POLITECHNIKA WROCŁAWSKA

Wydział mechaniczny

Interdyscyplinarny projekt zespołowy

PROJEKT:

Robot kroczący dwunożny

Skład grupy:

Alicja Pernal – 261290

Anna Nawara – 261335

Wojciech Lipowicz – 261298

Franek Morawa – 261297

Semestr 6

rok akademicki 2022/2023

Wrocław

1. Wstęp

Wykonanie projektu polega na zaprojektowaniu i wykonaniu robota, zdolnego do poruszania się, poprzez wykonywanie kroków. Robot jest sterowany z wykorzystaniem płytki Arduino, a jego napęd stanowi 6 mikro serw.

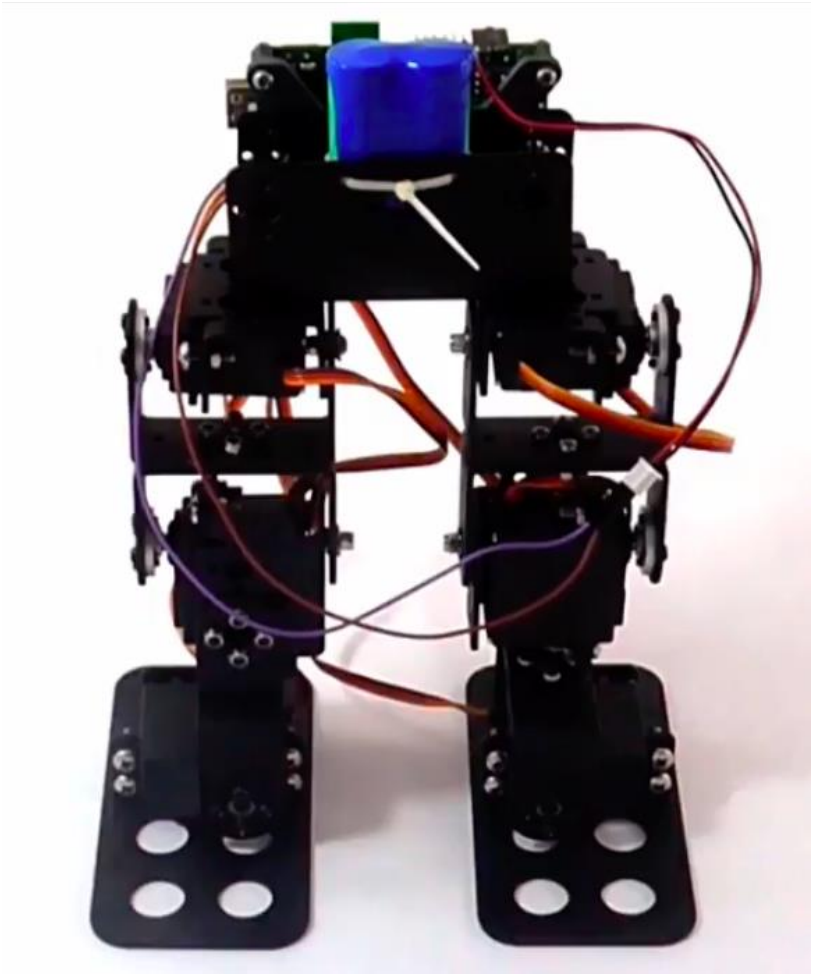
Raport zawiera:

- Przegląd rozwiązań
- Opis procesu wykonanie robota, składający się z części mechanicznej, elektronicznej i programistycznej
- Podsumowanie
- Przewodnik po githubie

2. Przegląd rozwiązań

Przykłady budowy robotów kroczących:

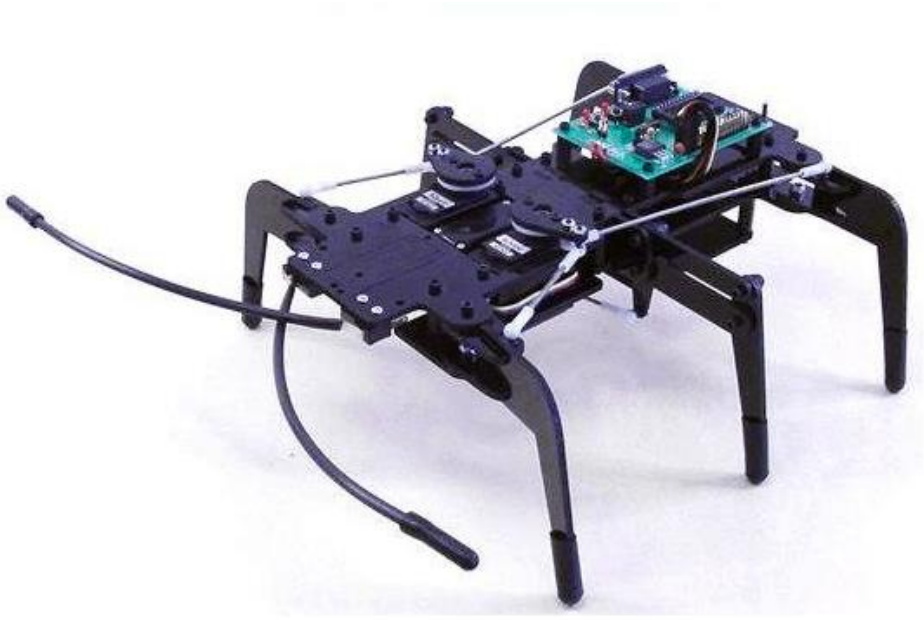
1. **Robot dwunożny, poruszający się przez odchylanie nóg:**



Rysunek 1

Ten robot porusza się poprzez unoszenie nóg na boki połączone z odchylaniem stóp do środka, co zmniejsza jego stabilność oraz nie jest zbyt naturalnym sposobem chodu.

2. **Robot sześcionożny:**



Rysunek 2

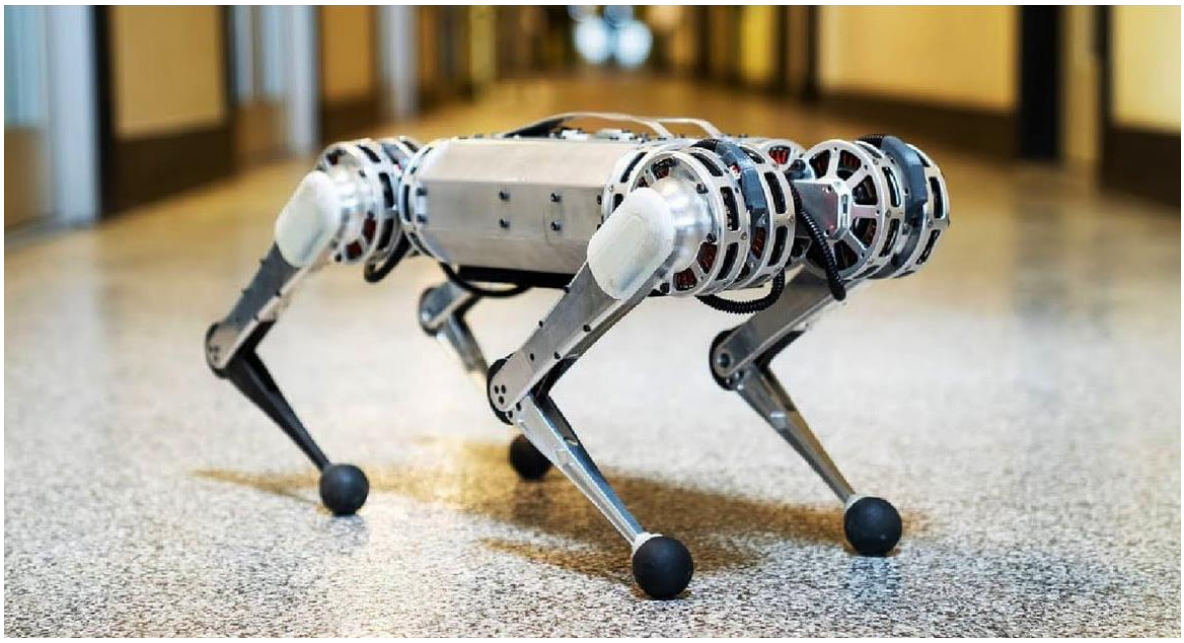
Taki robot jest wzorowany na owadach. Jest on niewątpliwie bardziej stabilny od poprzedniego, którego zaprezentowaliśmy, jednak jest też bardziej skomplikowany, przez zwiększoną ilość kończyn, co sprawia, że wymaga on więcej napędów.

3. **Robot jednonożny - skaczący:**

Rysunek 3

Robot ten porusza się skacząc, co wymaga, aby ciągle był on w ruchu. Taki robot nie jest zbyt stabilny oraz wymaga skomplikowanego algorytmu sterowania. Poza tym, fakt, że nie może on zatrzymać się w stabilnej pozycji powoduje, że robot łatwo może zostać uszkodzony, jeśli wystąpi jakiś błąd. Tą koncepcję również odrzucamy.

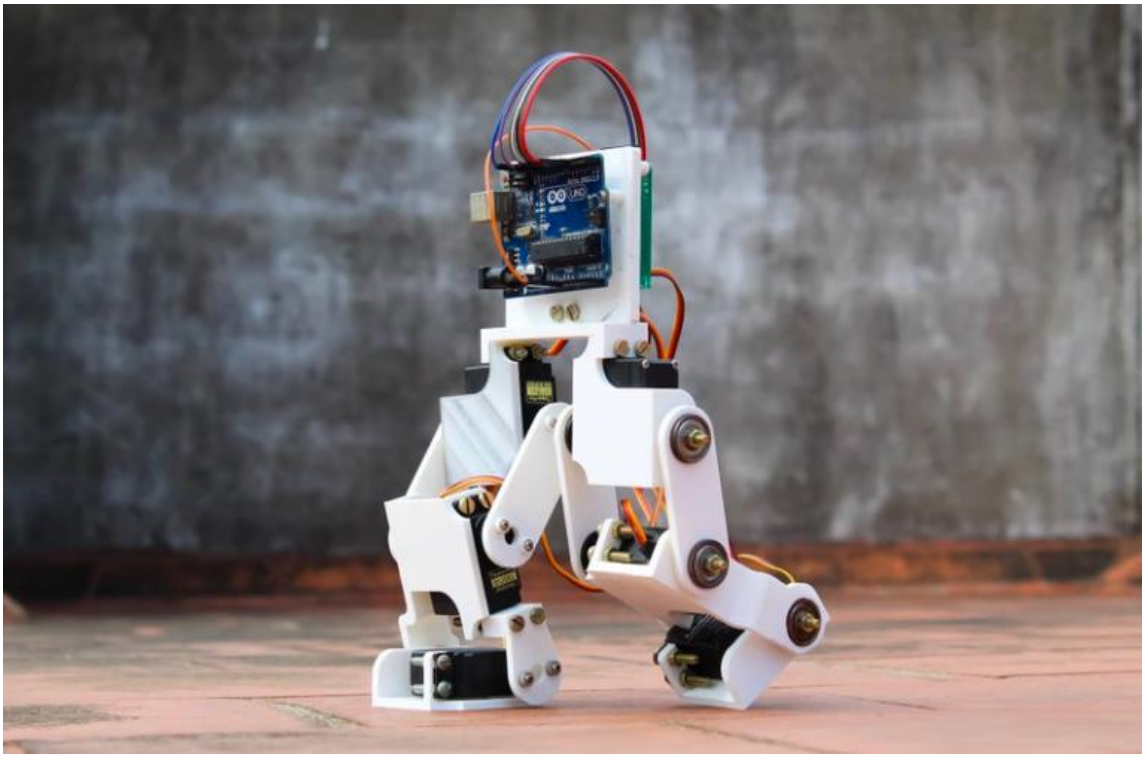
4. **Quadropedy – roboty o czterech nogach:**



Rysunek 4

Taki robot jest wzorowany na zwierzętach czworonożnych i może się poruszać na wiele sposobów, w zależności od sposobu wykonania jego nóg. Podobnie jak w przypadku robota sześcionożnego, jest on bardziej skomplikowany i wymaga więcej napędów niż robot humanoidalny.

