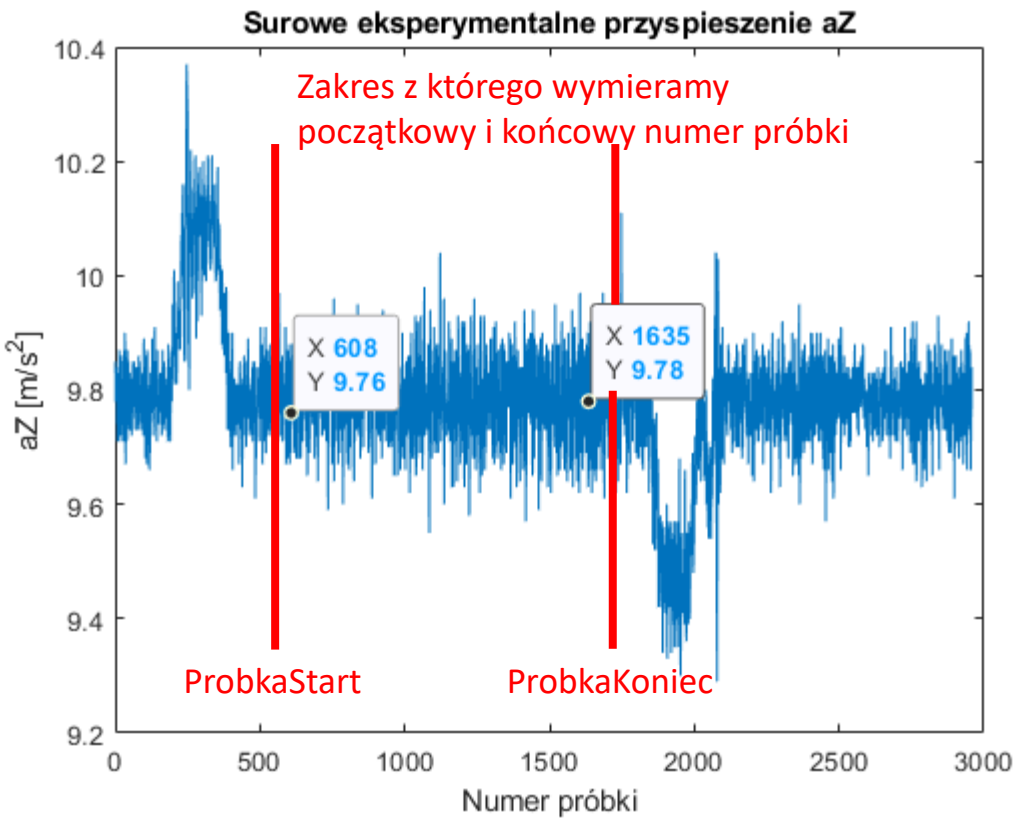
Rys. 23b

Matlab

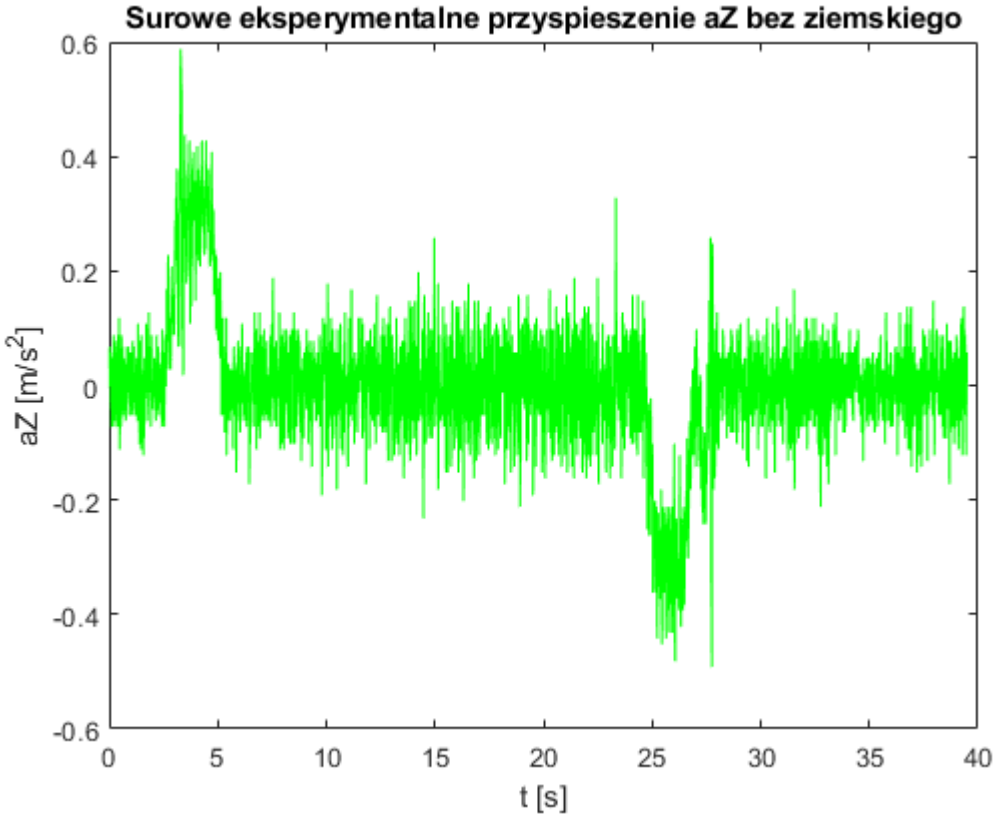


Na podstawie wygenerowanego wykresu, patrz rys. 23b, znajdujemy wartość początkową ProbkaStart i ProbkaKoniec (rys. 23c) potrzebne do określenia przedziału na jakim będzie liczona średnia arytmetyczna przyspieszenia ziemskiego g. Wartości wpisujemy do programu, a w efekcie otrzymamy wykres surowego przyspieszenia eksperymentalnego pomniejszonego o przyspieszenie ziemskie (rys. 23d)

