Pomiar przyspieszenia (013; 09.08.2009; *arduino; processing; blender*)

W ostatnim czasie akcelerometry (czujniki przyspieszenia) znalazły nowe zastosowanie, a mianowicie montowane są w telefonach komórkowych, co pozwala określać ich orientację i ruch. W przeciwieństwie do drogich czujników przemysłowych, te stosowane w telefonach są bardzo małe, kompaktowe, tanie i o niskim pasmie przenoszonych częstotliwości. Opiszę zatem, jak jeden z takich akcelerometrów wykorzystać wraz z Arduino, kreślić wykresy w Processingu, oraz jak przykładowo zastosować to w prostej grze w środowisku Blender'a.

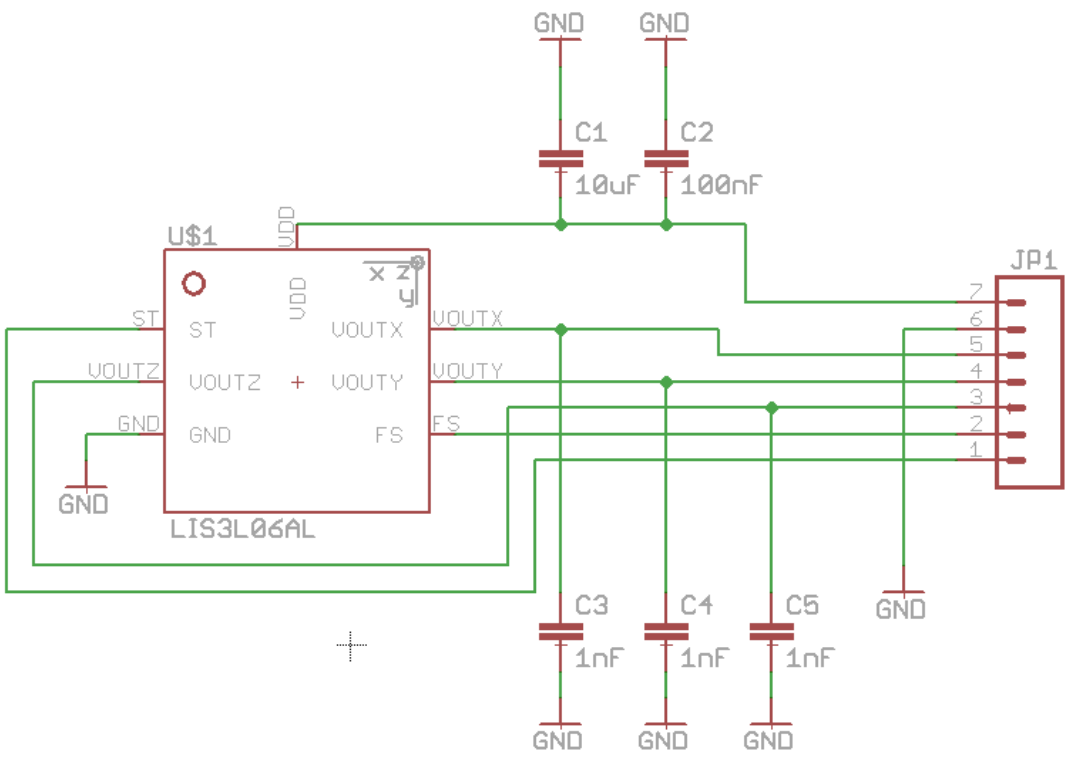
Akcelerometr.

W opisywanym projekcie zastosowałem akcelerometr LIS3L06AL o następujących parametrach:

- zakres pomiarowy: ±2g/6g,
- napięcie zasilania: 2,4V – 3,6V,
- częstotliwość rezonansowa 1,5kHz,
- parametry wyjściowe odniesione do napięcia zasilania,
- wymiary: 5mm x 5mm x 1,6mm.