5. **Robot dwunożny, ze stawami kolanowymi:**



Rysunek 5

Jest to robot o dwóch nogach, tak samo jak w koncepcji 1, jednak w tym przypadku, robot porusza się uginając nogi w kolanach, przez co jego chód jest bardziej stabilny i zbliżony do ludzkiego. Jest to koncepcja, którą zamierzamy stosować w naszym projekcie.

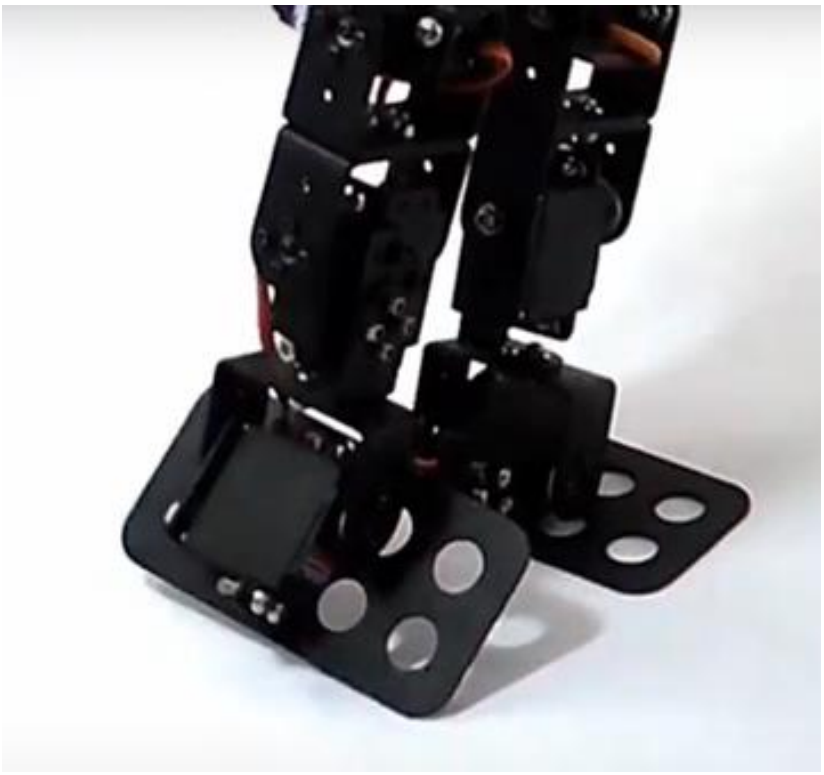
Konstrukcja robota kroczącego dwunożnego:

1. **Krok**

Aby robot mógł wykonywać kroki zbliżone do ludzkiego chodu, jego nogi muszą być zbudowane podobnie do ludzkich, tzn. każda noga musi być podzielona na 3 części (odpowiedniki stopy, łydki i uda) oraz zawierać 3 napędy imitujące ruchome stawy.

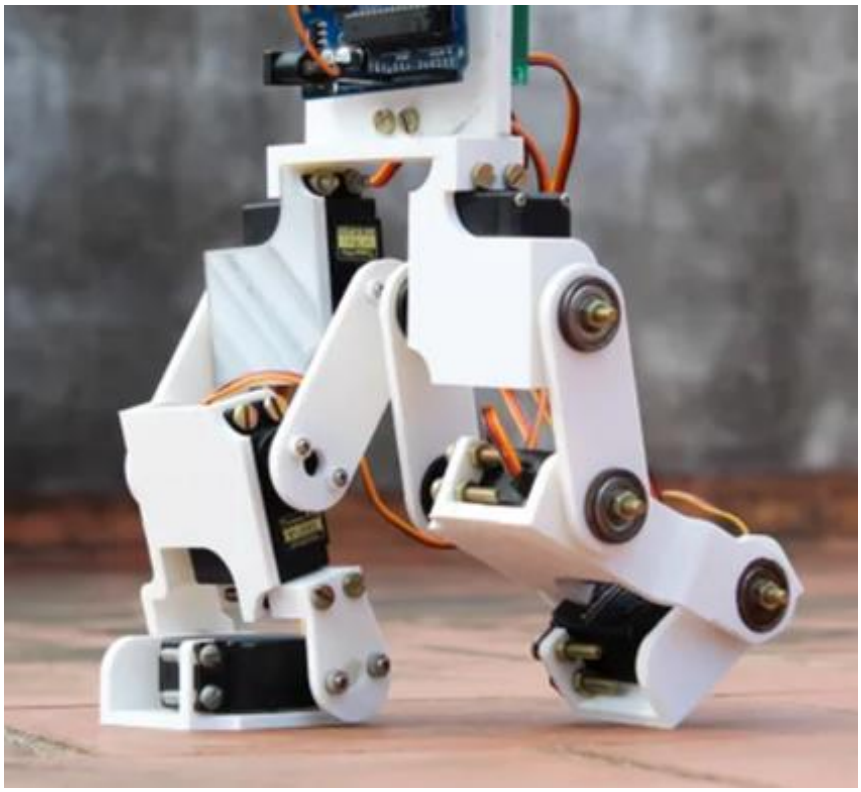
A. **Stopa:**

Rozważaliśmy dwa możliwe ułożenia stopy. W pierwszym z nich, stopa podczas podniesienia nogi odginałaby się na bok od chodu. W tym rozwiązaniu robot prawdopodobnie nie potykał się o przednią część stopy, jednakże wymagałoby to bardziej skomplikowanej konstrukcji unoszenia nogi, która wiązałaby się z kołysaniem się robota na lewo i prawo.



Rysunek 6

Druga możliwość zakłada przód stopy poruszający się góra dół. Może to spowodować konieczność wyższego podnoszenia „kolan” robota, ale powinno zwiększyć stabilność i elegancję jego kroku.



Rysunek 7

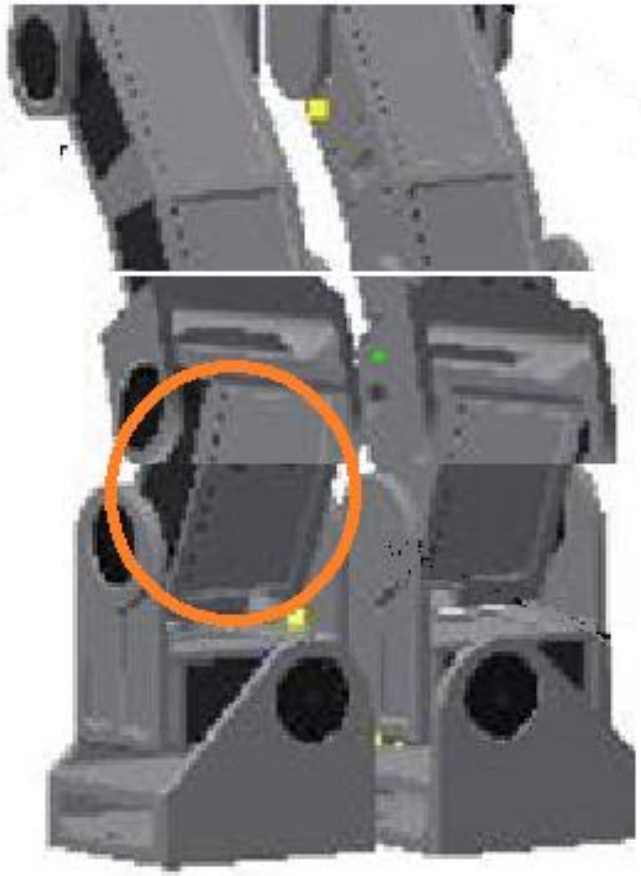
B. Kolano

Przy konstrukcji kolana największe wyzwanie stanowi sposób zamocowania „łydki” z „udem” poprzez servo napęd.

Głównym problemem jest tu umiejscowienie napędu w odpowiednim miejscu. Większość robotów tego typu, ma bardzo podobną konstrukcję nóg. Na przykładowym schemacie, serwonapęd znajduje się w dolnej części uda

Rysunek 8

Sam Napęd będzie stosunkowo ciężkim elementem przy częściach wydrukowanych filamentem, dlatego zdecydowaliśmy się na umiejscowienie napędu możliwie jak najniżej (będzie osadzony w łydce), aby obniżyć środek ciężkości robota.



Rysunek 9

C. Stawy

Sposób ruchu poszczególnych elementów bardzo zależy od tego jak będą wyglądać elementy obrotowe poszczególnych części robota.

Śruby - jest to najtańsze z możliwych rozwiązań, ale mogłyby powodować rozbieżności w dokładności chodu robota oraz zacinać się o szorstką powierzchnie drukowanych części.

Rysunek 10

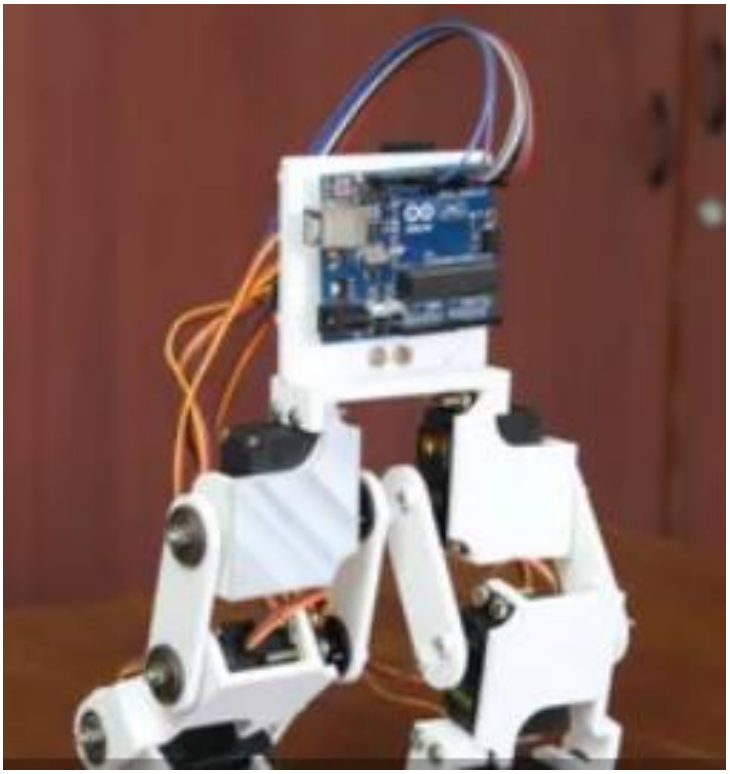
Łożyska - są droższe, jednakże dużo łatwiej jest je połączyć z ruchomymi częściami servo napędów. Odpowiedni rozmiar okazały się mieć łożyska przeznaczone do deskorolki.

Rysunek 11

2. Tors

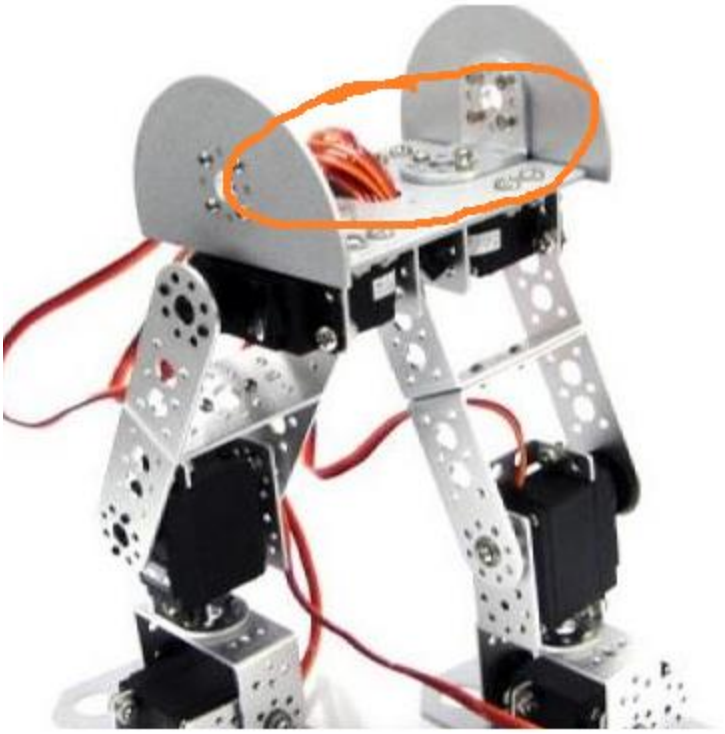
Tors będzie miał za zadanie trzymanie płytki arduino i sensorów w robocie. Można go zamocować w pozycji pionowej lub poziomej.