Matlab

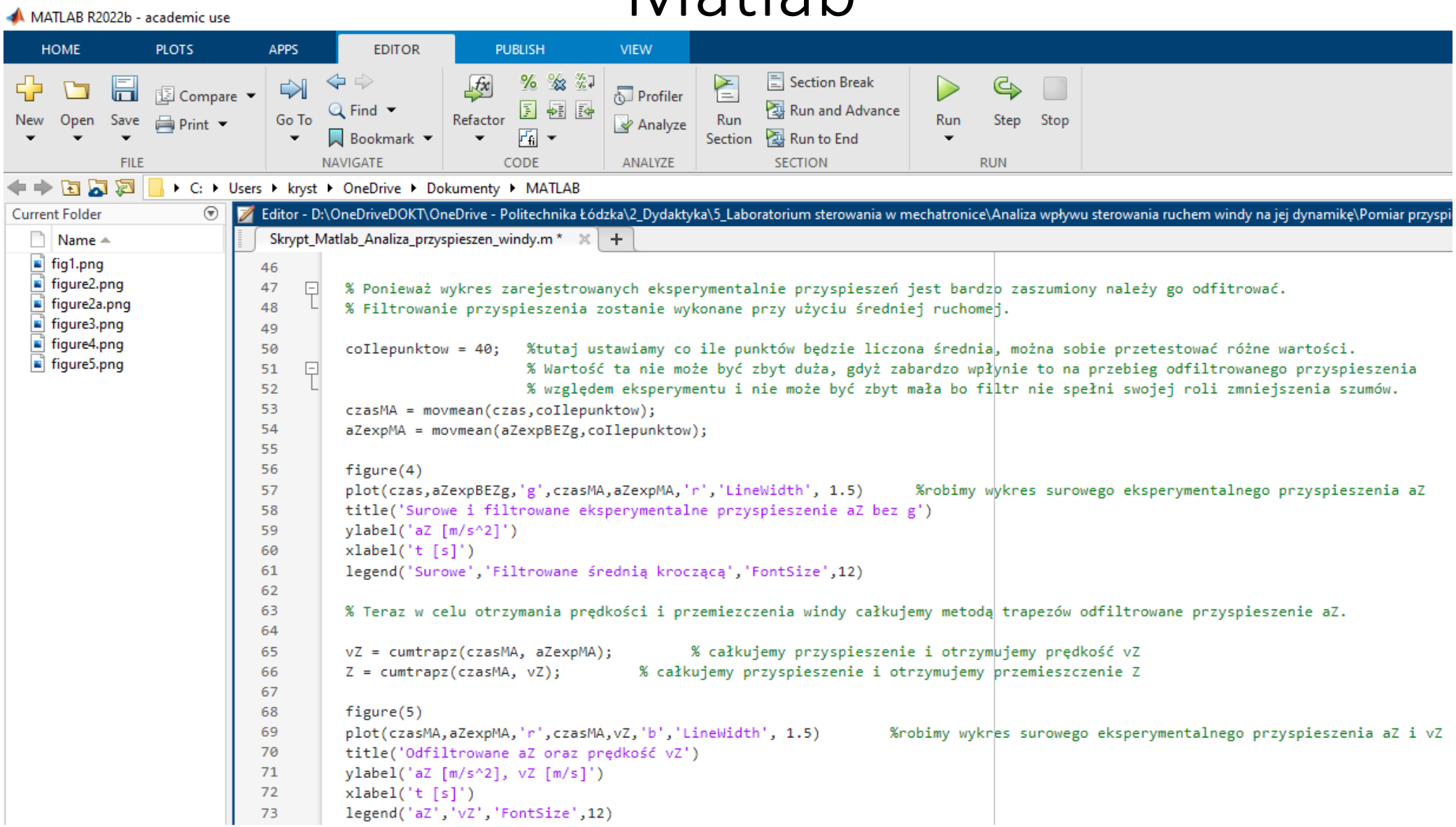


Rys. 23c



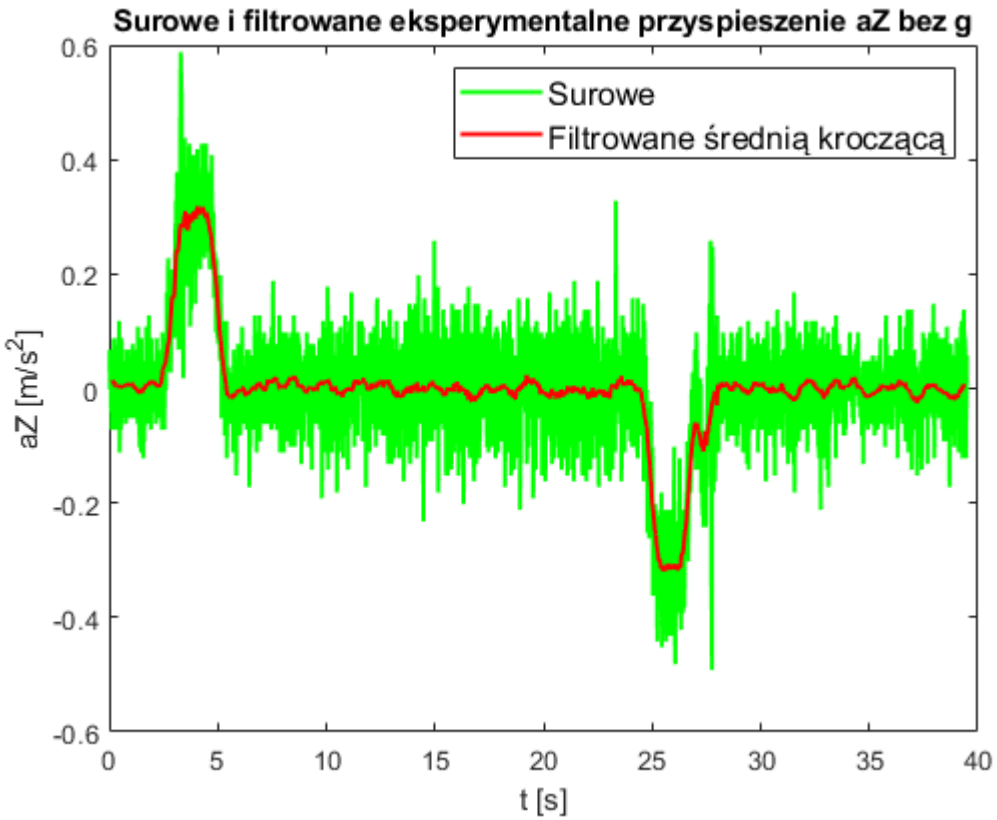
Rys. 23d

Matlab

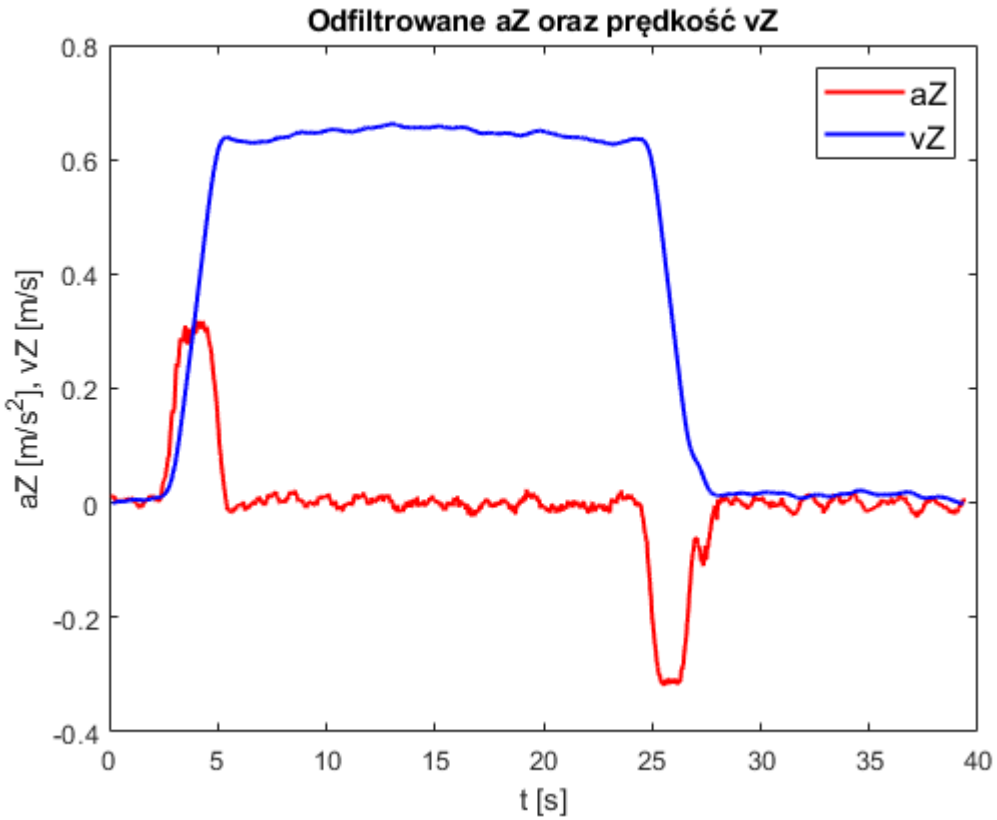


Po dopisaniu kolejnych linijek programu otrzymamy wykres odfiltrowany przebieg przyspieszenia aZ, patrz rys. 23e. Na jego podstawie zostaną obliczone prędkość i przemieszczenie windy, rys. 23f.