Przy wyborze kierowałem się głównie niską ceną (29zł) i łatwą dostępnością układu. Nastręczyło mi to jednak nie lada kłopotów podczas montażu układu na obwodzie płytki i przy okazji uszkodzenia jednego pola akcelerometru (odpowiedzialnego za pomiar przyspieszenia wzdłuż osi x). Polecam zatem używać większych i łatwiejszych do lutowania układów.

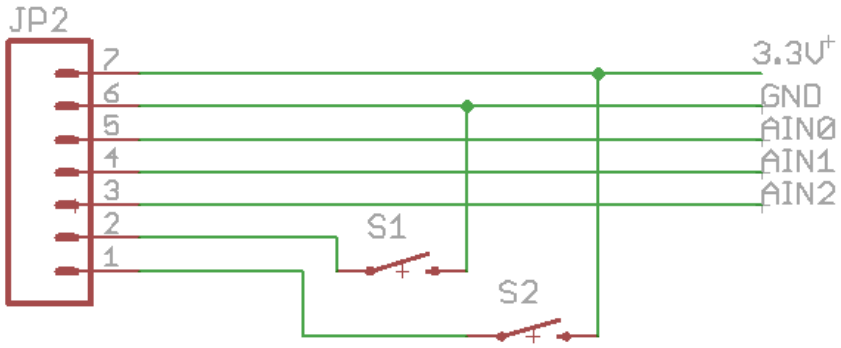
Do montażu akcelerometru i niezbędnych kondensatorów zaprojektowałem obwód drukowany. Schemat połączeń jest następujący:



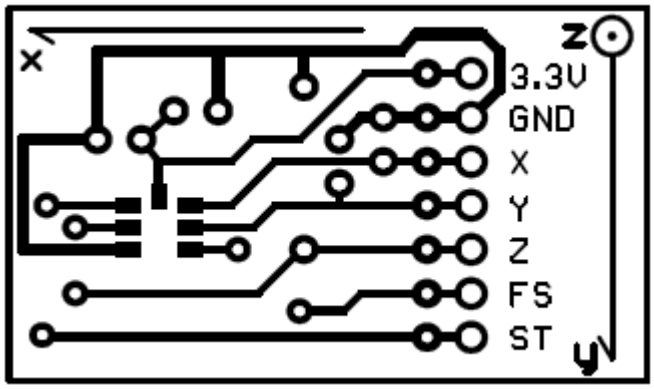
Postanowiłem zasilić akcelerometr prosto ze złącza 3.3V Arduino, gdyż jest to napięcie stabilizowane prosto z mikrokontrolera (jednak w tym przypadku zwarcie w układzie akcelerometru może zaburzyć pracę mikrokontrolera). Kondensatory C1 i C2 mają za zadanie usunąć zakłócenia i wahania zasilania. Kondensatory C3, C4 i C5 tworzą wraz z wewnętrznymi rezystorami akcelerometru (110kOhm) filtry dolnoprzepustowe o częstotliwości około 1450Hz.

Złącze logiczne FS (full scale) w stanie niepodłączonym skutkuje ustaleniem zakresu akcelerometru na 6g, w przypadku podłączenia do masy zakres zmienia się na 2g.
Złącze ST (self test) podczas podłączenia do zasilania (takiego samego jak całegoakcelerometru) powoduje przyłożenie siły do elementów pomiarowych wewnątrz akcelerometru co powinno skutkować wzrostem napięcia wyjściowego pinów Vouty i Voutz o około 50mV i spadkiem napięcia wyjściowego pinu Voutx o około 50mV.

Połączenie płytki z zamontowanym akcelerometrem do Arduino zrealizowałem za pomocą następującego schematu (złącze JP2 łączy się z JP1):



Schemat obwodu drukowanego jest następujący (http://myinventions.pl/013/013board.pdf):



Kalibracja.