Tors zamocowany w pozycji pionowej wygląda lepiej i znacznie ułatwia dostęp do płytki, co ułatwi ewentualne wprowadzenie zmian w projekcie i naprawy wszelkiego rodzaju błędów w podpięciu układu.



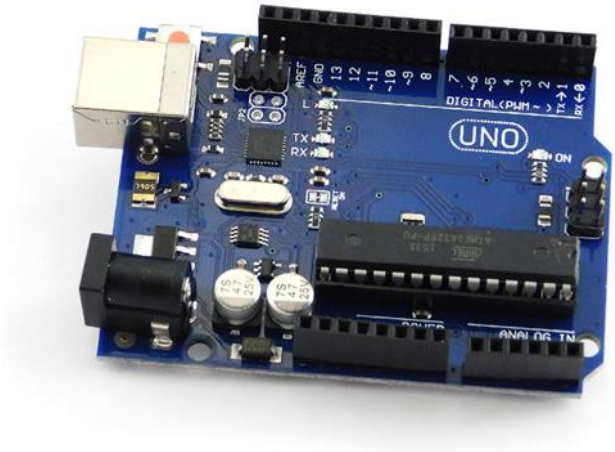
Rysunek 12

Tors zamocowany w pozycji poziomej spowoduje obniżenie środka ciężkości robota oraz zwiększy jego stabilność. Docelowo chcemy zamocować tors pionowo, ale jeżeli będzie taka konieczność, możemy zmienić tą koncepcję.



Rysunek 13

Do sterowania serwomotorami można wykorzystać mikrokontroler np. Arduino czy Raspberry Pi lub kontroler serwa, który jest właściwie mikrokontrolerem dedykowanym do obsługi serw. Zdecydowaną wadą wykorzystania kontrolera jest jego cena, która jest znacznie większa niż np. Arduino, a w przypadku tańszych opcji kontrolerów wymagają one współpracy z mikrokontrolerem, dlatego cena ponownie rośnie. Zatem użycie mikrokontrolera (w naszym przypadku Arduino) jest najbardziej uzasadnione ze względu na możliwość sterowania wymaganą ilością 6 serw używanych w „stawach" robota oraz obsługa czujników.

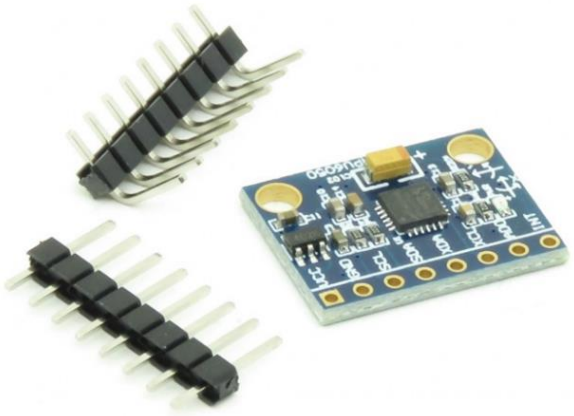


Rysunek 14

Rysunek 15

Żeby robot był „świadom” otaczającego go świata należy zastosować odpowiedni czujnik do wykrywania przeszkód itp. Działanie robota pod tym względem skupia się głównie na pomiarze odległości od przeszkody. Przykładowymi sensorami kompatybilnymi z Arduino są czujniki ultradźwiękowe, indukcyjne, optyczne w tym czujniki laserowe, Lidar czy też czujniki działające w paśmie podczerwieni. Czujniki indukcyjne można wyeliminować ze względu na ich późny czas działania w znaczeniu, że reagują one dopiero tuż przed przeszkodą. Wybór czujników ultradźwiękowych ponad czujniki optyczne pozwoli na ograniczenie kosztów bez utraty wymaganych przez robota cech wykrywania przeszkód. Kolejnymi czujnikami wymaganymi przez robota będzie żyroskop oraz akcelerometr.

Rysunek 16

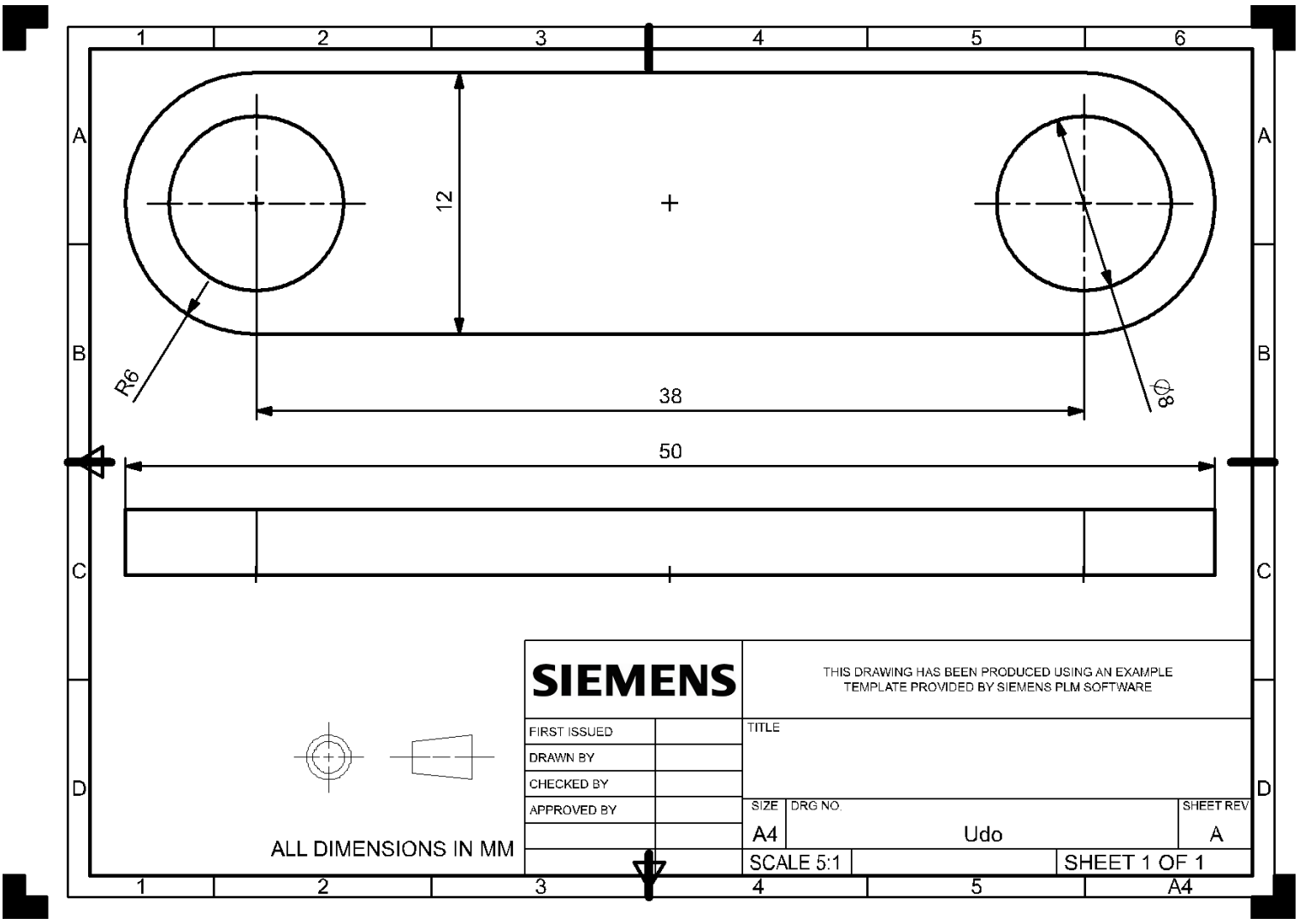


Rysunek 17

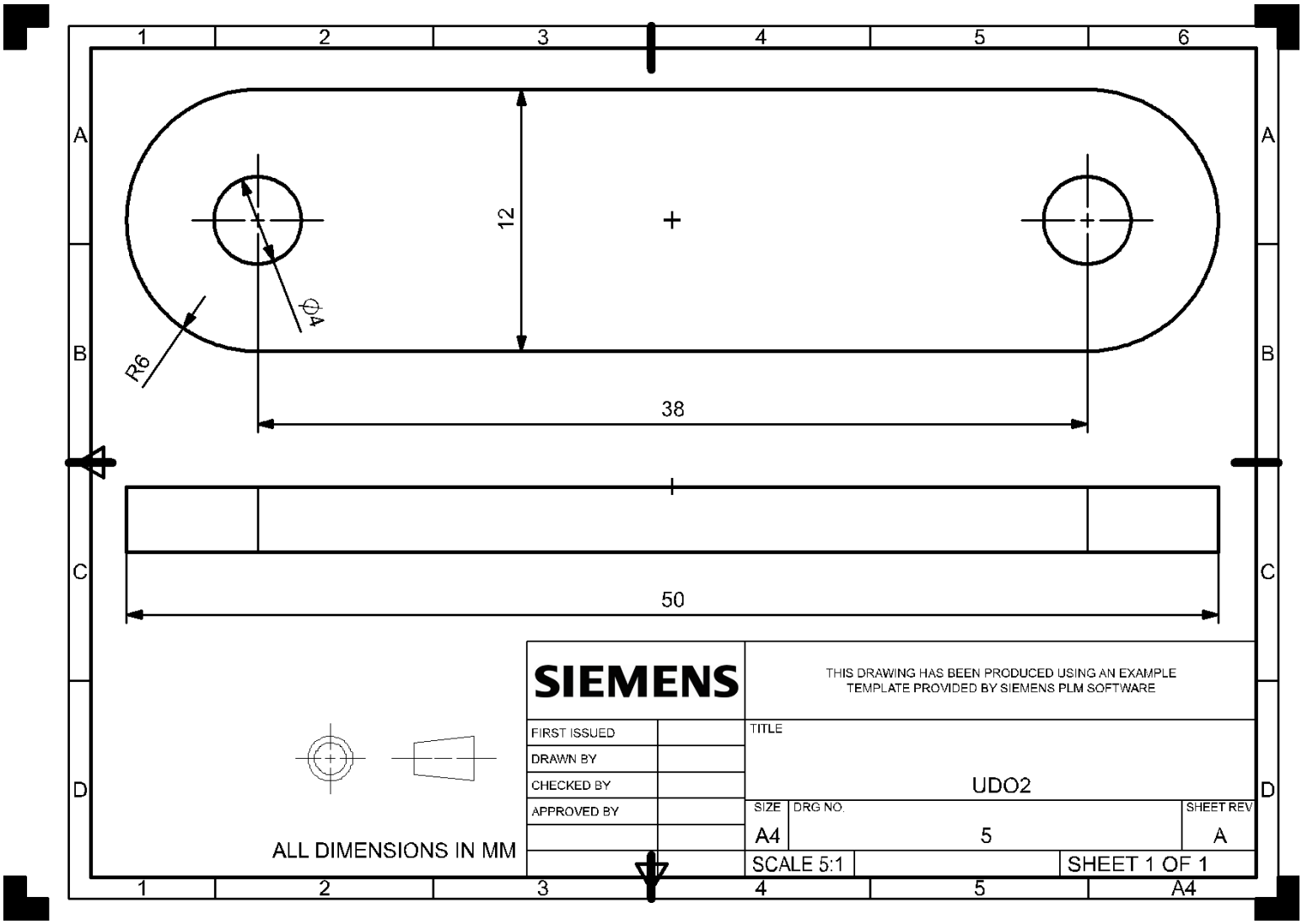
3. Wykonanie projektu - część mechaniczna, model