Matlab

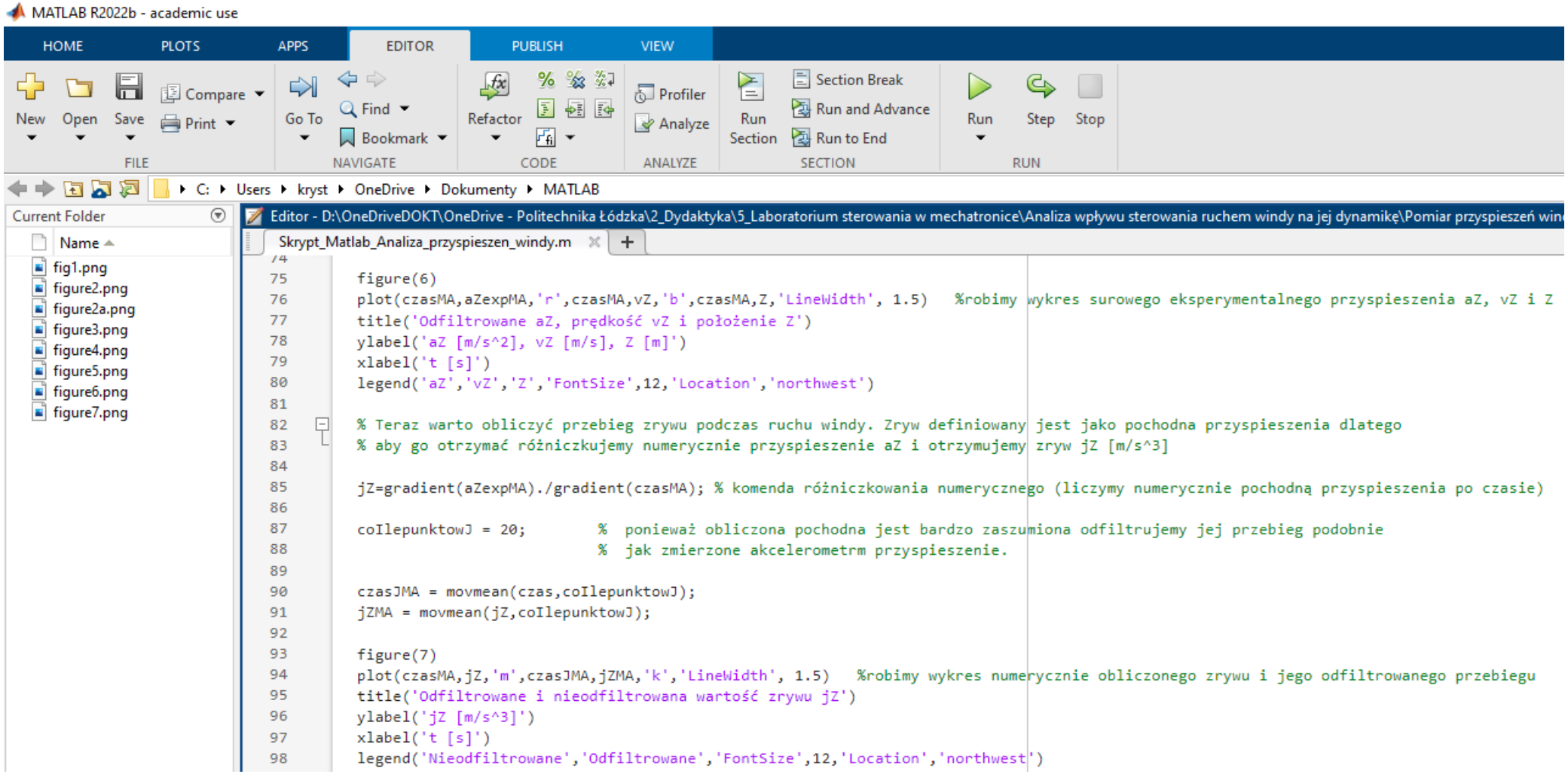


Rys. 23e



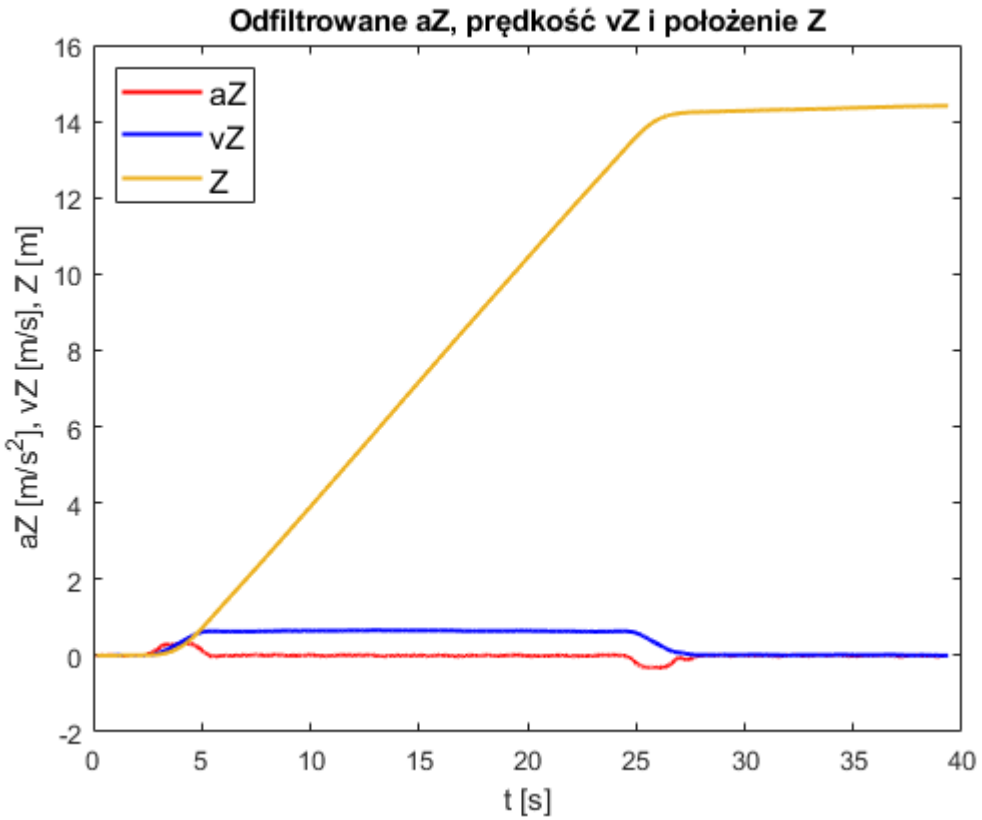
Rys. 23f

Matlab

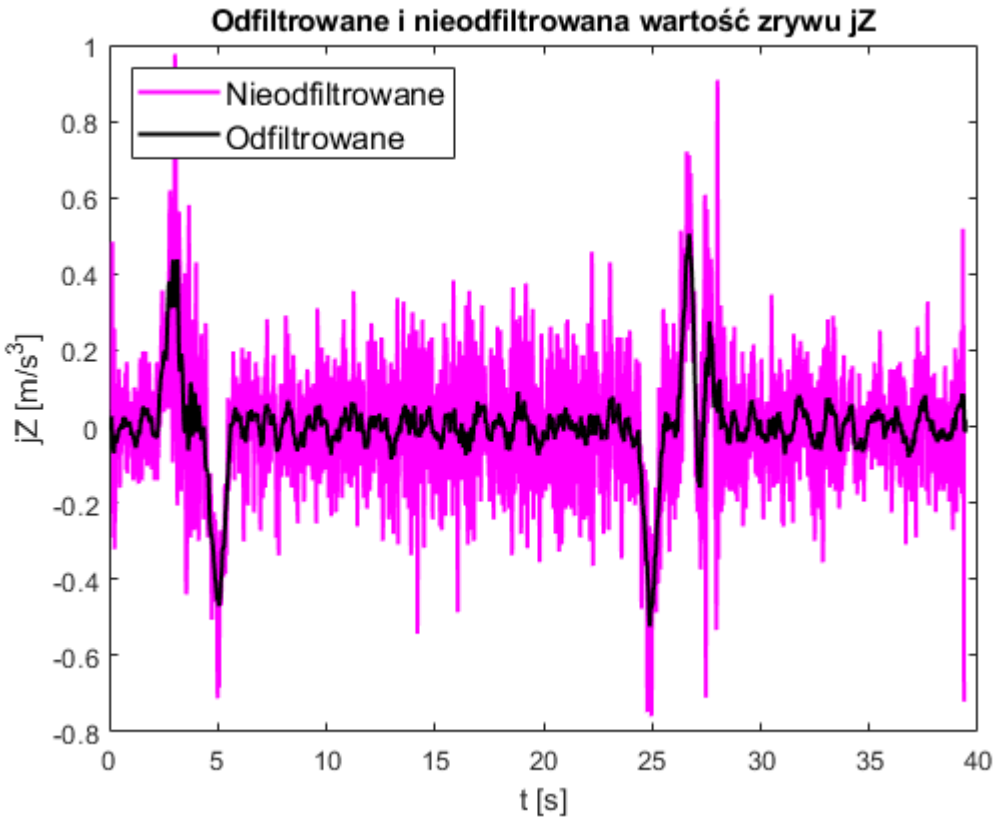


Po dopisaniu kolejnych linijek programu otrzymamy wykres z rys. 23f, ale z dodanym przebiegiem przemieszczenia windy (rys. 23g). Ponadto obliczymy numerycznie zryw windy i dokonamy filtracji jego przebiegu, rys. 23h.