Aby rozpocząć korzystanie z czujnika należy rozpoznać wartości napięcia wyjściowego dla charakterystycznych łatwo osiągalnych wartości przyspieszenia (0, 1g, -1g). Zastosuję następujący kod do Arduino:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print(analogRead(0));
  Serial.print(" ");
  Serial.print(analogRead(1));
  Serial.print(" ");
  Serial.println(analogRead(2));
  delay(100);
}
```

Obserwując wartości wyświetlone w Serial Monitor w różnych położeniach czujnika można przedstawić następującą tabelę szacunkowej kalibracji (wartości odczytane w rozdzielczości 10bitów i w przeliczeniu na wolty):

		2 g (FS=0)	6 g (FS=1)	2g (ST=1 FS=0)
Y	0g	325 (1,587V)	333 (1,626V)	334 (1,631V)
	+1g	186 (0,908V)	284 (1,387V)	195 (0,952V)
	-1g	460 (2,246V)	377 (1,841V)	471 (2,300V)
Z	0g	335 (1,636V)	330 (1,611V)	345 (1,685V)
	+1g	211 (1,030V)	293 (1,431V)	222 (1,084V)
	-1g	469 (2,290V)	382 (1,865V)	481 (2,349V)

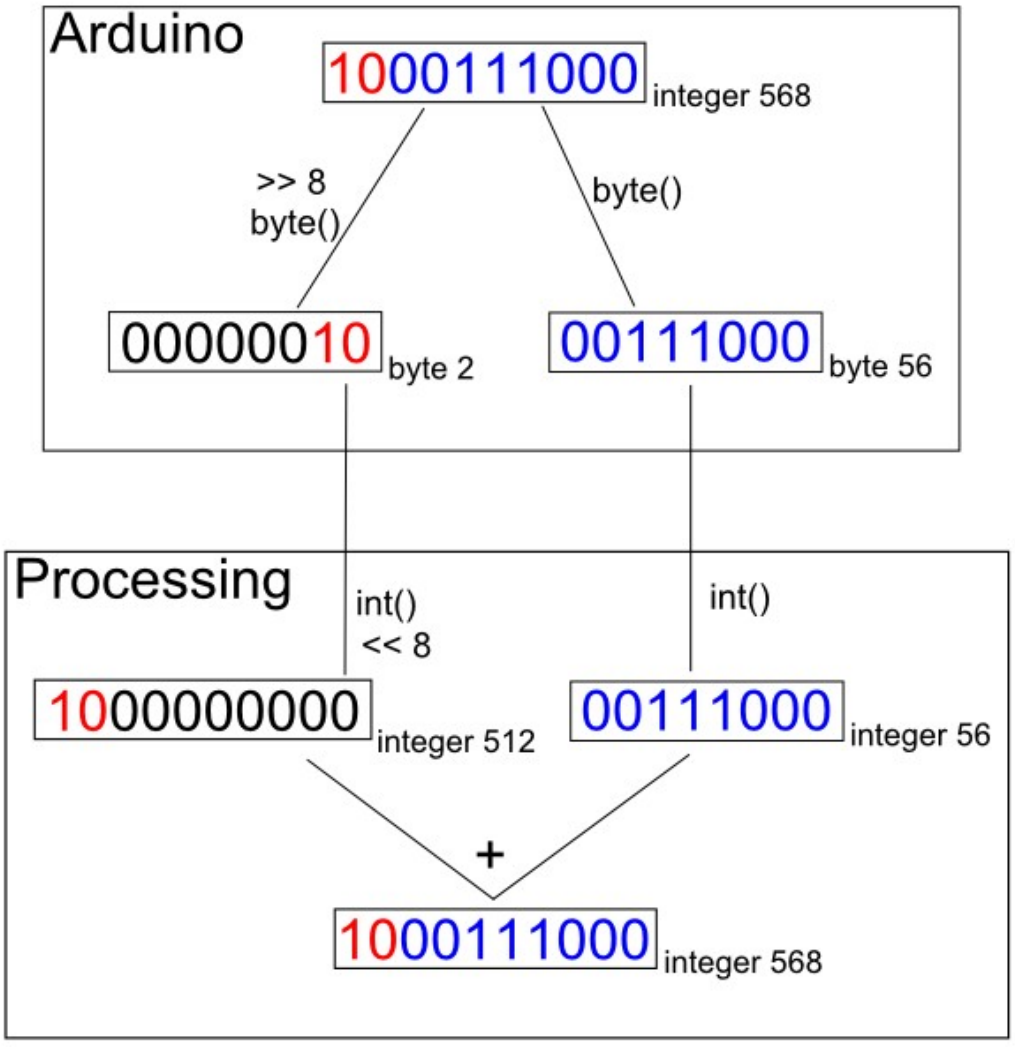
Tak jak podano w specyfikacji, napięcie dla zerowego przyspieszenia jest w bliskie połowie napięcia zasilającego, a użycie funkcji ST zwiększa napięcia o około 50mV.

Możemy też określić szacunkowo czułość i porównać ją z wartością katalogową:

	2g	6g
Y	0,669 V/g	0,227 V/g
Z	0,630 V/g	0,217 V/g
specyfikacja	Vdd/5±10% = 0,66V±10%	Vdd/15±10% = 0,22V±10%

Prezentacja danych na wykresach.

Rozważmy przesyłanie danych z dwóch równoczesnych pomiarów poprzez Arduino do środowiska Processing bez utraty danych (bez zmiany wartości na zakres 8bitowy). W naszym przypadku mamy do przesłania liczby 10bitowe, zatem ich przesył realizować będziemy w postaci dwóch bajtów na każdą liczbę. Schemat przesłania przykładowej liczby przedstawia się następująco:



Tenże sposób konwersji praktycznie prześledzić można w następującym kodzie Processingu:

```
int L=568;
byte A;
byte B;
int C=0;

print("L="); print(L); print(" bin:"); println(binary(L));

A=byte(L); print("A="); print(int(A)); print(" bin:"); println(binary(A));
B=byte (L>>8); print("B="); print(int(B)); print(" bin:"); println(binary(B));


print("A="); print(int(A)); print(" bin:"); println(binary(A));
C = int(B) << 8; print("C="); print(C); print(" bin:"); println(binary(C));
L=int(A)+int(C); print("L="); print(L); print(" bin:"); println(binary(L));
```

Praktyczna realizacja przesyłania danych 10bitowych między Arduino a Processingiem może odbywać się za pomocą następujących kodów:

```
int L;
byte A;
byte B;
void setup() {
  Serial.begin(9600);
}
void loop() {
  L=analogRead(0);
  A=byte(L);
  B=byte(L >> 8);
  Serial.print(A);
  Serial.print(B);
  delay(500);
}
```

```
int A;
int B;
int L;
boolean ab=true;
import processing.serial.*;
Serial myPort;

void setup() {
  myPort = new Serial(this,"COM4", 9600);
}
void draw() {
  if ( myPort.available() > 0) {
    if (ab == true) {
```

```
      A=int(myPort.read());
      ab=false;
    }
    else if (ab == false) {
      B = int(myPort.read()) << 8;
      L=A+B;
      println(L);
      ab=true;
    }
  }
}
```

Konieczne było tu przełączanie pomiędzy kolejnymi pętlami tak, aby wywoływać funkcję myPort.available(), która uchroni nas od gubienia danych i łączenia w pary liczb A i B nie pochodzących z tej samej liczby L.

Ostatecznie do pomiaru przyspieszenia z dwóch kierunków akcelerometru i wysłania wyżej opisaną metodą danych do środowiska Processing użyję kodu Arduino:

```
int Y;
int Z;
byte A;
byte B;

void setup() {
  Serial.begin(9600);
}
void loop() {
  Y=analogRead(0);
  Z=analogRead(1);
  A=byte(Y);
  B=byte(Y >> 8);
  Serial.print(A);
  Serial.print(B);
  A=byte(Z);
  B=byte(Z >> 8);
  Serial.print(A);
  Serial.print(B);
  delay(100);
}
```

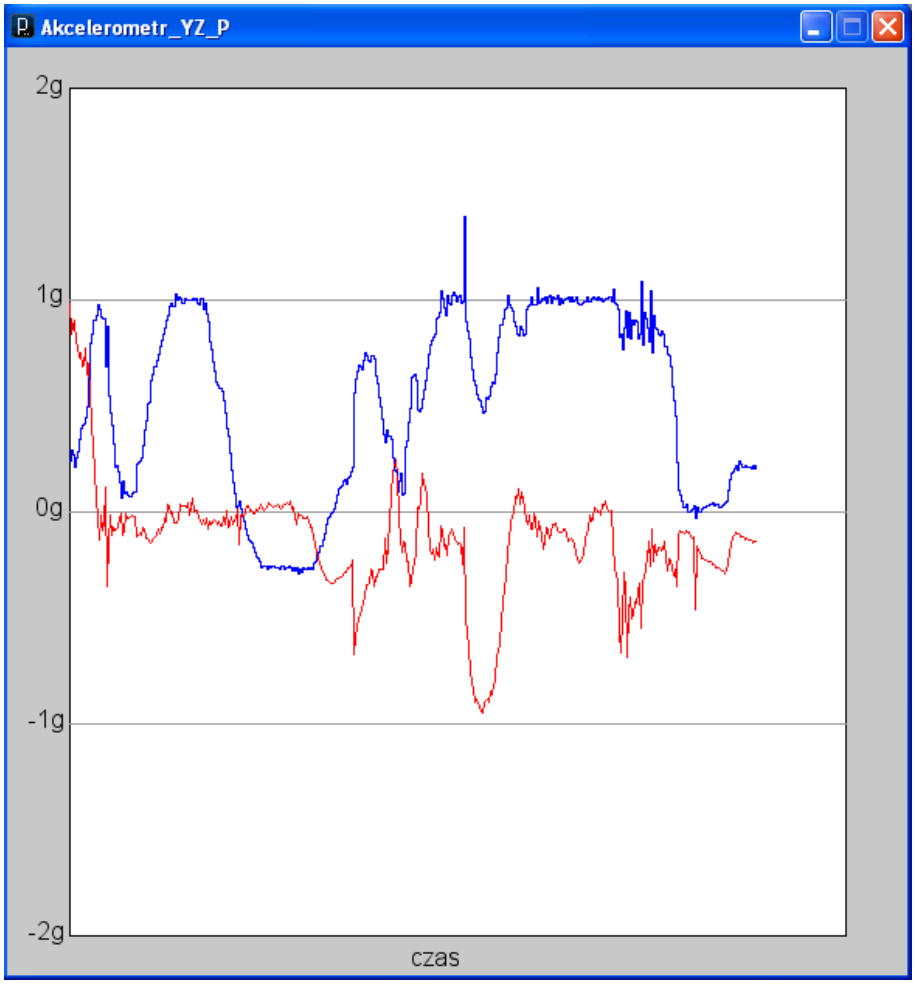
Odpowiedni kod w Processingu kreślący wykresy:

```
import processing.serial.*;
Serial myPort;
int A, B, Y, Z;
int aybyazbz=1;
int t=40;
int pt, pY, pZ;
int offset=330;
void setup() {
  size(580, 600); background(200);
  rect(40,26,500,548);
  stroke(150);
  line(40,163,540,163); line(40,300,540,300); line(40,437,540,437);
  PFont font; font = loadFont("ArialMT-16.vlw"); textFont(font,16);
  fill(0);
  text("2g",18,30); text("1g",18,165);
  text("0g",18,303); text("-1g",14,440);
  text("-2g",14,577); text("czas",260,594);
  myPort = new Serial(this,"COM4", 9600);
}
void draw() {
  if ( myPort.available() > 0) {
    switch(aybyazbz) {
    case 1:
      A=int(myPort.read());
      aybyazbz=2;
      break;
    case 2:
      B = int(myPort.read()) << 8;
      Y=300-(A+B)+offset;
      stroke(255,0,0);
      if (t==40) {
        point(t,Y);
        pt=t;
        pY=Y;
      }
      else {
        line(pt, pY, t, Y);
        pt=t;
        pY=Y;
      }
      aybyazbz=3;
      break;
    case 3:
      A=int(myPort.read());
      aybyazbz=4;
      break;
    case 4:
      B = int(myPort.read()) << 8;
      Z=300-(A+B)+offset;
      stroke(0,0,255);
      if (t==40) {
        point(t,Z);
        pt=t;
```