Wykonanie robota rozpoczęliśmy od zamodelowania w programie NX elementów składowych, które stworzą korpus robota. Elementy te zostały przedstawione na poniższych rysunkach:

Udo x2:

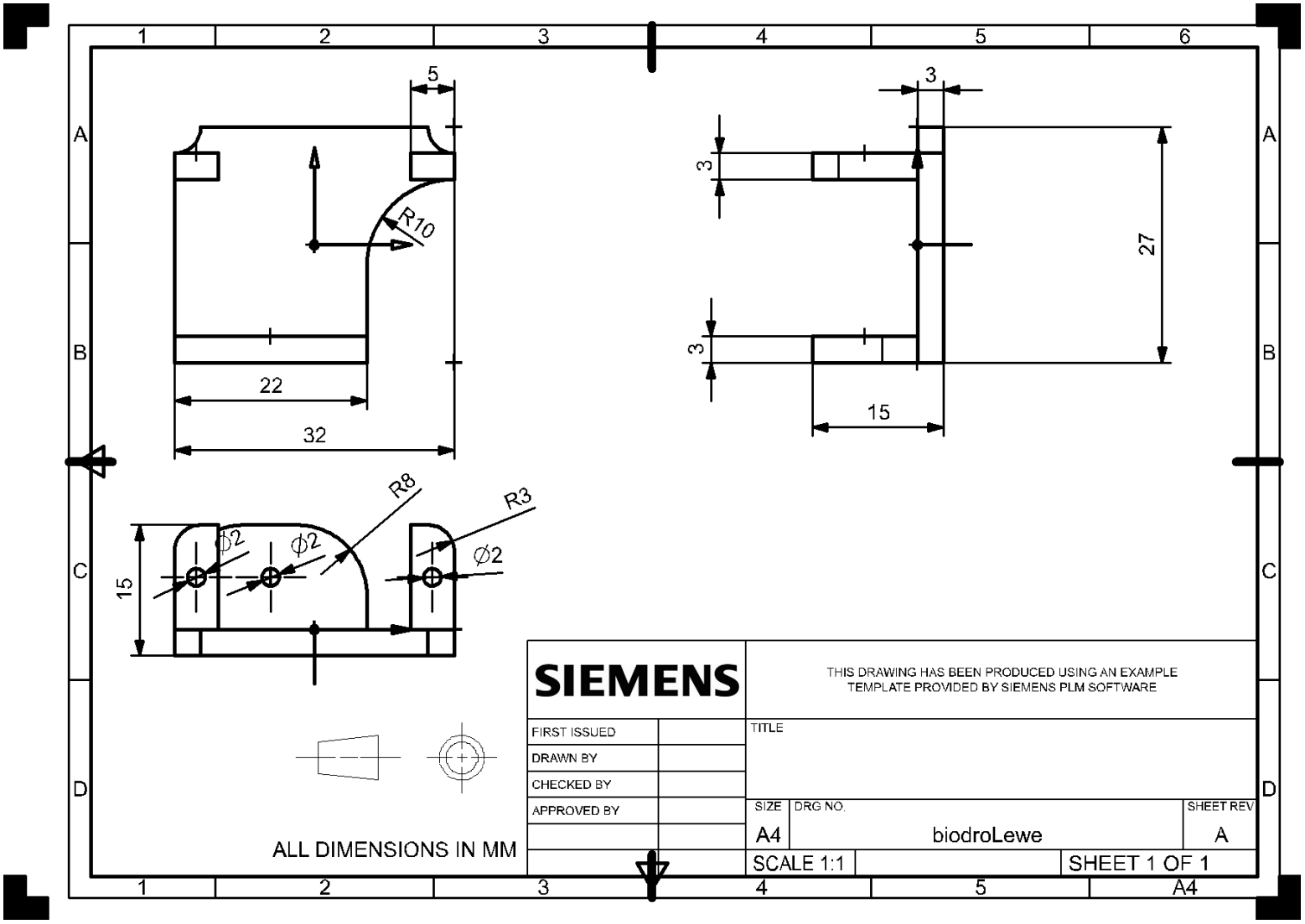


Rysunek 18


1
2
3
4
5
6
A
B
C
D
12
38
50
R6
Φ4
SIEMENS
THIS DRAWING HAS BEEN PRODUCED USING AN EXAMPLE TEMPLATE PROVIDED BY SIEMENS PLM SOFTWARE
FIRST ISSUED
DRAWN BY
CHECKED BY
APPROVED BY
TITLE
UDO2
SIZE
A4
DRG NO.
5
SHEET REV
A
SCALE 5:1
SHEET 1 OF 1
ALL DIMENSIONS IN MM
A4

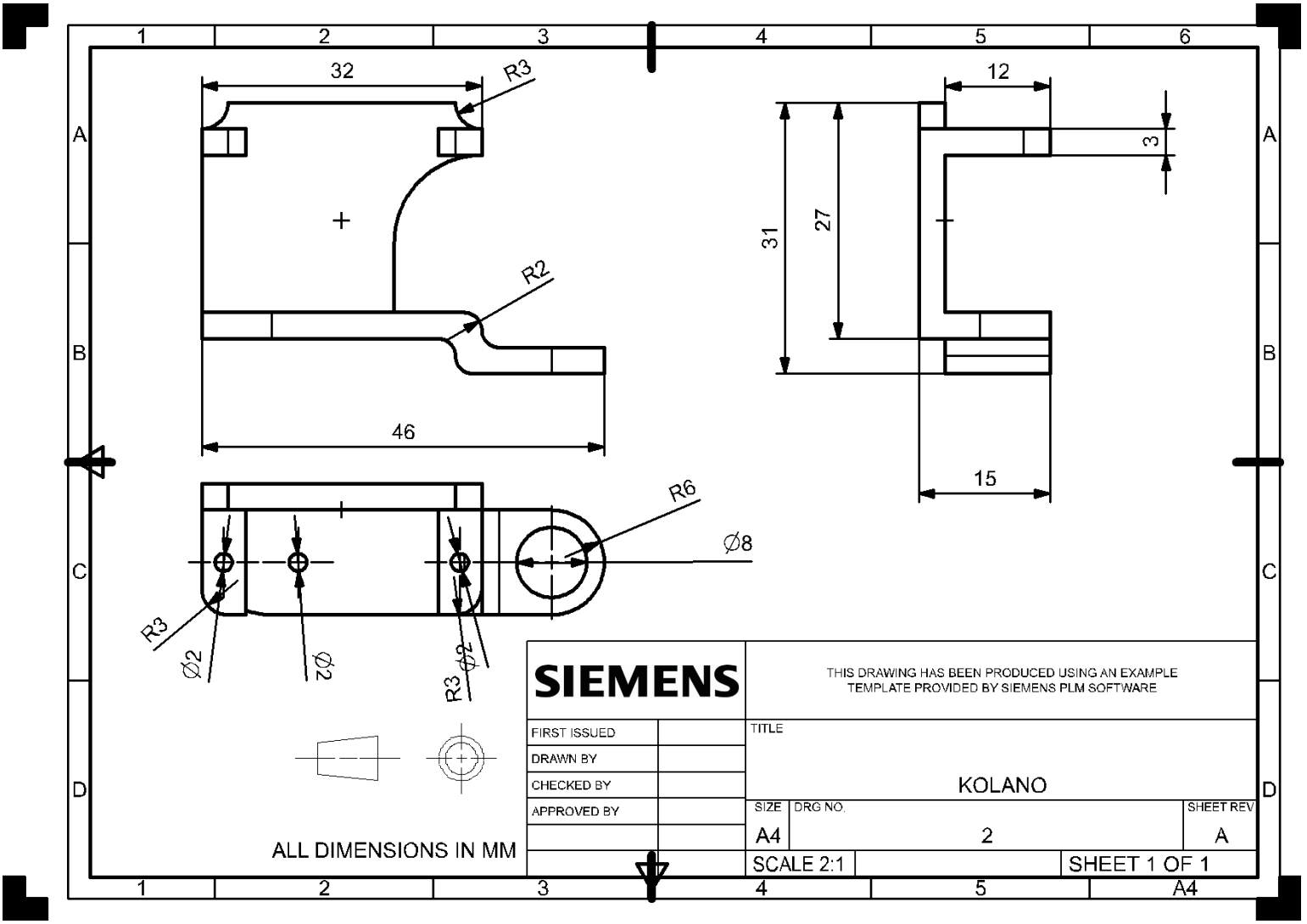
Rysunek 19

Biodro (lewe i prawe):



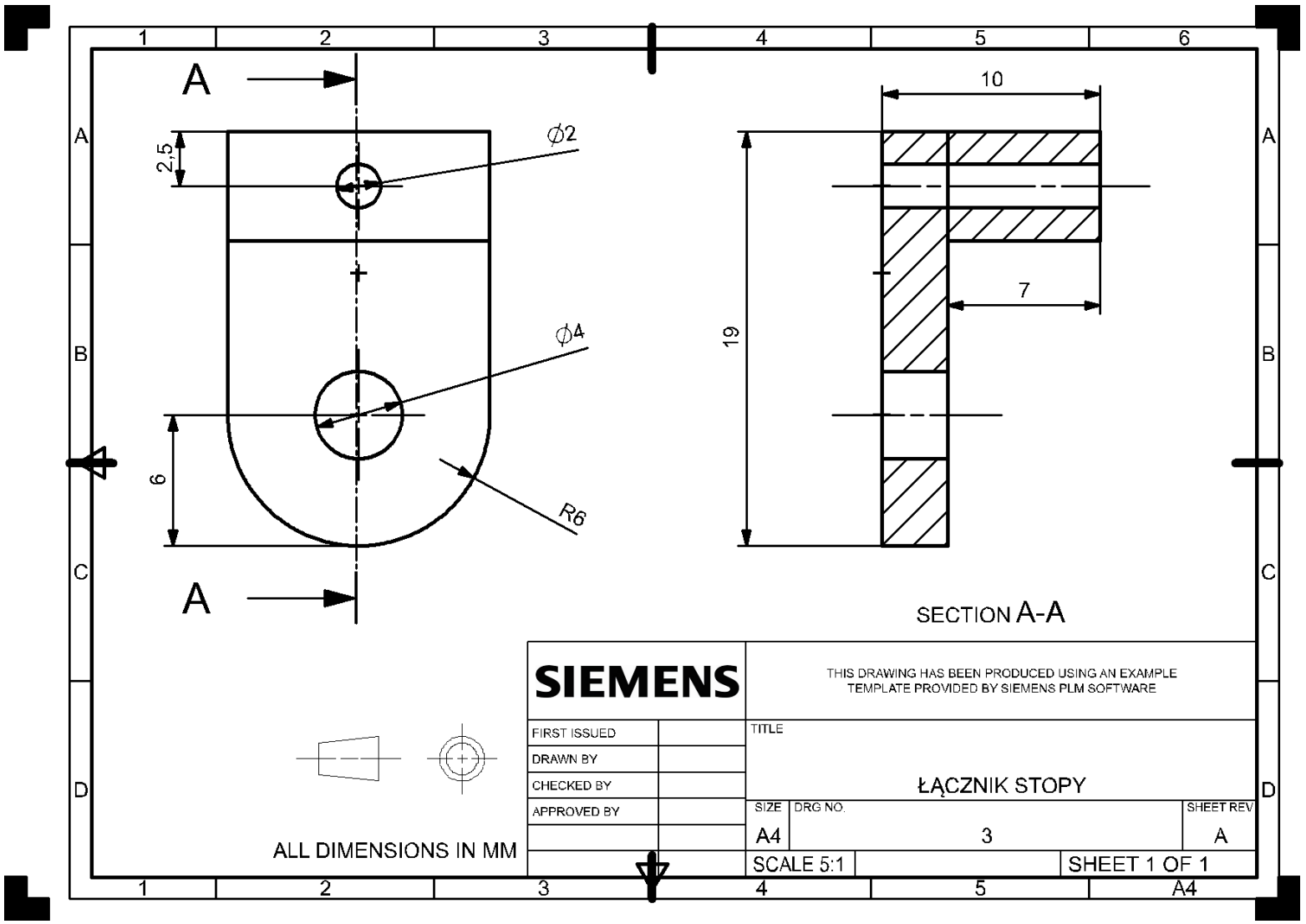
Rysunek 20

Kolano (lewe i prawe):