Matlab

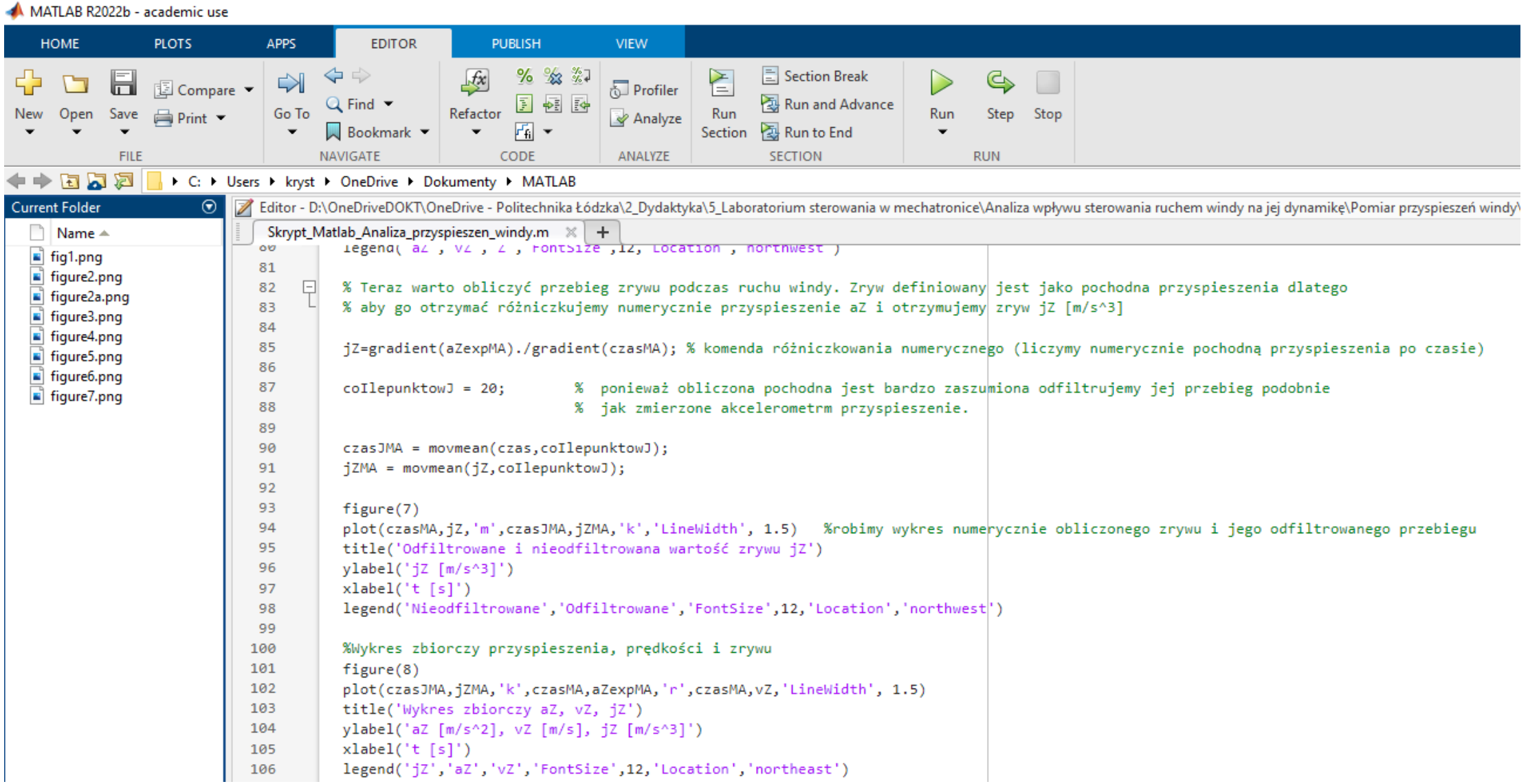


Rys. 23g

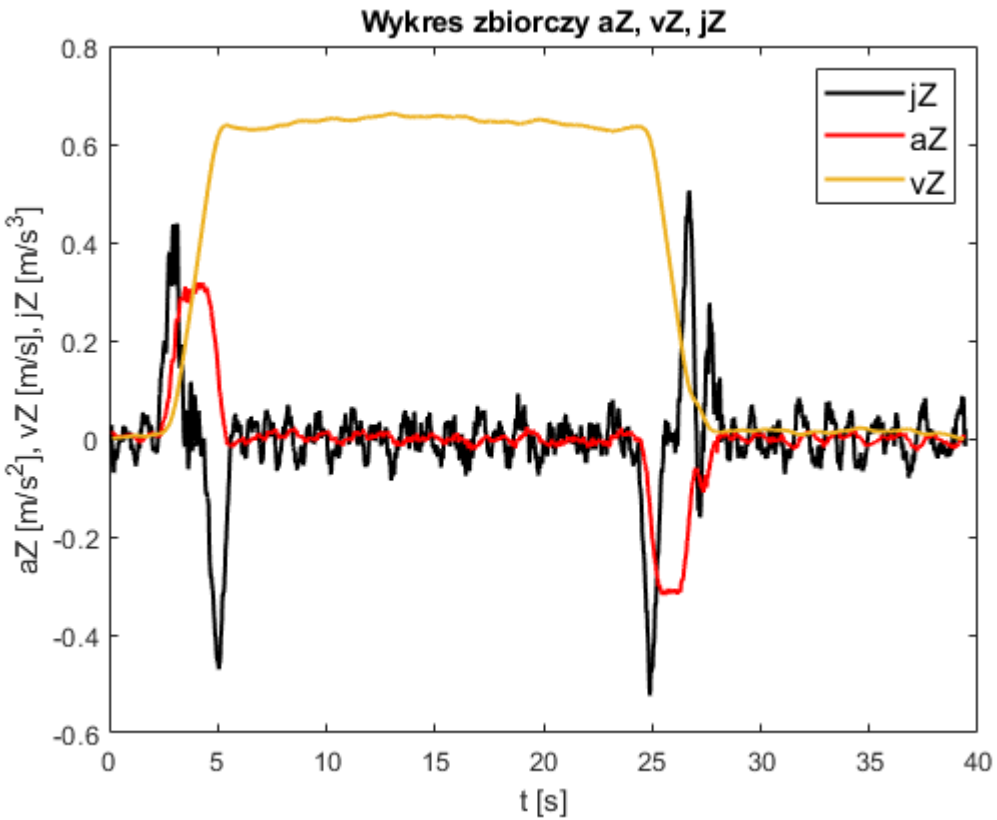


Rys. 23h

Matlab



Po dopisaniu ostatnich linijek programu otrzymamy wykres z rys. 23i będący zbiorczym wykresem wszystkich przebiegów zmiennych charakteryzujących ruch windy


Wykres zbiorczy aZ, vZ, jZ
jZ
aZ
vZ
aZ [m/s^2], vZ [m/s], jZ [m/s^3]
t [s]
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
5
10
15
20
25
30
35
40

Rys. 23i

Zadania do wykonania przez studentów

1. Podłączenie układu pomiarowego do laptopa.
2. Zainstalowanie w środowisku Arduino IDE biblioteki do modułu MPU6050 (gotowy plik .ZIP otrzymacie od prowadzącego).
3. Wgranie gotowego programu (otrzymanego od prowadzącego) do Arduino przy pomocy środowiska Arduino IDE. Sprawdzenie poprawności działania układu pomiarowego, wykonanie testowego pomiaru przyspieszeń.
4. Udanie się do wskazanych przez prowadzącego wind w budynku wydziału razem z układem pomiarowym i wykonanie pomiarów przyspieszeń windy podczas podróży między piętrami. Pomiary należy wykonać przy pomocy programu CoolTerm i zapisać zarejestrowane