```
        pZ=Z;
      }
      else {
        line(pt, pZ, t, Z);
        pt=t;
        pZ=Z;
      }
      aybyazbz=1;
      t++;
      break;
    }
    if (t>=540){
      t=40;
      stroke(0); fill(255);
      rect(40,20,500,560); line(40,163,540,163);
      line(40,300,540,300); line(40,437,540,437);
      fill(0);
    }
  }
}
```

Pliki do pobrania:
http://myinventions.pl/013/Akcelerometr_wykres_A.pde
http://myinventions.pl/013/Akcelerometr_wykres_P.pde

Przykładowy wykres otrzymany z pomiarów:



Niestety ze względu na mała wydajność aplikacji java pod kątem aktualizacji obrazu, w tym przypadku możliwa do uzyskania jest tylko częstotliwość nanoszenia próbek na wykres równa około 10Hz.

Obracanie obiektem 3D w Processingu.

Podane niżej kody pozwolą wykorzystać akcelerometr do obracania obiektem 3D w czasie rzeczywistym. Ze względu na mała wydajność w przypadku przesyłania danych 10bitowych (uzyskałem częstotliwość odświeżania obrazu tylko około 16Hz) zastosuję prosty przesył danych 8 bitowych.

Kod do Arduino:

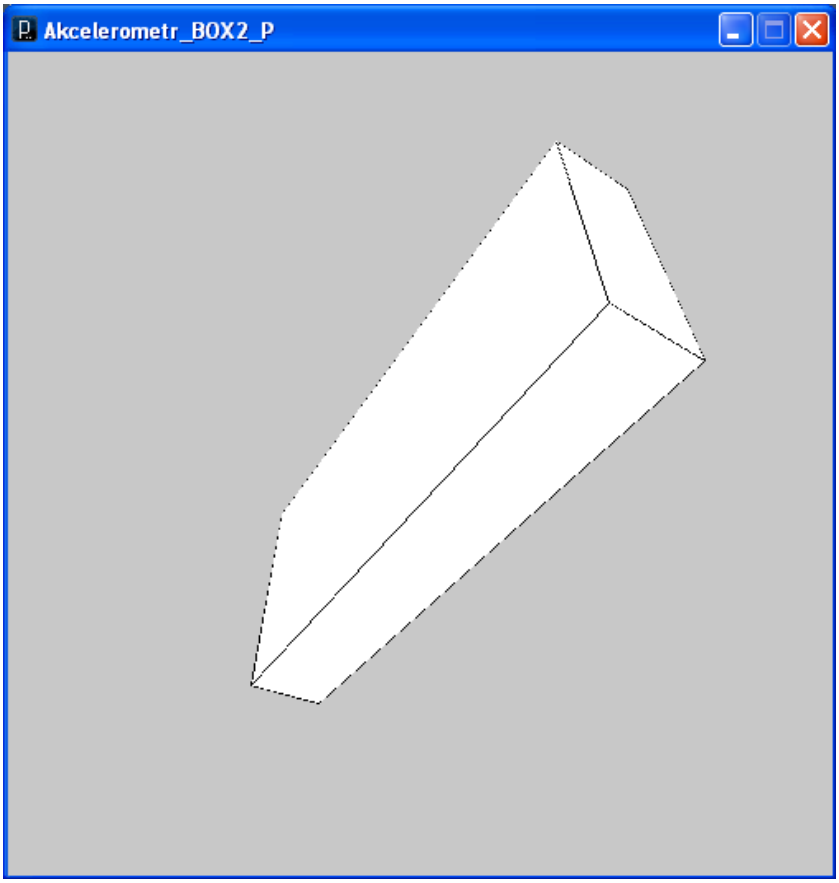
```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print(analogRead(0)/4, BYTE);
  Serial.print(analogRead(1)/4, BYTE);
  delay(40);
}
```

Kod do Processingu:

```
import processing.serial.*;
Serial myPort;
float Y, Z;
boolean yz=true;
float Ry, Rz;
void setup() {
  size(500, 500, P3D); // P3D uruchamia renderer do obiektów 3D
  myPort = new Serial(this,"COM4", 9600);
}
void draw() {
  if ( myPort.available() > 0) {
    if (yz==true){
      Y=myPort.read()-82;
      Ry=Y/20;
```

```
      yz=false;
    }
    else {
      Z = myPort.read()-86;
      Rz=Z/20;
      yz=true;
    }
  }
  background(200);
  translate(width/2, height/2);
  rotateX(Ry);
  rotateZ(Rz);
  box(50,350,150);
}
```

Polecenie translate(width/2, height/2) powoduje przeniesienie układu odniesienia na środek okna roboczego (obowiązuje tylko do końca aktualnej pętli). Polecenia rotateX(), RotateY() i RotateZ() pozwalają obracać układ współrzędnych wokół osi tego układu w kierunku zgodnym z ruchem wskazówek zegara o kąt podany w radianach. Zrzut ekranu z opisanego programu:



Pliki do pobrania:
http://myinventions.pl/013/Akcelerometr_Box_A.pde
http://myinventions.pl/013/Akcelerometr_Box_P.pde

Prosta gra w Blenderze z zastosowaniem akcelerometru.

Stosując metodę przesyłania danych pomiędzy Arduino a Blenderem opisaną w artykule 010 stworzyłem prostą grę, w której do odbijania piłeczki zastosowany jest ruchomy krążek sterowany ruchem akcelerometru.

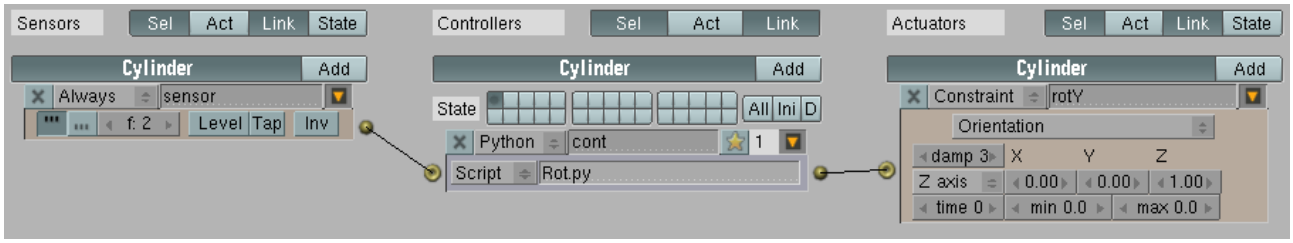
Kod do Arduino:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print(analogRead(0)/4, BYTE);
  Serial.print(analogRead(1)/4, BYTE);
  delay(40);
}
```

Skrypt Pythona do zapisu danych z portu szeregowego do pliku:

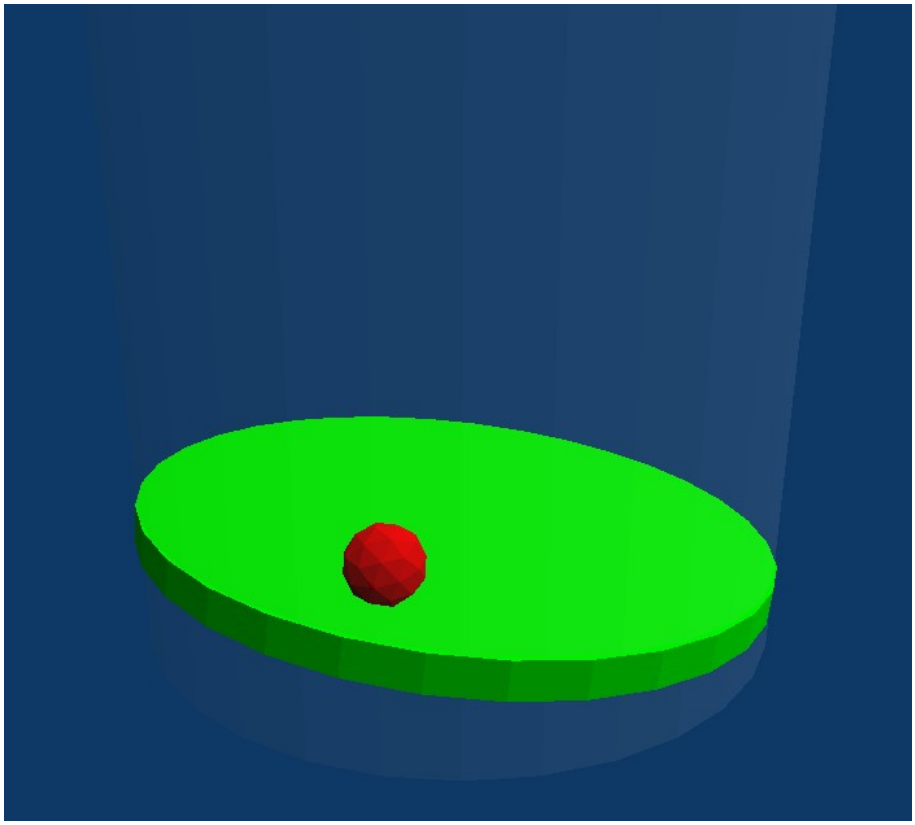
```
import serial
y=0
z=0
port = serial.Serial('COM4', 9600)
for i in range(0, 1500):
    dane=open('daneYZ.txt', 'r+b')
    y=port.read(size=1)
    z=port.read(size=1)
    dane.write(y+z)
    dane.close()
port.close()
```

Widok połączeń logiki i skrypt do Blendera:



```
import GameLogic
import math
daneYZ=open('daneYZ.txt', 'rb')
y=daneYZ.read(1)
daneYZ.seek(1, 0)
z=daneYZ.read(1)
daneYZ.close()
contr = GameLogic.getCurrentController()
rotate = contr.actuators["rot"]
Ry = float(ord(y)-85)/50
Rz = float(ord(z)-85)/50
Vz = math.cos(Ry)
Vy = math.sin(Ry)
Vx = math.sin(Rz)
rotate.direction = [Vx, Vy, Vz]
contr.activate(rotate)
```

Do realizacji zmiany orientacji obiektu użylem aktuatora Constraint/Orientation. Wybrana w nim oś Z zostaje obrócona względem swojego początkowego punktu tak, aby przeciąć punkt o współrzędnych (X,Y,Z), które zadawane są za każdym wywołaniem skryptu poleceniem: rotate.direction = [Vx, Vy, Vz]. Ustalone tłumienie (damp) pozwala zminimalizować drgania sterowanego obiektu wywołane gwałtownymi ruchami czujnika.



Komplet plików do pobrania:

http://myinventions.pl/013/daneYZ.txt

http://myinventions.pl/013/daneYZ.py

http://myinventions.pl/013/Akcelerometr.blend

Więcej o poleceniach pozwalających korzystać z aktuatora Constraint:

http://www.blender.org/documentation/249PythonDoc/GE/GameTypes.KX_ConstraintActuator-class.html

Filmik prezentujący działanie opisanego układu: http://www.vimeo.com/6018686

Do pobrania plik z grą w odbijanie piłeczki sterowaną za pomocą ruchu myszy:

http://myinventions.pl/013/BoxMouse.blend