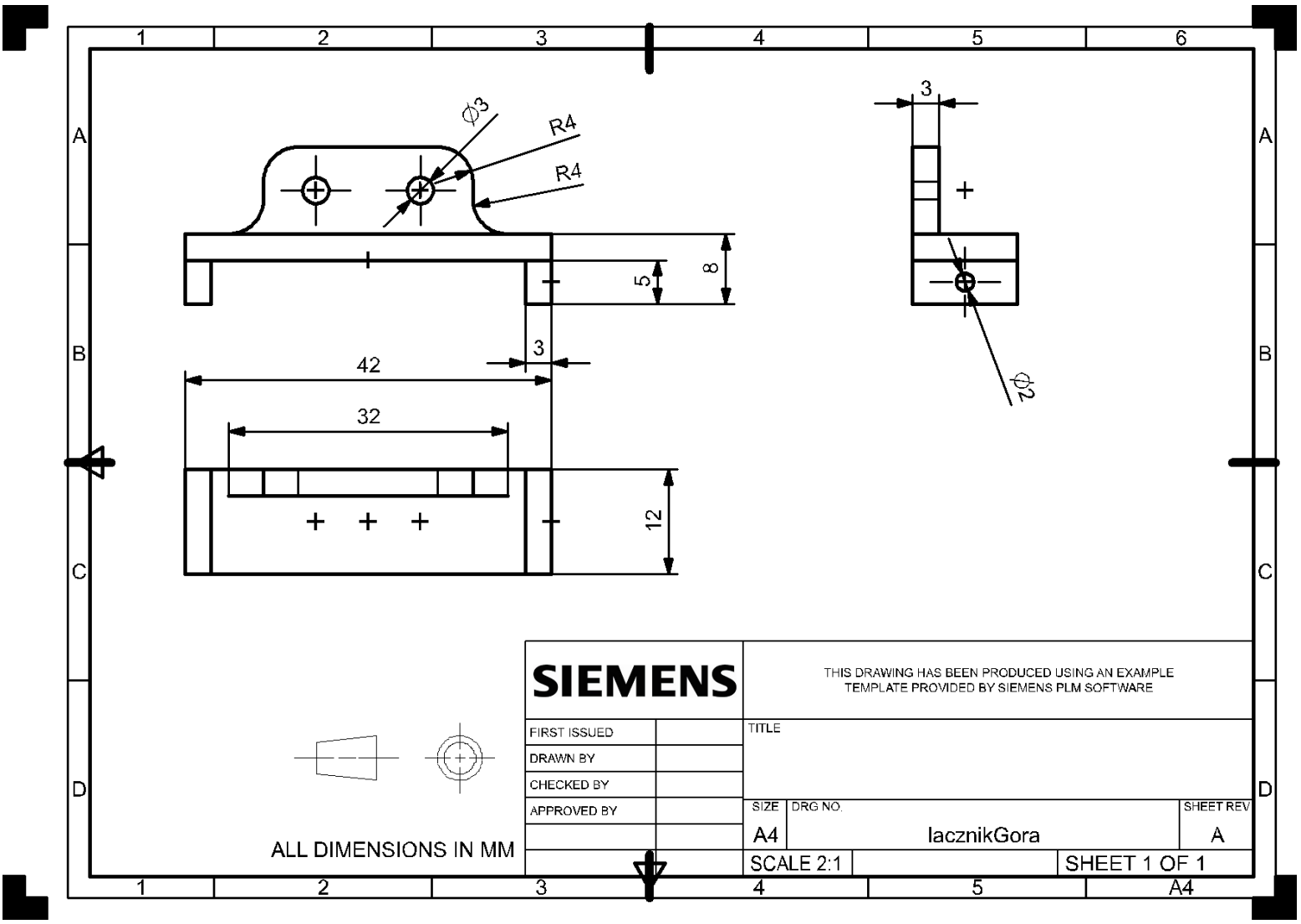
Rysunek 21

Łącznik stopy x2:



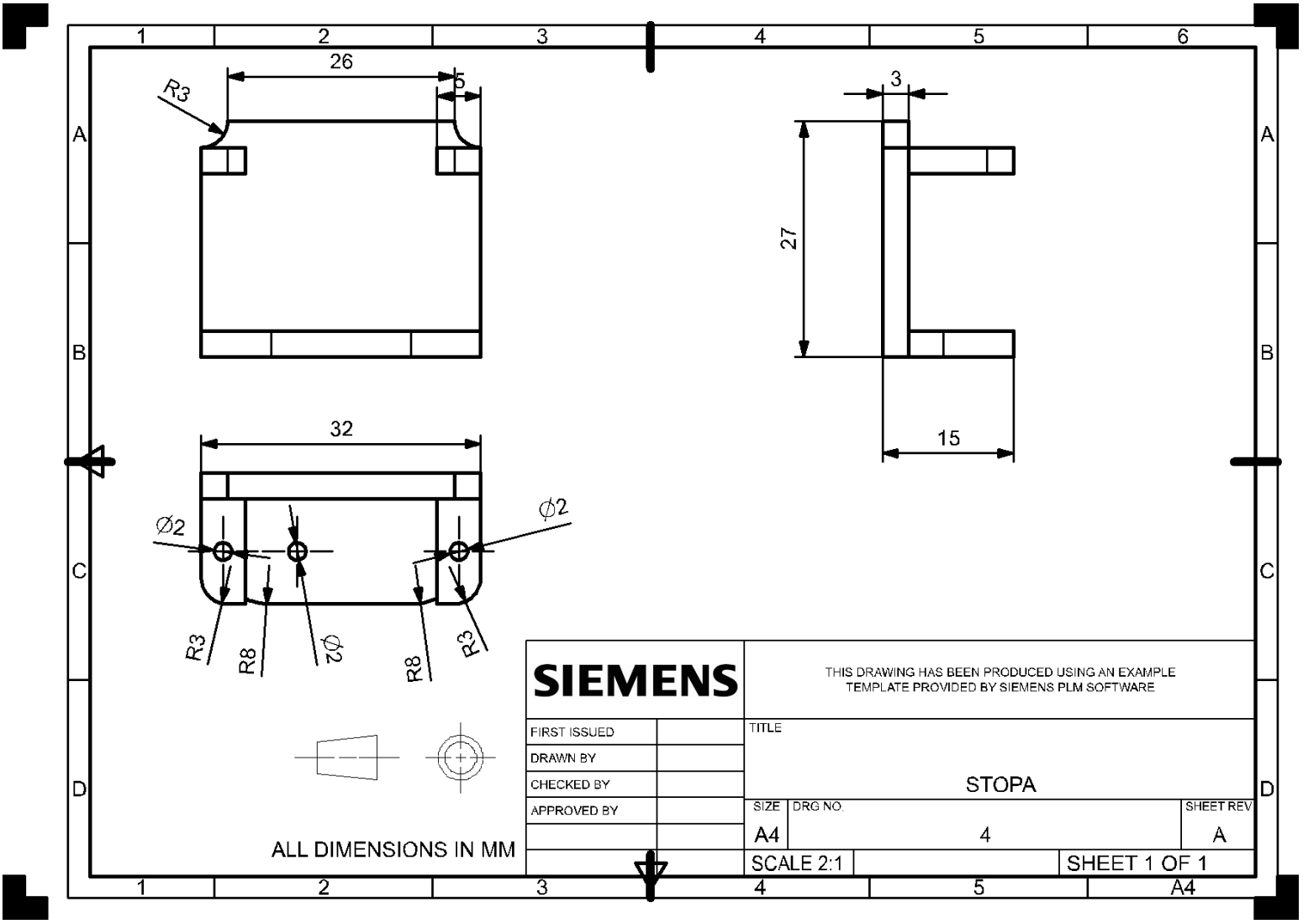
Rysunek 22

Łącznik:



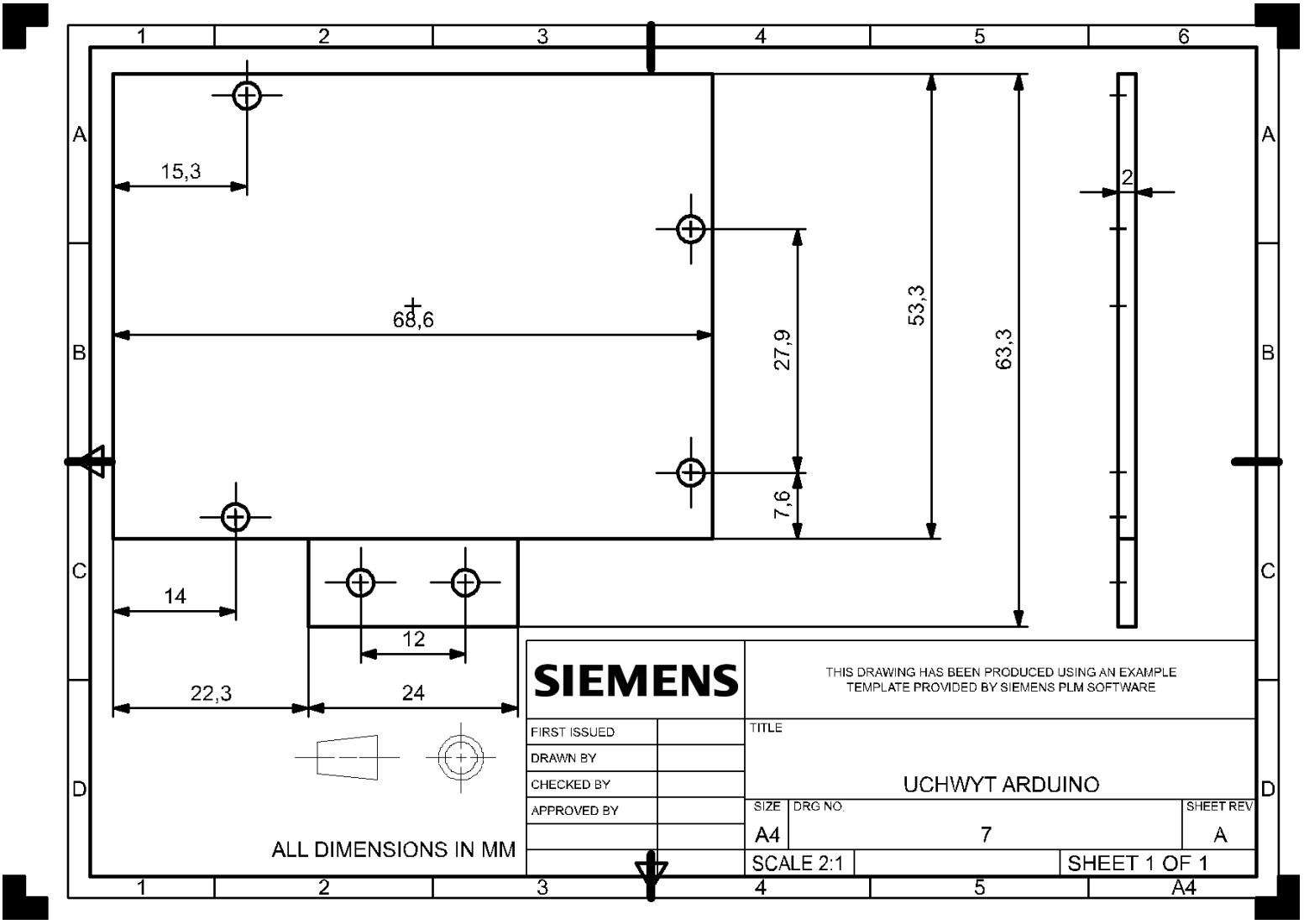
Rysunek 23

Stopa (lewa i prawa):