wyniki. Pomiary wykonać zarówno w jeździe do góry jak i do dołu. Podczas pomiarów NIE PORUSZAĆ SIĘ W WINDZIE, ani nie wykonywać gwałtownych ruchów. Najlepiej po uruchomieniu pomiaru nie poruszać się. Należy mieć na uwadze, że akcelerometr może zarejestrować przyspieszenia pochodzące od drgań otwierających się drzwi. W czasie wykonywania ćwiczenia nie utrudniać przemieszczania się innym użytkownikom windy. Najlepiej przepuścić ich, a pomiar dokonać w obecności tylko członków grupy laboratoryjnej. W czasie pomiaru nie poruszać akcelerometrem oraz uważać na przewód łączący go z laptopem. Jeśli grupa dysponuje własnym laptopem (z baterią mogącą wytrzymać na czas pomiarów) zachęca się do skorzystania z niego, ułatwi to późniejsze wykonywanie sprawozdania w miejscu innym niż PŁ.
5. Na podstawie wykonanych pomiarów przyspieszeń dokonać analizy dynamiki windy w programie Matlab. Program można pobrać na stronie (https://www.mathworks.com/products/matlab.html). Każdy ze studentów posiada licencję na to oprogramowanie po zalogowaniu się przy użyciu konta uczelnianego.
6. Wykonać sprawozdanie.