Rysunek 24

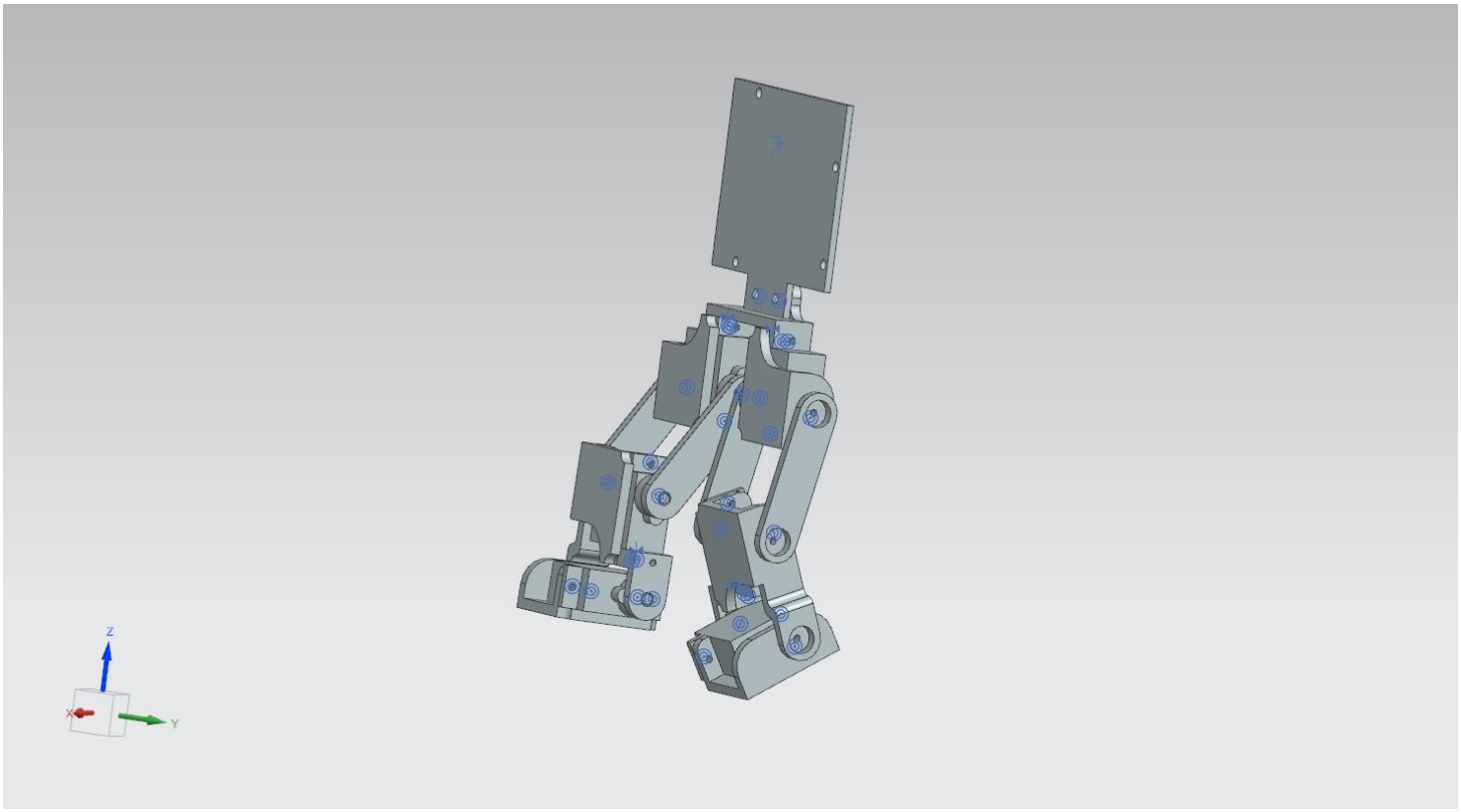
Uchwyt na płytkę Arduino i płytkę stykową:



Rysunek 25

Wymiary tych elementów są dostosowane do wymiarów stosowanych przez nas serw i zawierają otwory pozwalające na zmontowanie ranem elementów (wymiary otworów uwzględniają zachowanie się filamentu przy drukowaniu).

Złożony z tych elementów korpus robota prezentuje się następująco:



Rysunek 26

Elementy zostały wydrukowane na drukarce 3D.

4. Wykonanie projektu – część elektroniczna, układ elektroniczny

Potrzebne elementy:

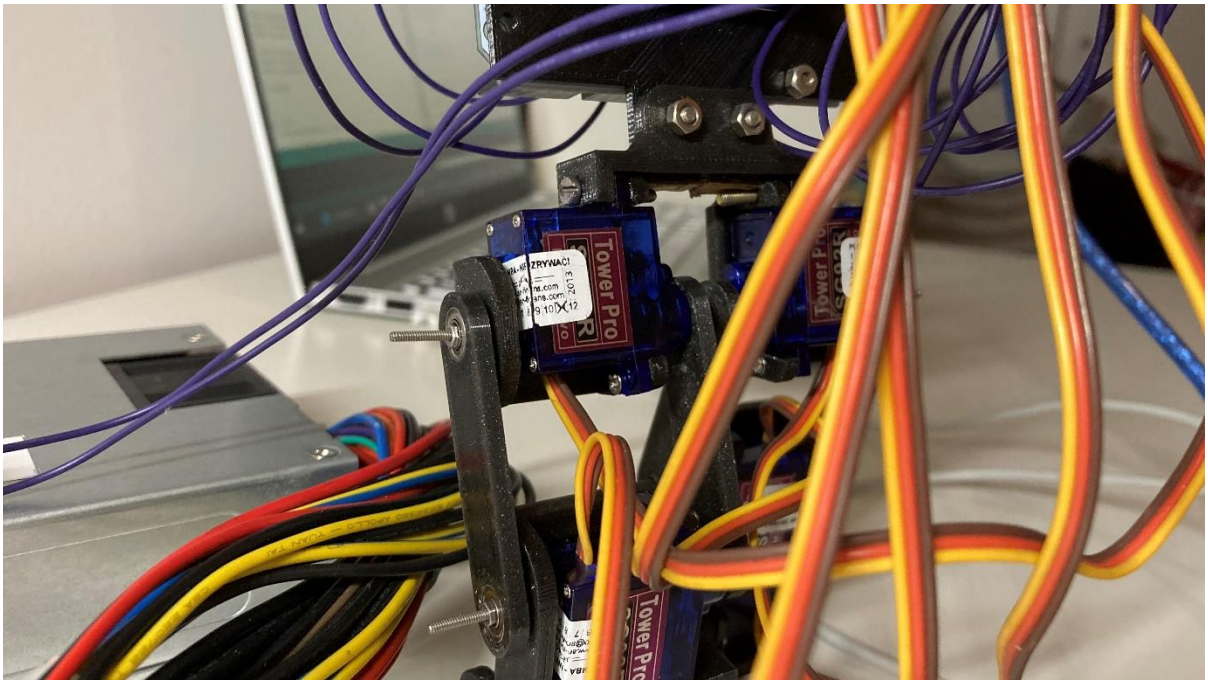
- Arduino Uno
- TowerPro SG92R MicroServo x6
- Płytka stykowa
- Przewody m-m
- Przewody f-f
- Złącze m

Do połączenia tych części z elementami korpusu zastosowaliśmy:

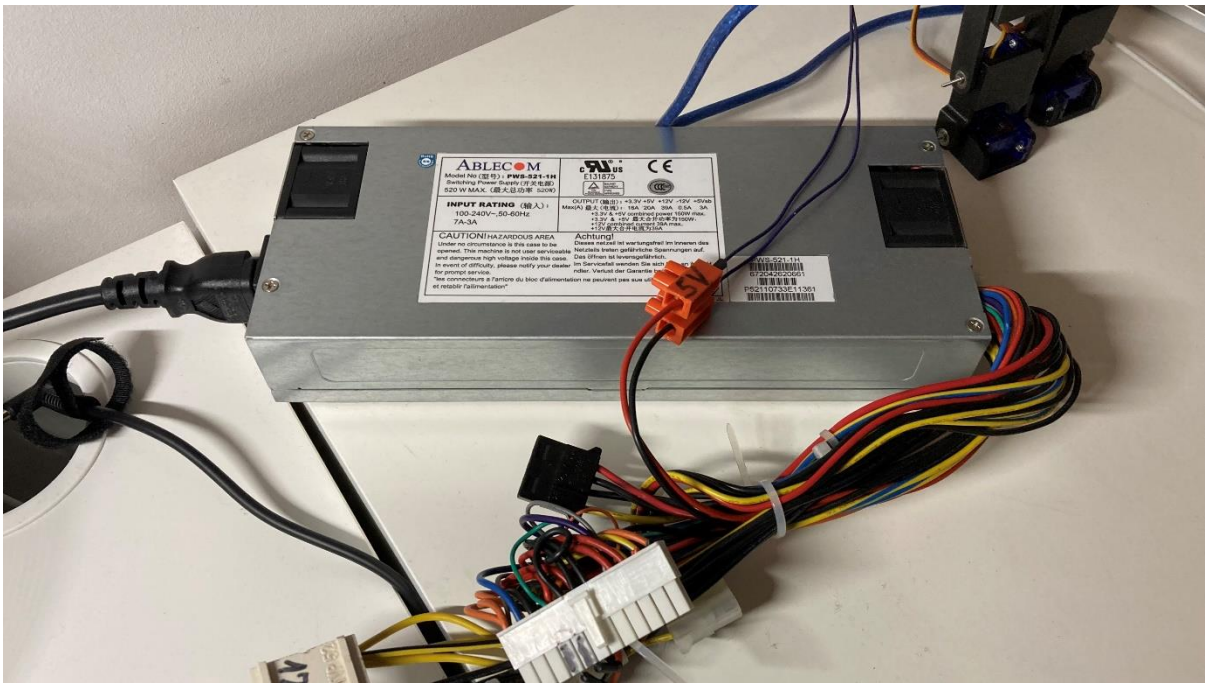
- Łożyska do deskorolki x6
- Nakrętki i śruby M4

Dodatkowo, serw potrzebują zasilania zewnętrzeno, które w tym przypadku dostarczył zasilacz 5V.

Serwa oraz zasilacz przedstawione są poniżej:

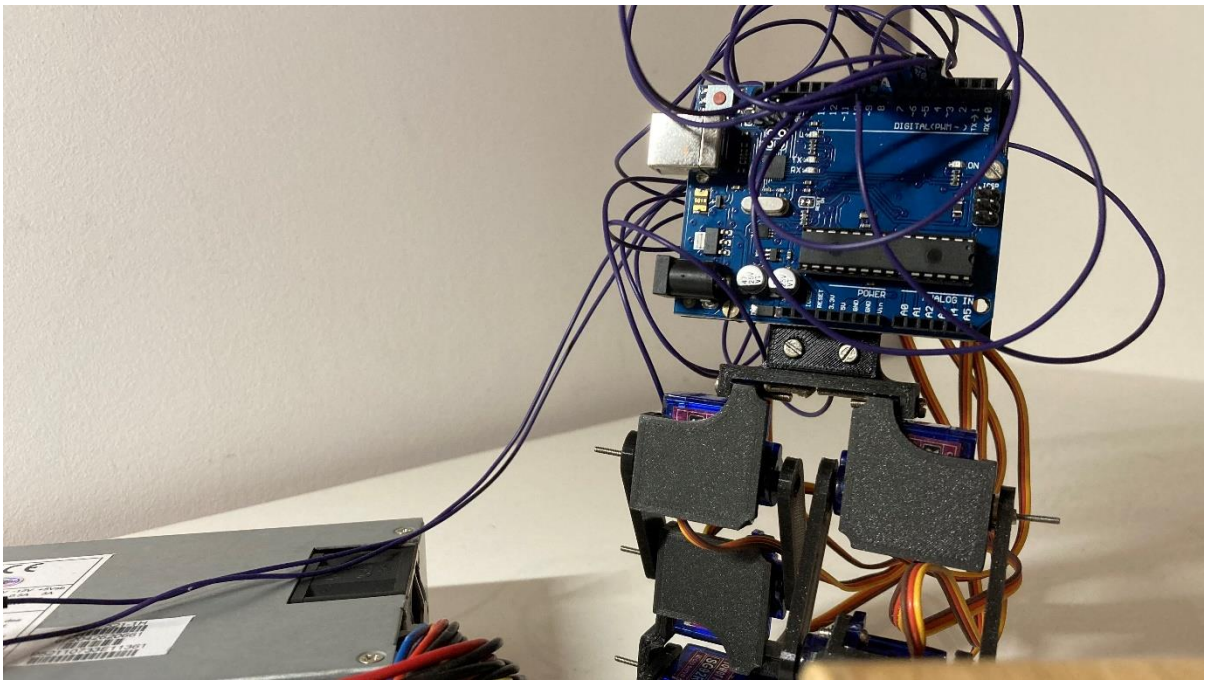


Rysunek 27

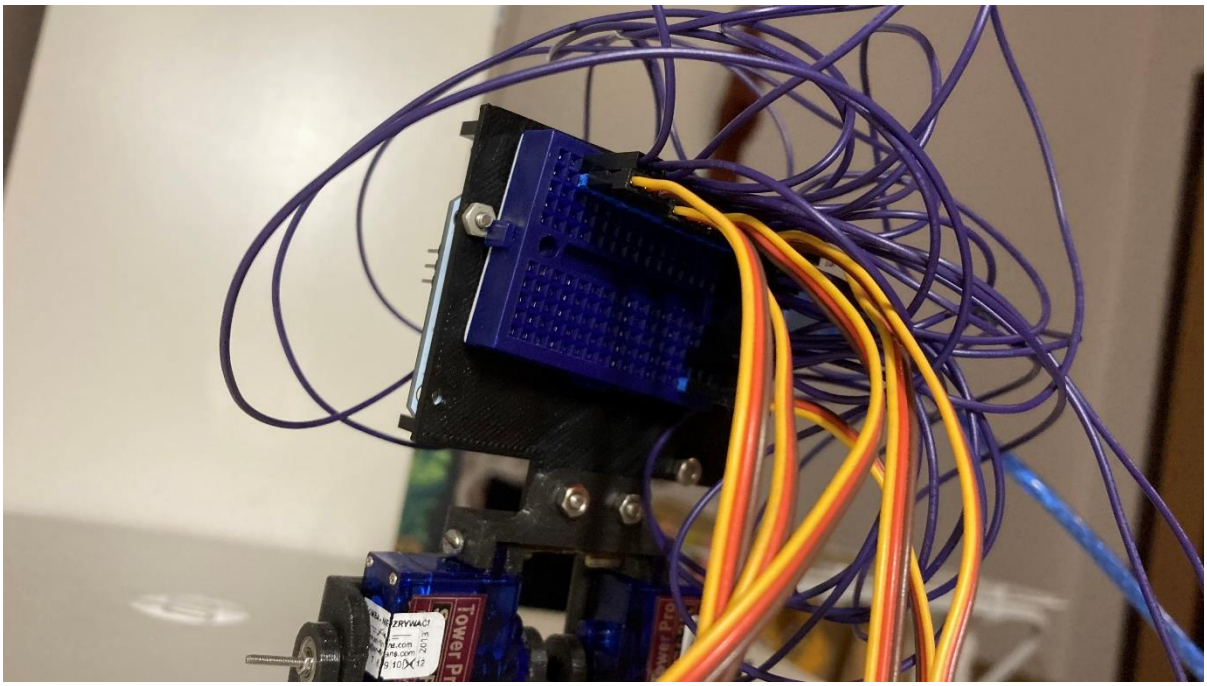


Rysunek 28

Pytka Arduino oraz płytka stykowa:



Rysunek 29



Rysunek 30

Podłączenie układu polega na podłączeniu zasilania („+” i „-”) oraz sygnału sterującego z odpowiedniego kanału (tak, jak zostało przydzielone w programie) i odpowiednim podłączeniu serw według opisu wyprowadzeń:

Czerwone – zasilanie
Brązowe – masa
Pomarańczowe – sygnał

5. Wykonanie projektu – część programistyczna, program sterujący robotem

Program sterujący robotem składa się z dwóch części. Każda z części programu korzysta z pliku nagłówkowego, który zawiera zdefiniowane startowe pozycje kątowe serw (konieczne dla obu programów) oraz zdefiniowane (konieczne do działania programu realizującego chód):

- Długość uda robota
- Długość łydki robota
- Odległość od podłogi do biodra podczas wykonywania kroku
- Wysokość, na jaką robot podnosi nogę

a) Część odpowiedzialna, za ustawienie robota w pozycji pionowej. Ten program jest potrzebny, aby ustalić pozycję startową dla serw, do której będziemy się odnosić w drugiej części programu.
b) Część odpowiedzialna za chód.

Plik nagłówkowy – constants.h:

```
#define hipLOffset 35

#define kneeLOffset 180

#define ankleLOffset 80

#define hipROffset 120

#define kneeROffset 60

#define ankleROffset 85


#define l1 3.8

#define l2 3


#define stepClearance 0.7

#define stepHeight 7
```

Gdzie:

I1 – długość od biodra do kolana
I2 – długość od kolana do stawu skokowego

stepClearance – wysokość uniesienia stopy
stepHeight – odległość od ziemi do biodra podczas kroku

Wartości te będą wymagać zmiany dla innych wymiarów serw, a więc innych wymiarów robota.

Program ustawiający robota w pozycji startowej – initial_setup.ino:

```
#include <Servo.h>
#include "constants.h"

Servo hipL;
Servo hipR;
Servo kneeL;
Servo kneeR;
Servo ankleL;
Servo ankleR;

void setup() {
  // put your setup code here, to run once:
  hipL.attach(9);
  hipR.attach(8);
  kneeL.attach(7);
  kneeR.attach(6);
  ankleL.attach(5);
  ankleR.attach(4);

  hipL.write(hipLOffset);
  kneeL.write(kneeLOffset);
  ankleL.write(ankleLOffset);
```

```
  hipR.write(hipROffset);
  kneeR.write(kneeROffset);
  ankleR.write(ankleROffset);

  delay(5000);

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Program odpowiedzialny za chód – humaniod_robot.ino:

```
#include <Servo.h>
#include "constants.h"

Servo hipL;
Servo hipR;
Servo kneeL;
Servo kneeR;
Servo ankleL;
Servo ankleR;

void updateServoPos(int target1, int target2, int target3, char leg){
  if (leg == 'l'){
    hipL.write(hipLOffset - target1);
    kneeL.write(kneeLOffset - target2);
    ankleL.write(2*ankleLOffset - target3);
```

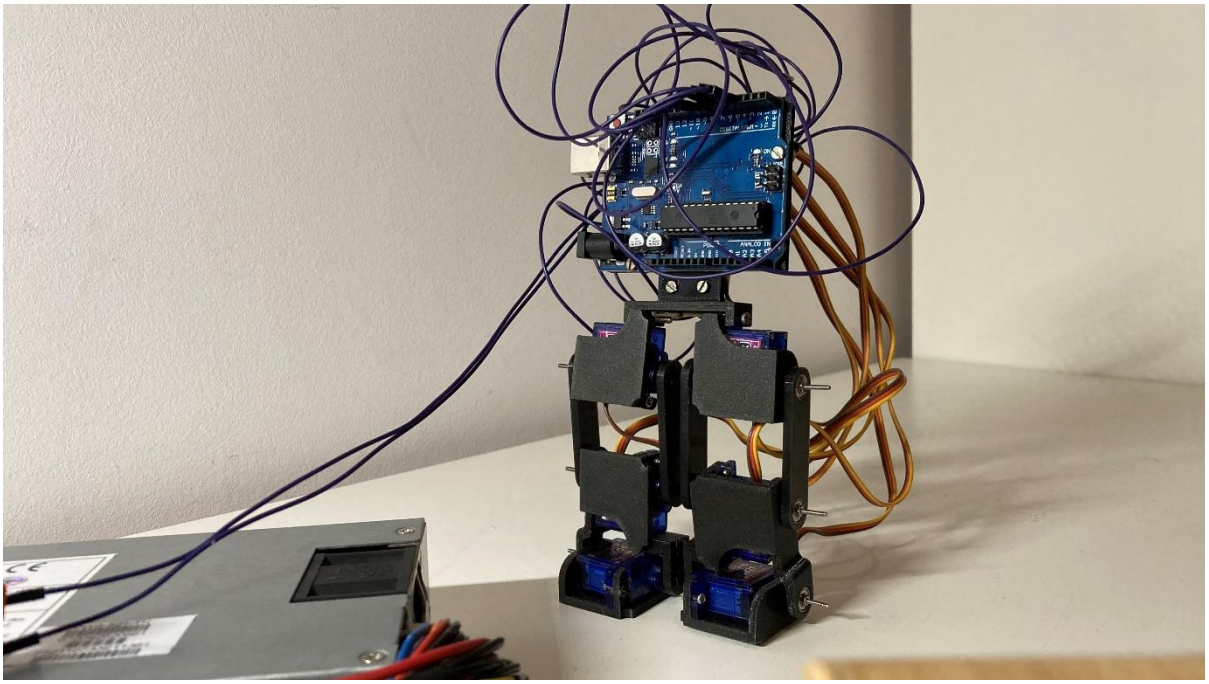
```
  }
  else if (leg == 'r'){
    hipR.write(hipROffset + target1);
    kneeR.write(kneeROffset - target2);
    ankleR.write(target3);
  }
}

void pos(float x, float z, char leg){
  float hipRad2 = atan(x/z);
  float hipDeg2 = hipRad2 * (180/PI);

  float z2 = z/cos(hipRad2);

  float hipRad1 = acos((sq(l1) + sq(z2) - sq(l2))/(2*l1*z2));
  float hipDeg1 = hipRad1 * (180/PI);

  float kneeRad = PI - acos((sq(l1) + sq(l2) - sq(z2))/(2*l1*l2));

  float ankleRad = PI/2 + hipRad2 - acos((sq(l2) + sq(z2) - sq(l1))/(2*l2*z2));

  float hipDeg = hipDeg1 + hipDeg2;
  float kneeDeg = kneeRad * (180/PI);
  float ankleDeg = ankleRad * (180/PI);

//  Serial.print(hipDeg);
//  Serial.print("\t");
//  Serial.print(kneeDeg);
```

```
//  Serial.print("\t");
//  Serial.println(ankleDeg);

  updateServoPos(hipDeg, kneeDeg, ankleDeg, leg);
}

void takeStep(float stepLength, int stepVelocity){
  for (float i = stepLength; i >= -stepLength; i-=0.5){
    pos(i, stepHeight, 'r');
    pos(-i, stepHeight - stepClearance, 'l');
    delay(stepVelocity);
  }

  for (float i = stepLength; i >= -stepLength; i-=0.5){
    pos(-i, stepHeight - stepClearance, 'r');
    pos(i, stepHeight, 'l');
    delay(stepVelocity);
  }
}

void initialize(){
  for (float i = 10.7; i >= stepHeight; i-=0.1){
    pos(0, i, 'l');
    pos(0, i, 'r');
  }
}

void setup() {
```

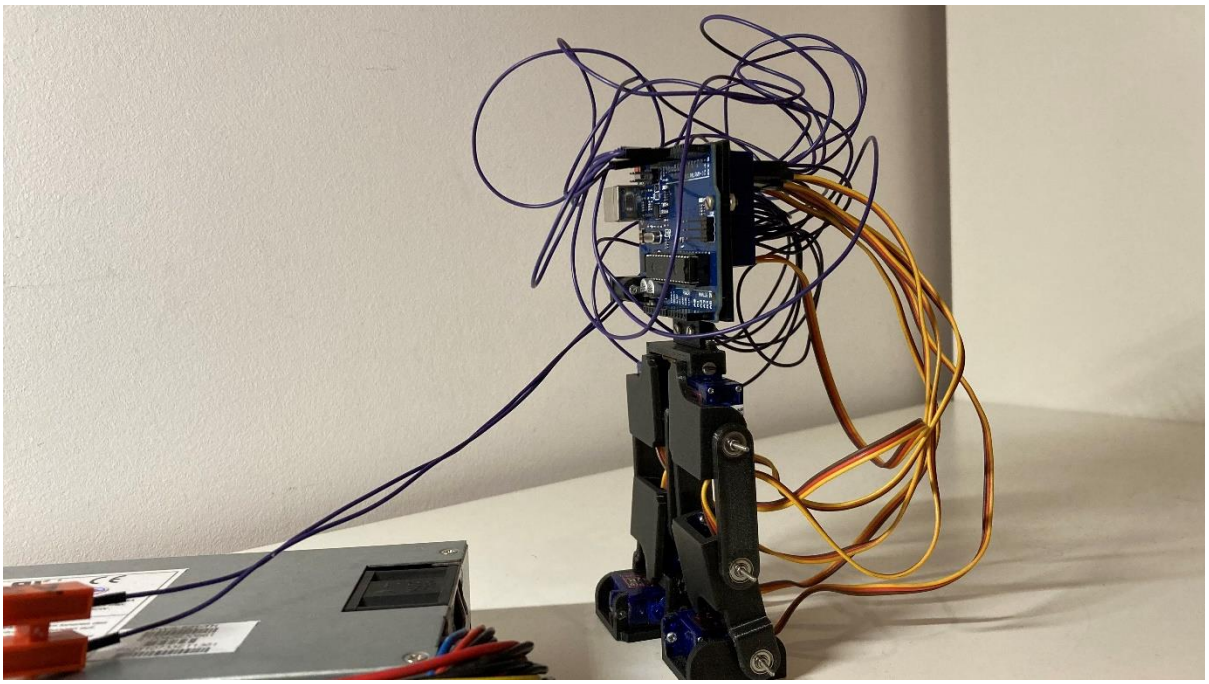
```
  // put your setup code here, to run once:
  Serial.begin(9600);
  hipL.attach(9);
  hipR.attach(8);
  kneeL.attach(7);
  kneeR.attach(6);
  ankleL.attach(5);
  ankleR.attach(4);

  hipL.write(hipLOffset);
  kneeL.write(kneeLOffset);
  ankleL.write(ankleLOffset);

  hipR.write(hipROffset);
  kneeR.write(kneeROffset);
  ankleR.write(ankleROffset);

  delay(5000);

  initialize();
}

void loop() {
  // put your main code here, to run repeatedly:
  takeStep(2, 0);
}
```

6. Podsumowanie

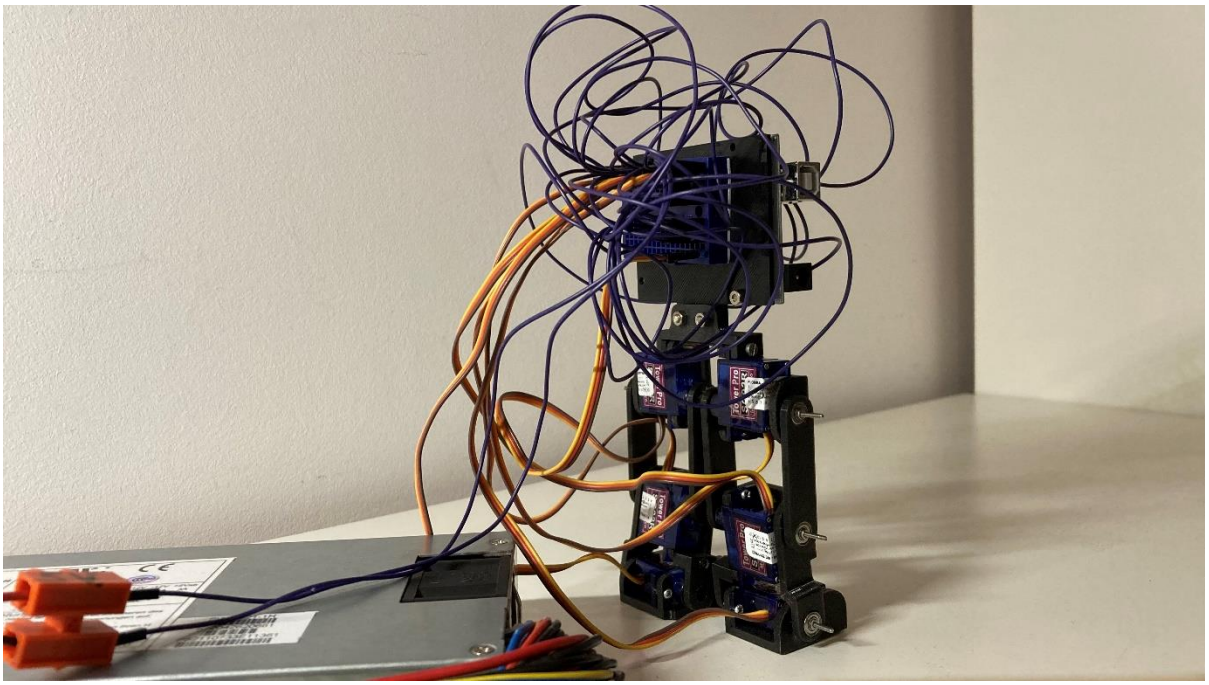
Skończony robot prezentuje się następująco:



Rysunek 31



Rysunek 32



Rysunek 33

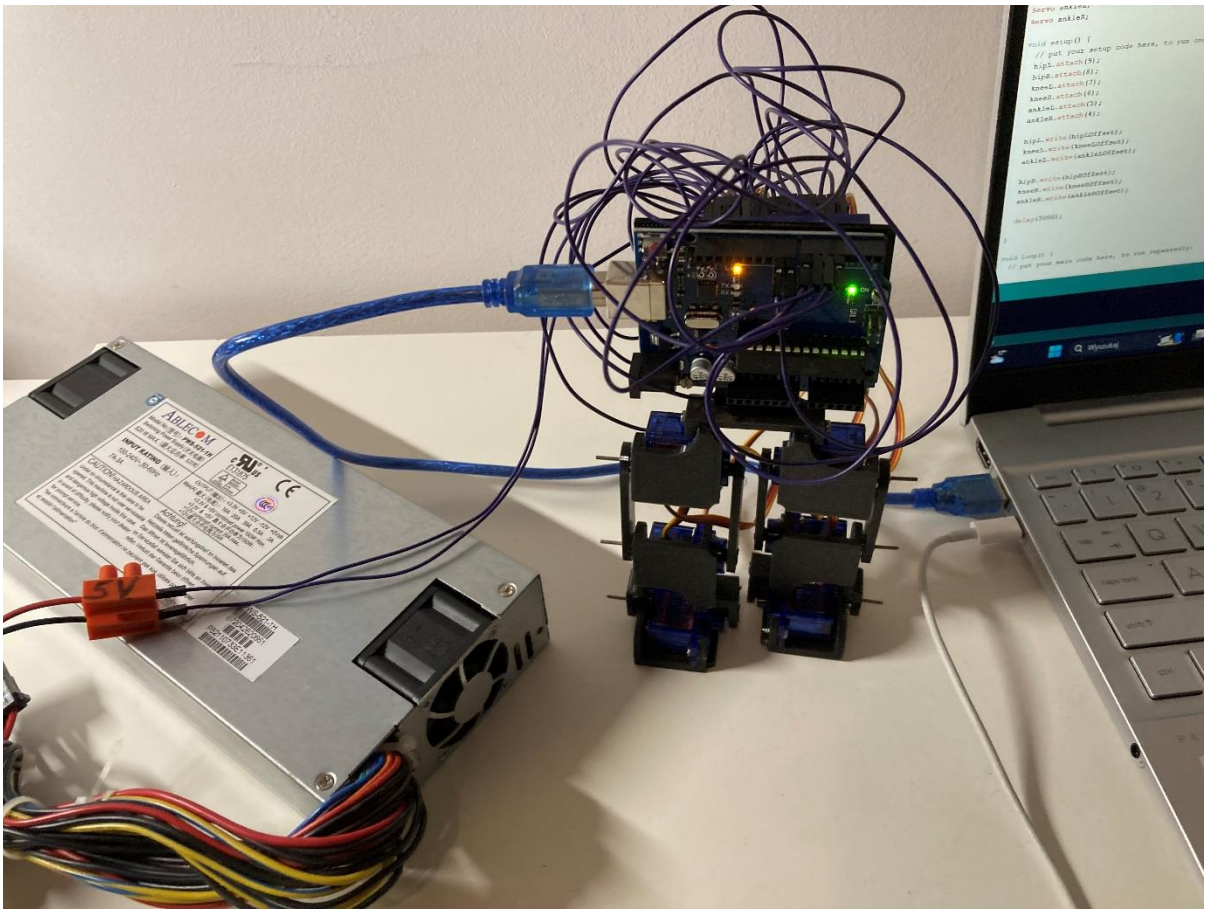
Do wykonanie robota niezbędna była wiedza z zakresu:

- Mechaniki
- Elektroniki
- Informayki

Możemy w związku z tym powiedzieć, że jest to projekt odpowiednio reprezentujący kierunek, jakim jest mechatronika.

Wykonanie tego projektu może nas nauczyć:

- Korzystania z programu do modelowania (np. NX)
- Tworzenie podstawowych układów elektronicznych
- Programowania płytek Arduino
- Dobierania elementów potrzebnych do wykonania danego układu elektronicznego
- Wykorzystywania mikro serw



Rysunek 34

Robot podłączony do zasilacza i do komputera z programem.

7. Przewodnik po githubie

Materiały do części mechanicznej:

Folder „model”:

- Folder „RYSUNKI” – zawiera rysunki elementów korpusu w formacie .png
- Folder „części” – zawiera zamodelowane elementy korpusu w formacie .prt (format programu NX)
- Plik „złożenie.prt” – złożenie zamodelowanych elementów

Materiały do części programistycznej:

Folder „program”:

- constants.h – plik nagłówkowy
- initial_setup.ino – pozycja startowa robota
- humanoid_robot.ino - chód

Dodatkowo:

- Link do filmu prezentującego robota
- Obrazek przedstawiający model robota
- Zdjęcie gotowego robota

8. Źródła

[1] https://www.instructables.com/Arduino-Controlled-Robotic-Biped/

[2] https://kamami.pl/plytki-stykowe/557414-prototypowa-plytka-stykowa-170-punktow-35x47-mm-kolor-niebieski.html

[3] https://allegro.pl/oferta/micro-serwo-9g-sg92r-sg92-sg-92r-sg-92-silne-2-5kg-8189680434?utm_feed=aa34192d-eee2-4419-9a9a-de66b9dfae24&ev_campaign_id=17966335829

[4] https://sklep.avt.pl/serwomechanizm-z-orczykami-sg92g-tower-pro-2-5kg.html